# SQream DB

***Release 2022.1***

## SQream Documentation

**Aug 27, 2023**

# CONTENTS:

SQream DB is a columnar analytic SQL database management system.

SQream DB supports regular SQL including *a substantial amount of ANSI SQL*, uses *serializable transactions*, and *scales horizontally* for concurrent statements.

Even a *basic SQream DB machine* can support tens to hundreds of terabytes of data.

SQream DB easily plugs in to third-party tools like Tableau comes with standard SQL client drivers, including *JDBC*, *ODBC*, and Python DB-API.

| **Get Started** | **Reference** | **Guides** |
|---|---|---|
| Getting Started<br>*SQL Feature Checklist*<br>*Bulk load CSVs* | *SQL Reference*<br>*SQL Statements*<br>*SQL Functions* | Setting up SQream<br>*Best practices* |
| **Releases** | **Driver and Deployment** | **Help and Support** |
| *2022.1*<br>*2021.2*<br>*2021.1*<br>*2020.3*<br>*2020.2*<br>*2020.1*<br>*All recent releases* | *Client drivers* | *Troubleshooting* guide<br>*Gathering Information for SQream Support* |

### Need help?

If you couldn't find what you're looking for, we're always happy to help. Visit SQream's support portal for additional support.

CHAPTER

ONE

# GETTING STARTED

The **Getting Started** page describes the following things you need to start using SQream:

## 1.1 Preparing Your Machine to Install SQream

To prepare your machine to install SQream, do the following:

- Set up your local machine according to SQream's recommended pre-installation configurations.

- Verify you have an NVIDIA-capable server, either on-premise or on supported cloud platforms:
  - Red Hat Enterprise Linux v7.x
  - CentOS v7.x
  - Amazon Linux 7
- Verify that you have the following:
  - An NVIDIA GPU - SQream recommends using a Tesla GPU.
  - An SSH connection to your server.
  - SUDO permissions for installation and configuration purposes.
  - A SQream license - Contact support@sqream.com or your SQream account manager for your license key.

For more information, see the following:

- *Recommended Pre-Installation Configuration*
- *Hardware Guide*

## 1.2 Installing SQream

The **Installing SQream** section includes the following SQream installation methods:

- Installing SQream natively - Describes installing SQream using binary packages provided by SQream.
- Installing SQream with Kubernetes - Describes installing SQream using the Kubernetes open source platform.
- Installing and running SQream in a Docker container - Describes how to run SQream in a Docker container.

## 1.3 Executing Statements in SQream

You can execute statements in SQream using one of the following tools:

- SQream SQL CLI - a command line interface
- SQream Acceleration Studio - an intuitive and easy-to-use interface.

## 1.4 Performing Basic SQream Operations

After installing SQream you can perform the operations described on this page:

### 1.4.1 Running the SQream SQL Client

The following example shows how to run the SQream SQL client:

```
$ sqream sql --port=5000 --username=rhendricks -d master
Password:

Interactive client mode
To quit, use ^D or \q.

master=> _
```

Running the SQream SQL client prompts you to provide your password. Use the username and password that you have set up, or your DBA has provided.

**Tip:**

- You can exit the shell by typing `\q` or `Ctrl-d`.
- A new SQream cluster contains a database named *master,* which is the database used in the examples on this page.

### 1.4.2 Creating Your First Table

The **Creating Your First Table** section describes the following:

- *Creating a table*
- *Replacing a table*
- *Listing a CREATE TABLE statement*
- *Dropping a table*

**Creating a Table**

The `CREATE TABLE` syntax is used to create your first table. This table includes a table name and column specifications, as shown in the following example:

```
CREATE TABLE cool_animals (
   id INT NOT NULL,
   name TEXT(20),
```

(continues on next page)

(continued from previous page)

```
   weight INT
);
```

For more information on creating a table, see create_table.

**Replacing a Table**

You can drop an existing table and create a new one by adding the `OR REPLACE` parameter after the `CREATE` keyword, as shown in the following example:

```
CREATE OR REPLACE TABLE cool_animals (
   id INT NOT NULL,
   name TEXT(20),
   weight INT
);
```

**Listing a CREATE TABLE Statement**

You can list the full, verbose `CREATE TABLE` statement for a table by using the **GET DDL** function with the table name as shown in the following example:

```
test=> SELECT GET_DDL('cool_animals');
create table "public"."cool_animals" (
"id" int not null,
"name" text(20),
"weight" int
);
```

**Note:**

- SQream DB identifier names such as table names and column names are not case sensitive. SQream DB lowercases all identifiers bu default. If you want to maintain case, enclose the identifiers with double-quotes.
- SQream DB places all tables in the *public* schema, unless another schema is created and specified as part of the table name.

For information on listing a `CREATE TABLE` statement, see get_ddl.

**Dropping a Table**

When you have finished working with your table, you can drop the table to remove it table and its content, as shown in the following example:

```
test=> DROP TABLE cool_animals;

executed
```

For more information on dropping tables, see drop_table.

### 1.4.3 Listing Tables

To see the tables in the current database you can query the catalog, as shown in the following example:

```
test=> SELECT table_name FROM sqream_catalog.tables;
cool_animals

1 rows
```

### 1.4.4 Inserting Rows

The **Inserting Rows** section describes the following:

- *Inserting basic rows*
- *Changing value order*
- *Inserting multiple rows*
- *Omitting columns*

**Inserting Basic Rows**

You can insert basic rows into a table using the `INSERT` statement. The inserted statement includes the table name, an optional list of column names, and column values listed in the same order as the column names, as shown in the following example:

```
test=> INSERT INTO cool_animals VALUES (1, 'Dog', 7);

executed
```

**Changing Value Order**

You can change the order of values by specifying the column order, as shown in the following example:

```
test=> INSERT INTO cool_animals(weight, id, name) VALUES (3, 2, 'Possum');

executed
```

**Inserting Multiple Rows**

You can insert multiple rows using the `INSERT` statement by using sets of parentheses separated by commas, as shown in the following example:

```
test=> INSERT INTO cool_animals VALUES
      (3, 'Cat', 5) ,
      (4, 'Elephant', 6500) ,
      (5, 'Rhinoceros', 2100);

executed
```

**Note:** You can load large data sets using bulk loading methods instead. For more information, see *Inserting Data Overview*.

**Omitting Columns**

Omitting columns that have a default values (including default `NULL` values) uses the default value, as shown in the following example:

```
test=> INSERT INTO cool_animals (id) VALUES (6);

executed
```

```
test=> INSERT INTO cool_animals (id) VALUES (6);

executed
test=> SELECT * FROM cool_animals;
1,Dog                 ,7
2,Possum              ,3
3,Cat                 ,5
4,Elephant            ,6500
5,Rhinoceros          ,2100
6,\N,\N

6 rows
```

**Note:** Null row values are represented as `\N`

For more information on inserting rows, see insert.

For more information on default values, see default value.

### 1.4.5 Running Queries

The **Running Queries** section describes the following:

- *Running basic queries*
- *Outputting all columns*
- *Outputting shorthand table values*
- *Filtering results*
- *Sorting results*
- *Filtering null rows*

**Running Basic Queries**

You can run a basic query using the `SELECT` keyword, followed by a list of columns and values to be returned, and the table to get the data from, as shown in the following example:

```
test=> SELECT id, name, weight FROM cool_animals;
1,Dog                 ,7
2,Possum              ,3
3,Cat                 ,5
4,Elephant            ,6500
5,Rhinoceros          ,2100
6,\N,\N

6 rows
```

For more information on the `SELECT` keyword, see select.

**To Output All Columns**

You can output all columns without specifying them using the star operator `*`, as shown in the following example:

```
test=> SELECT * FROM cool_animals;
1,Dog                   ,7
2,Possum                ,3
3,Cat                   ,5
4,Elephant              ,6500
5,Rhinoceros            ,2100
6,\N,\N

6 rows
```

### Outputting Shorthand Table Values

You can output the number of values in a table without getting the full result set by using the `COUNT` statement:

```
test=> SELECT COUNT(*) FROM cool_animals;
6

1 row
```

### Filtering Results

You can filter results by adding a `WHERE` clause and specifying the filter condition, as shown in the following example:

```
test=> SELECT id, name, weight FROM cool_animals WHERE weight > 1000;
4,Elephant              ,6500
5,Rhinoceros            ,2100

2 rows
```

### Sorting Results

You can sort results by adding an `ORDER BY` clause and specifying ascending (`ASC`) or descending (`DESC`) order, as shown in the following example:

```
test=> SELECT * FROM cool_animals ORDER BY weight DESC;
4,Elephant              ,6500
5,Rhinoceros            ,2100
1,Dog                   ,7
3,Cat                   ,5
2,Possum                ,3
6,\N,\N

6 rows
```

### Filtering Null Rows

You can filter null rows by adding an `IS NOT NULL` filter, as shown in the following example:

```
test=> SELECT * FROM cool_animals WHERE weight IS NOT NULL ORDER BY weight DESC;
4,Elephant              ,6500
5,Rhinoceros            ,2100
1,Dog                   ,7
3,Cat                   ,5
2,Possum                ,3

5 rows
```

For more information, see the following:

- Outputting the number of values in a table without getting the full result set - COUNT(*).
- Filtering results - WHERE
- Sorting results - ORDER BY
- Filtering rows - IS NOT NULL

### 1.4.6 Deleting Rows

The **Deleting Rows** section describes the following:

- *Deleting selected rows*
- *Deleting all rows*

**Deleting Selected Rows**

You can delete rows in a table selectively using the `DELETE` command. You must include a table name and *WHERE* clause to specify the rows to delete, as shown in the following example:

```
test=> DELETE FROM cool_animals WHERE weight is null;

executed
master=> SELECT  * FROM cool_animals;
1,Dog                   ,7
2,Possum                ,3
3,Cat                   ,5
4,Elephant              ,6500
5,Rhinoceros            ,2100

5 rows
```

**Deleting All Rows**

You can delete all rows in a table using the `TRUNCATE` command followed by the table name, as shown in the following example:

```
test=> TRUNCATE TABLE cool_animals;

executed
```

**Note:** While truncate deletes data from disk immediately, delete does not physically remove the deleted rows.

For more information, see the following:

- Deleting selected rows - DELETE
- Deleting all rows - TRUNCATE

### 1.4.7 Saving Query Results to a CSV or PSV File

You can save query results to a CSV or PSV file using the `sqream sql` command from a CLI client. This saves your query results to the selected delimited file format, as shown in the following example:

```
$ sqream sql --username=mjordan --database=nba --host=localhost --port=5000 -c
↪"SELECT * FROM nba LIMIT 5" --results-only --delimiter='|' > nba.psv
$ cat nba.psv
Avery Bradley           |Boston Celtics        |0|PG|25|6-2 |180|Texas
↪|7730337
Jae Crowder             |Boston Celtics        |99|SF|25|6-6 |235|Marquette
↪ |6796117
John Holland            |Boston Celtics        |30|SG|27|6-5 |205|Boston University
↪ |\N
R.J. Hunter             |Boston Celtics        |28|SG|22|6-5 |185|Georgia State
↪ |1148640
Jonas Jerebko           |Boston Celtics        |8|PF|29|6-10|231|\N|5000000
```

For more output options, see *Controlling the Client Output*.

#### What's next?

- Explore all of SQream DB's *SQL Syntax*.
- See the full *SQream SQL CLI reference*.
- Connect a third party tool to start analyzing data.

For more information on other basic SQream operations, see the following:

- Creating a Database
- *Data Ingestion Sources*

## 1.5 Hardware Guide

The **Hardware Guide** describes the SQream reference architecture, emphasizing the benefits to the technical audience, and provides guidance for end-users on selecting the right configuration for a SQream installation.

#### Need help?

This page is intended as a "reference" to suggested hardware. However, different workloads require different solution sizes. SQream's experienced customer support has the experience to advise on these matters to ensure the best experience.

Visit SQream's support portal for additional support.

### 1.5.1 A SQream Cluster

SQream recommends rackmount servers by server manufacturers Dell, Lenovo, HP, Cisco, Supermicro, IBM, and others.

A typical SQream cluster includes one or more nodes, consisting of

- Two-socket enterprise processors, like the Intel® Xeon® Gold processor family or an IBM® POWER9 processors, providing the high performance required for compute-bound database workloads.

- NVIDIA Tesla GPU accelerators, with up to 5,120 CUDA and Tensor cores, running on PCIe or fast NVLINK busses, delivering high core count, and high-throughput performance on massive datasets.

- High density chassis design, offering between 2 and 4 GPUs in a 1U, 2U, or 3U package, for best-in-class performance per $cm^2$.

#### 1.5.1.1 Single-Node Cluster Example

A single-node SQream cluster can handle between 1 and 8 concurrent users, with up to 1PB of data storage (when connected via NAS).

An average single-node cluster can be a rackmount server or workstation, containing the following components:

| Component | Type |
|---|---|
| Server | Dell R750, Dell R940xa, HP ProLiant DL380 Gen10 or similar (Intel only) |
| Processor | 2x Intel Xeon Gold 6240 (18C/36HT) 2.6GHz or similar |
| RAM | 1.5 TB |
| Onboard storage | • 2x 960GB SSD 2.5in hot plug for OS, RAID1<br>• 2x 2TB SSD or NVMe, for temporary spooling, RAID1<br>• 10x 3.84TB SSD 2.5in Hot plug for storage, RAID6 |
| GPU | 2x A100 NVIDIA |
| Operating System | Red Hat Enterprise Linux v7.x or CentOS v7.x or Amazon Linux |

**Note:** If you are using internal storage, your volumes must be formatted as xfs.

In this system configuration, SQream can store about 200TB of raw data (assuming average compression ratio and ~50TB of usable raw storage).

If a NAS is used, the 14x SSD drives can be omitted, but SQream recommends 2TB of local spool space on SSD or NVMe drives.

### 1.5.1.2 Multi-Node Cluster Examples

Multi-node clusters can handle any number of concurrent users. A typical SQream cluster relies on a minimum of two GPU-enabled servers and shared storage connected over a network fabric, such as InfiniBand EDR, 40GbE, or 100GbE.

The **Multi-Node Cluster Examples** section describes the following specifications:

- *Hardware Specifications*
- *Metadata Specifications*

#### 1.5.1.2.1 Hardware Specifications

The following table shows SQream's recommended hardware specifications:

| Component | Type |
|---|---|
| Server | Dell R750, Dell R940xa, HP ProLiant DL380 Gen10 or similar (Intel only) |
| Processor | 2x Intel Xeon Gold 6240 (18C/36HT) 2.6GHz or similar |
| RAM | 2 TB |
| Onboard storage | • 2x 960GB SSD 2.5in hot plug for OS, RAID1<br>• 2x 2TB SSD or NVMe, for temporary spooling, RAID1 |
| External Storage | • Mellanox Connectx5/6 100G NVIDIA Network Card (if applicable) or other high speed network card minimum 40G compatible to customer's infrastructure<br>• 50 TB (NAS connected over GPFS, Lustre, or NFS) GPFS recommended |
| GPU | 2x A100 NVIDIA |
| Operating System | Red Hat Enterprise Linux v7.x or CentOS v7.x or Amazon Linux |

#### 1.5.1.2.2 Metadata Specifications

The following table shows SQream's recommended metadata server specifications:

| Component | Type |
|---|---|
| Processors | Two Intel Xeon Gold 6342 2.8 Ghz 24C processors or similar |
| RAM | 512GB DDR4 RAM 8x64GB RDIMM or similar |
| Discs | Two 960 GB MVMe SSD drives in RAID 1 or similar |
| Network Card (Storage) | Two Mellanox ConnectX-6 Single Port HDR VPI InfiniBand Adapter cards at 100GbE or similar. |
| Network Card (Corporate) | Two 1 GbE cards or similar |
| Power sources | Two Power Supplies - 800W AC 50/60Hz 100~240Vac/9.2-4.7A, 3139 BTU/hr |
| Operating System | Red Hat Enterprise Linux v7.x or CentOS v7.x or Amazon Linux |

**Note:** With a NAS connected over GPFS, Lustre, or NFS, each SQream worker can read data at up to 5GB/s.

#### 1.5.1.3 SQream Studio Server Example

The following table shows SQream's recommended Studio server specifications:

| Component | Type |
|---|---|
| Server | Physical or virtual machine |
| Processor | 1x Intel Core i7 |
| RAM | 16 GB |
| Onboard storage | 50 GB SSD 2.5in Hot plug for OS, RAID1 |
| Operating System | Red Hat Enterprise Linux v7.x or CentOS v7.x |

### 1.5.2 Cluster Design Considerations

This section describes the following cluster design considerations:

- In a SQream installation, the storage and compute are logically separated. While they may reside on the same machine in a standalone installation, they may also reside on different hosts, providing additional flexibility and scalability.
- SQream uses all resources in a machine, including CPU, RAM, and GPU to deliver the best performance. At least 256GB of RAM per physical GPU is recommended.
- Local disk space is required for good temporary spooling performance, particularly when performing intensive operations exceeding the available RAM, such as sorting. SQream recommends an SSD or NVMe drive in RAID 1 configuration with about twice the RAM size available for temporary storage. This can be shared with the operating system drive if necessary.
- When using SAN or NAS devices, SQream recommends approximately 5GB/s of burst throughput from storage per GPU.

#### 1.5.2.1 Balancing Cost and Performance

Prior to designing and deploying a SQream cluster, a number of important factors must be considered.

The **Balancing Cost and Performance** section provides a breakdown of deployment details to ensure that this installation exceeds or meets the stated requirements. The rationale provided includes the necessary information for modifying configurations to suit the customer use-case scenario, as shown in the following table:

| Component | Value |
|---|---|
| Compute - CPU | Balance price and performance |
| Compute – GPU | Balance price with performance and concurrency |
| Memory – GPU RAM | Balance price with concurrency and performance. |
| Memory - RAM | Balance price and performance |
| Operating System | Availability, reliability, and familiarity |
| Storage | Balance price with capacity and performance |
| Network | Balance price and performance |

#### 1.5.2.2 CPU Compute

SQream relies on multi-core Intel Gold Xeon processors or IBM POWER9 processors, and recommends a dual-socket machine populated with CPUs with 18C/36HT or better. While a higher core count may not necessarily affect query performance, more cores will enable higher concurrency and better load performance.

#### 1.5.2.3 GPU Compute and RAM

The NVIDIA Tesla range of high-throughput GPU accelerators provides the best performance for enterprise environments. Most cards have ECC memory, which is crucial for delivering correct results every time. SQream recommends the NVIDIA Tesla V100 32GB or NVIDIA Tesla A100 40GB GPU for best performance and highest concurrent user support.

GPU RAM, sometimes called GRAM or VRAM, is used for processing queries. It is possible to select GPUs with less RAM, like the NVIDIA Tesla V100 16GB or P100 16GB, or T4 16GB. However, the smaller GPU RAM results in reduced concurrency, as the GPU RAM is used extensively in operations like JOINs, ORDER BY, GROUP BY, and all SQL transforms.

#### 1.5.2.4 RAM

SQream requires using **Error-Correcting Code memory (ECC)**, standard on most enterprise servers. Large amounts of memory are required for improved performance for heavy external operations, such as sorting and joining.

SQream recommends at least 256GB of RAM per GPU on your machine.

#### 1.5.2.5 Operating System

SQream can run on the following 64-bit Linux operating systems:

- Red Hat Enterprise Linux (RHEL) v7
- CentOS v7
- Amazon Linux 2018.03
- Other Linux distributions may be supported via nvidia-docker

#### 1.5.2.6 Storage

For clustered scale-out installations, SQream relies on NAS/SAN storage. For stand-alone installations, SQream relies on redundant disk configurations, such as RAID 5, 6, or 10. These RAID configurations replicate blocks of data between disks to avoid data loss or system unavailability.

SQream recommends using enterprise-grade SAS SSD or NVMe drives. For a 32-user configuration, the number of GPUs should roughly match the number of users. SQream recommends 1 Tesla V100 or A100 GPU per 2 users, for full, uninterrupted dedicated access.

Download the full SQream Reference Architecture document.

CHAPTER

TWO

# INSTALLATION GUIDES

Before you get started using SQream, consider your business needs and available resources. SQream was designed to run in a number of environments, and to be installed using different methods depending on your requirements. This determines which installation method to use.

The **Installation Guides** section describes the following installation guide sets:

## 2.1 Installing and Launching SQream

The **Installing and Launching SQream** page includes the following installation guides:

### 2.1.1 Recommended Pre-Installation Configuration

Before installing SQream DB, SQream recommends you to tune your system for better performance and stability.

This page provides recommendations for production deployments of SQream and describes the following:



#### 2.1.1.1 Recommended BIOS Settings

The first step when setting your pre-installation configurations is to use the recommended BIOS settings.

The BIOS settings may have a variety of names, or may not exist on your system. Each system vendor has a different set of settings and variables. It is safe to skip any and all of the configuration steps, but this may impact performance.

If any doubt arises, consult the documentation for your server or your hardware vendor for the correct way to apply the settings.

| Item | Setting | Rationale |
|---|---|---|
| **Management console access** | **Connected** | Connection to OOB required to preserve continuous network uptime. |
| **All drives** | **Connected and displayed on RAID interface** | Prerequisite for cluster or OS installation. |
| **RAID volumes.** | **Configured according to project guidelines. Must be rebooted to take effect.** | Clustered to increase logical volume and provide redundancy. |
| **Fan speed Thermal Configuration.** | Dell fan speed: **High Maximum**. Specified minimum setting: **60**. HPe thermal configuration: **Increased cooling**. | NVIDIA Tesla GPUs are passively cooled and require high airflow to operate at full performance. |
| **Power regulator or iDRAC power unit policy** | HPe: **HP static high performance** mode enabled. Dell: **iDRAC power unit policy** (power cap policy) disabled. | Other power profiles (such as "balanced") throttle the CPU and diminishes performance. Throttling may also cause GPU failure. |
| **System Profile**, **Power Profile**, or **Performance Profile** | **High Performance** | The Performance profile provides potentially increased performance by maximizing processor frequency, and the disabling certain power saving features such as C-states. Use this setting for environments that are not sensitive to power consumption. |
| **Power Cap Policy** or **Dynamic power capping** | **Disabled** | Other power profiles (like "balanced") throttle the CPU and may diminish performance or cause GPU failure. This setting may appear together with the above (Power profile or Power regulator). This setting allows disabling system ROM power calibration during the boot process. Power regulator settings are named differently in BIOS and iLO/iDRAC. |
| **Intel Turbo Boost** | **Enabled** | Intel Turbo Boost enables overclocking the processor to boost CPU-bound operation performance. Overclocking may risk computational jitter due to changes in the processor's turbo frequency. This causes brief pauses in processor operation, introducing uncertainty into application processing time. Turbo operation is a function of power consumption, processor temperature, and the number of active cores. |
| **Logical Processor** | **HPe**: Enable **Hyperthreading Dell**: Enable **Logical Processor** | Hyperthreading doubles the amount of logical processors, which may improve performance by ~5-10% for CPU-bound operations. |
| **Intel Virtualization Technology** (VT-d) | **Disable** | VT-d is optimal for running VMs. However, when running Linux natively, disabling VT-d boosts performance by up to 10%. |
| **Logical Processor** | **HPe**: Enable **Hyperthreading Dell**: Enable **Logical Processor** | Hyperthreading doubles the amount of logical processors, which may improve performance by ~5-10% for CPU-bound operations. |
| **Intel Virtualization Technology** (VT-d) | **Disable** | VT-d is optimal for running VMs. However, when running Linux natively, disabling VT-d boosts performance by up to 10%. |
| **Processor C-States** (Minimum processor idle power core state) | **Disable** | Processor C-States reduce server power when the system is in an idle state. This causes slower cold-starts when the system transitions from an idle to a load state, and may reduce query performance by up to 15%. |
| **HPe**: **Energy/Performance bias** | **Maximum performance** | Configures processor sub-systems for high-performance and low-latency. Other power profiles (like "balanced") throttle the CPU and may diminish performance. Use this setting for environments that are not sensitive to power |

#### 2.1.1.2 Installing the Operating System

Once the BIOS settings have been set, you must install the operating system. Either the CentOS (versions 7.6-7.9) or RHEL (versions 7.6-7.9) must be installed before installing the SQream database, by either the customer or a SQream representative.

**To install the operating system:**

1. Select a language (English recommended).
2. From **Software Selection**, select **Minimal**.
3. Select the **Development Tools** group checkbox.
4. Continue the installation.
5. Set up the necessary drives and users as per the installation process.

   Using Debugging Tools is recommended for future problem-solving if necessary.

Selecting the **Development Tools** group installs the following tools:

- autoconf
- automake
- binutils
- bison
- flex
- gcc
- gcc-c++
- gettext
- libtool
- make
- patch
- pkgconfig
- redhat-rpm-config
- rpm-build
- rpm-sign

The root user is created and the OS shell is booted up.

#### 2.1.1.3 Configuring the Operating System

Once you've installted your operation system, you can configure it. When configuring the operating system, several basic settings related to creating a new server are required. Configuring these as part of your basic set-up increases your server's security and usability.

#### 2.1.1.3.1 Logging In to the Server

You can log in to the server using the server's IP address and password for the **root** user. The server's IP address and **root** user were created while installing the operating system above.

#### 2.1.1.3.2 Automatically Creating a SQream User

**To automatically create a SQream user:**

1. If a SQream user was created during installation, verify that the same ID is used on every server:

   ```
   $ sudo id sqream
   ```

   The ID **1000** is used on each server in the following example:

   ```
   $ uid=1000(sqream) gid=1000(sqream) groups=1000(sqream)
   ```

2. If the ID's are different, delete the SQream user and SQream group from both servers:

   ```
   $ sudo userdel sqream
   ```

3. Recreate it using the same ID:

   ```
   $ sudo rm /var/spool/mail/sqream
   ```

#### 2.1.1.3.3 Manually Creating a SQream User

**To manually create a SQream user:**

SQream enables you to manually create users. This section shows you how to manually create a user with the UID **1111**. You cannot manually create during the operating system installation procedure.

1. Add a user with an identical UID on all cluster nodes:

   ```
   $ useradd -u 1111 sqream
   ```

2. Add the user **sqream** to the **wheel** group.

   ```
   $ sudo usermod -aG wheel sqream
   ```

   You can remove the SQream user from the **wheel** group when the installation and configuration are complete:

   ```
   $ passwd sqream
   ```

3. Log out and log back in as **sqream**.

   **Note:** If you deleted the **sqream** user and recreated it with different ID, to avoid permission errors, you must change its ownership to /home/sqream.

4. Change the **sqream** user's ownership to /home/sqream:

   ```
   $ sudo chown -R sqream:sqream /home/sqream
   ```

#### 2.1.1.3.4 Setting Up A Locale

SQream enables you to set up a locale. In this example, the locale used is your own location.

**To set up a locale:**

1. Set the language of the locale:

```
$ sudo localectl set-locale LANG=en_US.UTF-8
```

2. Set the time stamp (time and date) of the locale:

```
$ sudo timedatectl set-timezone Asia/Jerusalem
```

If needed, you can run the **timedatectl list-timezones** command to see your current time-zone.

#### 2.1.1.3.5 Installing the Required Packages

You can install the required packages by running the following command:

```
$ sudo yum install ntp pciutils monit zlib-devel openssl-devel kernel-devel-$(uname -
↪r) kernel-headers-$(uname -r) gcc net-tools wget jq
```

#### 2.1.1.3.6 Installing the Recommended Tools

You can install the recommended tools by running the following command:

```
$ sudo yum install bash-completion.noarch vim-enhanced vim-common net-tools iotop␣
↪htop psmisc screen xfsprogs wget yum-utils deltarpm dos2unix
```

#### 2.1.1.3.7 Installing Python 3.6.7

1. Download the Python 3.6.7 source code tarball file from the following URL into the **/home/sqream** directory:

```
$ wget https://www.python.org/ftp/python/3.6.7/Python-3.6.7.tar.xz
```

2. Extract the Python 3.6.7 source code into your current directory:

```
$ tar -xf Python-3.6.7.tar.xz
```

3. Navigate to the Python 3.6.7 directory:

```
$ cd Python-3.6.7
```

4. Run the **./configure** script:

```
$ ./configure
```

5. Build the software:

```
$ make -j30
```

6. Install the software:

```
$ sudo make install
```

7. Verify that Python 3.6.7 has been installed:

```
$ python3
```

#### 2.1.1.3.8 Installing NodeJS on CentOS

**To install the node.js on CentOS:**

1. Download the setup_12.x file as a root user logged in shell:

```
$ curl -sL https://rpm.nodesource.com/setup_12.x | sudo bash -
```

2. Clear the YUM cache and update the local metadata:

```
$ sudo yum clean all && sudo yum makecache fast
```

3. Install the **node.js** file:

```
$ sudo yum install -y nodejs
```

4. Install npm and make it available for all users:

```
$ sudo npm install pm2 -g
```

#### 2.1.1.3.9 Installing NodeJS on Ubuntu

**To install the node.js file on Ubuntu:**

1. Download the setup_12.x file as a root user logged in shell:

```
$ curl -sL https://rpm.nodesource.com/setup_12.x | sudo bash -
```

2. Install the node.js file:

```
$ sudo apt-get install -y nodejs
```

3. Install npm and make it available for all users:

```
$ sudo npm install pm2 -g
```

#### 2.1.1.3.10 Installing NodeJS Offline

**To install NodeJS Offline**

1. Download the NodeJS source code tarball file from the following URL into the **/home/sqream** directory:

```
$ wget https://nodejs.org/dist/v12.13.0/node-v12.13.0-linux-x64.tar.xz
```

2. Move the node-v12.13.0-linux-x64 file to the */usr/local* directory.

```
$ sudo mv node-v12.13.0-linux-x64  /usr/local
```

3. Navigate to the */usr/bin/* directory:

```
$ cd /usr/bin
```

4. Create a symbolic link to the */local/node-v12.13.0-linux-x64/bin/node node* directory:

```
$ sudo ln -s ../local/node-v12.13.0-linux-x64/bin/node node
```

5. Create a symbolic link to the */local/node-v12.13.0-linux-x64/bin/npm npm* directory:

```
$ sudo ln -s ../local/node-v12.13.0-linux-x64/bin/npm npm
```

6. Create a symbolic link to the */local/node-v12.13.0-linux-x64/bin/npx npx* directory:

```
$ sudo ln -s ../local/node-v12.13.0-linux-x64/bin/npx npx
```

7. Verify that the node versions for the above are correct:

```
$ node --version
```

#### 2.1.1.3.11 Installing the pm2 Service Offline

**To install the pm2 Service Offline**

1. On a machine with internet access, install the following:
   - nodejs
   - npm
   - pm2
2. Extract the pm2 module to the correct directory:

```
$ cd /usr/local/node-v12.13.0-linux-x64/lib/node_modules
$ tar -czvf pm2_x86.tar.gz pm2
```

3. Copy the **pm2_x86.tar.gz** file to a server without access to the internet and extract it.

4. Move the **pm2** folder to the */usr/local/node-v12.13.0-linux-x64/lib/node_modules* directory:

```
$ sudo mv pm2 /usr/local/node-v12.13.0-linux-x64/lib/node_modules
```

5. Navigate back to the */usr/bin* directory:

```
$ cd /usr/bin again
```

6. Create a symbolink to the **pm2** service:

```
$ sudo ln -s /usr/local/node-v12.22.3-linux-x64/lib/node_modules/pm2/bin/pm2␣
↪pm2
```

7. Verify that installation was successful:

```
$ pm2 list
```

**Note:** This must be done as a **sqream** user, and not as a **sudo** user.

8. Verify that the node version is correct:

```
$ node -v
```

#### 2.1.1.3.12 Configuring the Network Time Protocol

This section describes how to configure your **Network Time Protocol (NTP)**.

If you don't have internet access, see Configure NTP Client to Synchronize with NTP Server.

**To configure your NTP:**

1. Install the NTP file.

```
$ sudo yum install ntp
```

2. Enable the **ntpd** program.

```
$ sudo systemctl enable ntpd
```

3. Start the **ntdp** program.

```
$ sudo systemctl start ntpd
```

4. Print a list of peers known to the server and a summary of their states.

```
$ sudo ntpq -p
```

#### 2.1.1.3.13 Configuring the Network Time Protocol Server

If your organization has an NTP server, you can configure it.

**To configure your NTP server:**

1. Output your NTP server address and append `/etc/ntpd.conf` to the outuput.

```
$ echo -e "\nserver <your NTP server address>\n" | sudo tee -a /etc/ntp.conf
```

2. Restart the service.

```
$ sudo systemctl restart ntpd
```

3. Check that synchronization is enabled:

```
$ sudo timedatectl
```

Checking that synchronization is enabled generates the following output:

```
 $ Local time: Sat 2019-10-12 17:26:13 EDT
 Universal time: Sat 2019-10-12 21:26:13 UTC
       RTC time: Sat 2019-10-12 21:26:13
      Time zone: America/New_York (EDT, -0400)
    NTP enabled: yes
NTP synchronized: yes
RTC in local TZ: no
     DST active: yes
Last DST change: DST began at
                 Sun 2019-03-10 01:59:59 EST
                 Sun 2019-03-10 03:00:00 EDT
Next DST change: DST ends (the clock jumps one hour backwards) at
                 Sun 2019-11-03 01:59:59 EDT
                 Sun 2019-11-03 01:00:00 EST
```

#### 2.1.1.3.14 Configuring the Server to Boot Without the UI

You can configure your server to boot without a UI in cases when it is not required (recommended) by running the following command:

```
$ sudo systemctl set-default multi-user.target
```

Running this command activates the **NO-UI** server mode.

#### 2.1.1.3.15 Configuring the Security Limits

The security limits refers to the number of open files, processes, etc.

You can configure the security limits by running the **echo -e** command as a root user logged in shell:

```
$ sudo bash
```

```
$ echo -e "sqream soft nproc 1000000\nsqream hard nproc 1000000\nsqream soft nofile␣
↪1000000\nsqream hard nofile 1000000\nsqream soft core unlimited\nsqream hard core␣
↪unlimited" >> /etc/security/limits.conf
```

#### 2.1.1.3.16 Configuring the Kernel Parameters

**To configure the kernel parameters:**

1. Insert a new line after each kernel parameter:

   ```
   $ echo -e "vm.dirty_background_ratio = 5 \n vm.dirty_ratio = 10 \n vm.swappiness␣
   ↪= 10 \n vm.vfs_cache_pressure = 200 \n vm.zone_reclaim_mode = 0 \n" >> /etc/
   ↪sysctl.conf
   ```

   **Note:** In the past, the **vm.zone_reclaim_mode** parameter was set to **7.** In the latest Sqream version, the vm.zone_reclaim_mode parameter must be set to **0**. If it is not set to **0**, when a numa node runs out of memory, the system will get stuck and will be unable to pull memory from other numa nodes.

2. Check the maximum value of the **fs.file**.

```
$ sysctl -n fs.file-max
```

3. If the maximum value of the **fs.file** is smaller than **2097152**, run the following command:

```
$ echo "fs.file-max=2097152" >> /etc/sysctl.conf
```

**IP4 forward** must be enabled for Docker and K8s installation only.

4. Run the following command:

```
$ sudo echo “net.ipv4.ip_forward = 1” >> /etc/sysctl.conf
```

5. Reboot your system:

```
$ sudo reboot
```

#### 2.1.1.3.17 Configuring the Firewall

The example in this section shows the open ports for four sqreamd sessions. If more than four are required, open the required ports as needed. Port 8080 in the example below is a new UI port.

**To configure the firewall:**

1. Start the service and enable FirewallID on boot:

```
$ systemctl start firewalld
```

2. Add the following ports to the permanent firewall:

```
$ firewall-cmd --zone=public --permanent --add-port=8080/tcp
$ firewall-cmd --zone=public --permanent --add-port=3105/tcp
$ firewall-cmd --zone=public --permanent --add-port=3108/tcp
$ firewall-cmd --zone=public --permanent --add-port=5000-5003/tcp
$ firewall-cmd --zone=public --permanent --add-port=5100-5103/tcp
$ firewall-cmd --permanent --list-all
```

3. Reload the firewall:

```
$ firewall-cmd --reload
```

4. Enable FirewallID on boot:

```
$ systemctl enable firewalld
```

If you do not need the firewall, you can disable it:

```
$ sudo systemctl disable firewalld
```

#### 2.1.1.3.18 Disabling selinux

**To disable selinux:**

1. Show the status of **selinux**:

```
$ sudo sestatus
```

2. If the output is not **disabled**, edit the **/etc/selinux/config** file:

```
$ sudo vim /etc/selinux/config
```

3. Change **SELINUX=enforcing** to **SELINUX=disabled**.

   The above changes will only take effect after rebooting the server.

   You can disable selinux immediately after rebooting the server by running the following command:

```
$ sudo setenforce 0
```

#### 2.1.1.3.19 Configuring the /etc/hosts File

**To configure the /etc/hosts file:**

1. Edit the **/etc/hosts** file:

```
$ sudo vim /etc/hosts
```

2. Call your local host:

```
$ 127.0.0.1        localhost
$ <server1 ip>     <server_name>
$ <server2 ip>     <server_name>
```

#### 2.1.1.3.20 Configuring the DNS

**To configure the DNS:**

1. Run the **ifconfig** commasnd to check your NIC name. In the following example, **eth0** is the NIC name:

```
$ sudo vim /etc/sysconfig/network-scripts/ifcfg-eth0
```

2. Replace the DNS lines from the example above with your own DNS addresses :

```
$ DNS1="4.4.4.4"
$ DNS2="8.8.8.8"
```

#### 2.1.1.4 Installing the Nvidia CUDA Driver

After configuring your operating system, you must install the Nvidia CUDA driver.

> **Warning:** If your UI runs on the server, the server must be stopped before installing the CUDA drivers.

##### 2.1.1.4.1 CUDA Driver Prerequisites

1. Verify that the NVIDIA card has been installed and is detected by the system:

```
$ lspci | grep -i nvidia
```

2. Check which version of gcc has been installed:

```
$ gcc --version
```

3. If gcc has not been installed, install it for one of the following operating systems:

   - On RHEL/CentOS:

   ```
   $ sudo yum install -y gcc
   ```

   - On Ubuntu:

   ```
   $ sudo apt-get install gcc
   ```

##### 2.1.1.4.2 Updating the Kernel Headers

**To update the kernel headers:**

1. Update the kernel headers on one of the following operating systems:

   - On RHEL/CentOS:

   ```
   $ sudo yum install kernel-devel-$(uname -r) kernel-headers-$(uname -r)
   ```

   - On Ubuntu:

   ```
   $ sudo apt-get install linux-headers-$(uname -r)
   ```

2. Install **wget** one of the following operating systems:

   - On RHEL/CentOS:

   ```
   $ sudo yum install wget
   ```

   - On Ubuntu:

   ```
   $ sudo apt-get install wget
   ```

#### 2.1.1.4.3 Disabling Nouveau

You can disable Nouveau, which is the default driver.

**To disable Nouveau:**

1. Check if the Nouveau driver has been loaded:

```
$ lsmod | grep nouveau
```

If the Nouveau driver has been loaded, the command above generates output.

2. Blacklist the Nouveau drivers to disable them:

```
$ cat <<EOF | sudo tee /etc/modprobe.d/blacklist-nouveau.conf
  blacklist nouveau
  options nouveau modeset=0
  EOF
```

3. Regenerate the kernel **initramfs** directory set:

1. Modify the **initramfs** directory set:

```
$ sudo dracut --force
```

2. Reboot the server:

```
$ sudo reboot
```

#### 2.1.1.4.4 Installing the CUDA Driver

This section describes how to install the CUDA driver.

**Note:** The version of the driver installed on the customer's server must be equal or higher than the driver included in the Sqream release package. Contact a Sqream customer service representative to identify the correct version to install.

The **Installing the CUDA Driver** section describes the following:

#### 2.1.1.4.4.1 Installing the CUDA Driver from the Repository

Installing the CUDA driver from the Repository is the recommended installation method.

> **Warning:** For A100 GPU and other A series GPUs, you must install the **cuda 11.4.3 driver**. The version of the driver installed on the customer server must be equal to or higher than the one used to build the SQream package. For questions related to which driver to install, contact SQream Customer Support.

**To install the CUDA driver from the Repository:**

1. Install the CUDA dependencies for one of the following operating systems:

   - For RHEL:

     ```
     $ sudo rpm -Uvh http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.
     ↪noarch.rpm
     ```

   - For CentOS:

     ```
     $ sudo yum install epel-release
     ```

2. Install the CUDA dependencies from the **epel** repository:

   ```
   $ sudo yum install dkms libvdpau
   ```

   Installing the CUDA depedendencies from the **epel** repository is only required for installing **runfile**.

3. Download and install the required local repository:

   - **Intel - CUDA 10.1 for RHEL7**:

     ```
     $ wget http://developer.download.nvidia.com/compute/cuda/10.1/Prod/
     ↪local_installers/cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-1.0-1.
     ↪x86_64.rpm
     $ sudo yum localinstall cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-
     ↪1.0-1.x86_64.rpm
     ```

   - **Intel - 11.4.3 repository**:

     ```
     $ wget https://developer.download.nvidia.com/compute/cuda/11.4.3/local_
     ↪installers/cuda-repo-rhel7-11-4-local-11.4.3_470.82.01-1.x86_64.rpm
     $ sudo yum localinstall cuda-repo-rhel7-11-4-local-11.4.3_470.82.01-1.
     ↪x86_64.rpm
     ```

   - **IBM Power9 - CUDA 10.1 for RHEL7**:

     ```
     $ wget https://developer.download.nvidia.com/compute/cuda/10.1/Prod/
     ↪local_installers/cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-1.0-1.
     ↪ppc64le.rpm
     $ sudo yum localinstall cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-
     ↪1.0-1.ppc64le.rpm
     ```

> **Warning:** For Power9 with V100 GPUs, you must install the **CUDA 10.1** driver.

4. Install the CUDA drivers:

a. Clear the YUM cache:

```
$ sudo yum clean all
```

b. Install the most current DKMS (Dynamic Kernel Module Support) NVIDIA driver:

```
$ sudo yum -y install nvidia-driver-latest-dkms
```

5. Verify that the installation was successful:

```
$ nvidia-smi
```

**Note:** If you do not have access to internet, you can set up a local repository offline.

You can prepare the CUDA driver offline from a server connected to the CUDA repo by running the following commands as a *root* user:

6. Query all the packages installed in your system, and verify that cuda-repo has been installed:

```
$ rpm -qa |grep cuda-repo
```

7. Navigate to the correct repository:

```
$ cd /etc/yum.repos.d/
```

8. List in long format and print lines matching a pattern for the cuda file:

```
$ ls -l |grep cuda
```

The following is an example of the correct output:

```
$ cuda-10-1-local.repo
```

9. Edit the **/etc/yum.repos.d/cuda-10-1-local.repo** file:

```
$ vim /etc/yum.repos.d/cuda-10-1-local.repo
```

The following is an example of the correct output:

```
$ name=cuda-10-1-local
```

10. Clone the repository to a location where it can be copied from:

```
$ reposync -g -l -m --repoid=cuda-10-1-local --download_path=/var/cuda-repo-
↪10.1-local
```

11. Copy the repository to the installation server and create the repository:

```
$ createrepo -g comps.xml /var/cuda-repo-10.1-local
```

12. Add a repo configuration file in **/etc/yum.repos.d/** by editing the **/etc/yum.repos.d/cuda-10.1-local.repo** repository:

```
$ [cuda-10.1-local]
$ name=cuda-10.1-local
$ baseurl=file:///var/cuda-repo-10.1-local
```

(continues on next page)

(continued from previous page)

```
$ enabled=1
$ gpgcheck=1
$ gpgkey=file:///var/cuda-repo-10-1-local/7fa2af80.pub
```

13. Install the CUDA drivers by installing the most current DKMS (Dynamic Kernel Module Support) NVIDIA driver as a root user logged in shell:

```
$ sudo yum -y install nvidia-driver-latest-dkms
```

##### 2.1.1.4.4.2 Tuning Up NVIDIA Performance

This section describes how to tune up NVIDIA performance.

**Note:** The procedures in this section are relevant to Intel only.



##### 2.1.1.4.4.3 To Tune Up NVIDIA Performance when Driver Installed from the Repository

**To tune up NVIDIA performance when the driver was installed from the repository:**

1. Check the service status:

```
$ sudo systemctl status nvidia-persistenced
```

   If the service exists, it will be stopped be default.

2. Start the service:

```
$ sudo systemctl start nvidia-persistenced
```

3. Verify that no errors have occurred:

```
$ sudo systemctl status nvidia-persistenced
```

4. Enable the service to start up on boot:

```
$ sudo systemctl enable nvidia-persistenced
```

5. For **V100/A100**, add the following lines:

```
$ nvidia-persistenced
```

   **Note:** The following are mandatory for IBM:

```
$ sudo systemctl start nvidia-persistenced
$ sudo systemctl enable nvidia-persistenced
```

6. Reboot the server and run the **NVIDIA System Management Interface (NVIDIA SMI)**:

```
$ nvidia-smi
```

**Note:** Setting up the NVIDIA POWER9 CUDA driver includes additional set-up requirements. The NVIDIA POWER9 CUDA driver will not function properly if the additional set-up requirements are not followed. See POWER9 Setup for the additional set-up requirements.

#### 2.1.1.4.4.4 To Tune Up NVIDIA Performance when Driver Installed from the Runfile

**To tune up NVIDIA performance when the driver was installed from the runfile:**

1. Change the permissions on the **rc.local** file to **executable**:

```
$ sudo chmod +x /etc/rc.local
```

2. Edit the **/etc/yum.repos.d/cuda-10-1-local.repo** file:

```
$ sudo vim /etc/rc.local
```

3. Add the following lines:

   - **For V100/A100**:

```
$ nvidia-persistenced
```

   - **For IBM (mandatory)**:

```
$ sudo systemctl start nvidia-persistenced
$ sudo systemctl enable nvidia-persistenced
```

   - **For K80**:

```
$ nvidia-persistenced
$ nvidia-smi -pm 1
$ nvidia-smi -acp 0
$ nvidia-smi --auto-boost-permission=0
$ nvidia-smi --auto-boost-default=0
```

4. Reboot the server and run the **NVIDIA System Management Interface (NVIDIA SMI)**:

```
$ nvidia-smi
```

**Note:** Setting up the NVIDIA POWER9 CUDA driver includes additional set-up requirements. The NVIDIA POWER9 CUDA driver will not function properly if the additional set-up requirements are not followed. See POWER9 Setup for the additional set-up requirements.

#### 2.1.1.4.4.5 Disabling Automatic Bug Reporting Tools

**To disable automatic bug reporting tools:**

1. Run the following **abort** commands:

```
$ for i in abrt-ccpp.service abrtd.service abrt-oops.service abrt-pstoreoops.
↪service abrt-vmcore.service abrt-xorg.service ; do sudo systemctl disable $i;␣
↪sudo systemctl stop $i; done
```

The server is ready for the SQream software installation.

2. Run the following checks:

   a. Check the OS release:

   ```
   $ cat /etc/os-release
   ```

   b. Verify that a SQream user exists and has the same ID on all cluster member services:

   ```
   $ id sqream
   ```

   c. Verify that the storage is mounted:

   ```
   $ mount
   ```

   d. Verify that the driver has been installed correctly:

   ```
   $ nvidia-smi
   ```

   e. Check the maximum value of the **fs.file**:

   ```
   $ sysctl -n fs.file-max
   ```

   f. Run the following command as a SQream user:

   ```
   $ ulimit -c -u -n
   ```

   The following shows the desired output:

   ```
   $ core file size (blocks, -c) unlimited
   $ max user processes (-u) 1000000
   $ open files (-n) 1000000
   ```

### 2.1.1.5 Enabling Core Dumps

After installing the Nvidia CUDA driver, you can enable your core dumps. While SQream recommends enabling your core dumps, it is optional.

The **Enabling Core Dumps** section describes the following:

#### 2.1.1.5.1 Checking the abrtd Status

**To check the abrtd status:**

1. Check if **abrtd** is running:

```
$ sudo ps -ef |grep abrt
```

2. If **abrtd** is running, stop it:

```
$ sudo service abrtd stop
$ sudo chkconfig abrt-ccpp off
$ sudo chkconfig abrt-oops off
$ sudo chkconfig abrt-vmcore off
$ sudo chkconfig abrt-xorg off
$ sudo chkconfig abrtd off
```

#### 2.1.1.5.2 Setting the Limits

**To set the limits:**

1. Set the limits:

```
$ ulimit -c
```

2. If the output is **0**, add the following lines to the **limits.conf** file (/etc/security):

```
$ *           soft     core           unlimited
$ *           hard     core           unlimited
```

3. Log out and log in to apply the limit changes.

#### 2.1.1.5.3 Creating the Core Dumps Directory

**To set the core dumps directory:**

1. Make the **/tmp/core_dumps** directory:

```
$ mkdir /tmp/core_dumps
```

2. Set the ownership of the **/tmp/core_dumps** directory:

```
$ sudo chown sqream.sqream /tmp/core_dumps
```

3. Grant read, write, and execute permissions to all users:

```
$ sudo chmod -R 777 /tmp/core_dumps
```

**Warning:** Because the core dump file may be the size of total RAM on the server, verify that you have sufficient disk space. In the example above, the core dump is configured to the */tmp/core_dumps* directory. You must replace path according to your own environment and disk space.

#### 2.1.1.5.4 Setting the Output Directory of the /etc/sysctl.conf File

**To set the output directory of the /etc/sysctl.conf file:**

1. Edit the **/etc/sysctl.conf** file:

```
$ sudo vim /etc/sysctl.conf
```

2. Add the following to the bottom of the file:

```
$ kernel.core_uses_pid = 1
$ kernel.core_pattern = /<tmp/core_dumps>/core-%e-%s-%u-%g-%p-%t
$ fs.suid_dumpable = 2
```

3. To apply the changes without rebooting the server, run the following:

```
$ sudo sysctl -p
```

4. Check that the core output directory points to the following:

```
$ sudo cat /proc/sys/kernel/core_pattern
```

The following shows the correct generated output:

```
$ /tmp/core_dumps/core-%e-%s-%u-%g-%p-%t
```

5. Verify that the core dumping works:

```
$ select abort_server();
```

#### 2.1.1.5.5 Verifying that the Core Dumps Work

You can verify that the core dumps work only after installing and running SQream. This causes the server to crash and a new core.xxx file to be included in the folder that is written in **/etc/sysctl.conf**

**To verify that the core dumps work:**

1. Stop and restart all SQream services.

2. Connect to SQream with ClientCmd and run the following command:

```
$ select abort_server();
```

#### 2.1.1.5.6 Troubleshooting Core Dumping

This section describes the troubleshooting procedure to be followed if all parameters have been configured correctly, but the cores have not been created.

**To troubleshoot core dumping:**

1. Reboot the server.

2. Verify that you have folder permissions:

```
$ sudo chmod -R 777 /tmp/core_dumps
```

3. Verify that the limits have been set correctly:

```
$ ulimit -c
```

   If all parameters have been configured correctly, the correct output is:

```
$ core file size          (blocks, -c) unlimited
$ open files                      (-n) 1000000
```

4. If all parameters have been configured correctly, but running **ulimit -c** outputs **0**, run the following:

```
$ sudo vim /etc/profile
```

5. Search for line and tag it with the **hash** symbol:

```
$ ulimit -S -c 0 > /dev/null 2>&1
```

6. Log out and log in.

7. Run the ulimit -c command:

```
$ ulimit -c command
```

8. If the line is not found in **/etc/profile** directory, do the following:

   a. Run the following command:

```
$ sudo vim /etc/init.d/functions
```

   b. Search for the following:

```
$ ulimit -S -c ${DAEMON_COREFILE_LIMIT:-0} >/dev/null 2>&1
```

   c. If the line is found, tag it with the **hash** symbol and reboot the server.

### 2.1.2 Installing SQream Using Binary Packages

This procedure describes how to install SQream using Binary packages and must be done on all servers.

**To install SQream using Binary packages:**

1. Copy the SQream package to the **/home/sqream** directory for the current version:

```
$ tar -xf sqream-db-v<2020.2>.tar.gz
```

2. Append the version number to the name of the SQream folder. The version number in the following example is **v2020.2**:

```
$ mv sqream sqream-db-v<2020.2>
```

3. Move the new version of the SQream folder to the **/usr/local/** directory:

```
$ sudo mv sqream-db-v<2020.2> /usr/local/
```

4. Change the ownership of the folder to **sqream folder**:

```
$ sudo chown -R sqream:sqream  /usr/local/sqream-db-v<2020.2>
```

5. Navigate to the **/usr/local/** directory and create a symbolic link to SQream:

```
$ cd /usr/local
$ sudo ln -s sqream-db-v<2020.2> sqream
```

6. Verify that the symbolic link that you created points to the folder that you created:

```
$ ls -l
```

7. Verify that the symbolic link that you created points to the folder that you created:

```
$ sqream -> sqream-db-v<2020.2>
```

8. Create the SQream configuration file destination folders and set their ownership to **sqream**:

```
$ sudo mkdir /etc/sqream
$ sudo chown -R sqream:sqream /etc/sqream
```

9. Create the SQream service log destination folders and set their ownership to **sqream**:

```
$ sudo mkdir /var/log/sqream
$ sudo chown -R sqream:sqream /var/log/sqream
```

10. Navigate to the **/usr/local/** directory and copy the SQream configuration files from them:

```
$ cd /usr/local/sqream/etc/
$ cp * /etc/sqream
```

The configuration files are **service configuration files**, and the JSON files are **SQream configuration files**, for a total of four files. The number of SQream configuration files and JSON files must be identical.

**Note:** Verify that the JSON files have been configured correctly and that all required flags have been set to the correct values.

In each JSON file, the following parameters **must be updated**:

- instanceId
- machineIP
- metadataServerIp
- spoolMemoryGB
- limitQueryMemoryGB
- gpu
- port
- ssl_port

Note the following:

- The value of the **metadataServerIp** parameter must point to the IP that the metadata is running on.
- The value of the **machineIP** parameter must point to the IP of your local machine.

It would be same on server running metadataserver and different on other server nodes.

11. **Optional** - To run additional SQream services, copy the required configuration files and create additional JSON files:

```
$ cp sqream2_config.json sqream3_config.json
$ vim sqream3_config.json
```

**Note:** A unique **instanceID** must be used in each JSON file. IN the example above, the instanceID **sqream_2** is changed to **sqream_3**.

12. **Optional** - If you created additional services in **Step 11**, verify that you have also created their additional configuration files:

```
$ cp sqream2-service.conf sqream3-service.conf
$ vim sqream3-service.conf
```

13. For each SQream service configuration file, do the following:
    1. Change the **SERVICE_NAME=sqream2** value to **SERVICE_NAME=sqream3**.
    2. Change **LOGFILE=/var/log/sqream/sqream2.log** to **LOGFILE=/var/log/sqream/sqream3.log**.

**Note:** If you are running SQream on more than one server, you must configure the `serverpicker` and `metadatserver` services to start on only one of the servers. If **metadataserver** is running on the first server, the `metadataServerIP` value in the second server's /etc/sqream/sqream1_config.json file must point to the IP of the server on which the `metadataserver` service is running.

14. Set up **servicepicker**:
    1. Do the following:

    ```
    $ vim /etc/sqream/server_picker.conf
    ```

    2. Change the IP **127.0.0.1** to the IP of the server that the **metadataserver** service is running on.
    3. Change the **CLUSTER** to the value of the cluster path.

15. Set up your service files:

```
$ cd /usr/local/sqream/service/
$ cp sqream2.service sqream3.service
$ vim sqream3.service
```

16. Increment each **EnvironmentFile=/etc/sqream/sqream2-service.conf** configuration file for each SQream service file, as shown below:

```
$ EnvironmentFile=/etc/sqream/sqream<3>-service.conf
```

17. Copy and register your service files into systemd:

```
$ sudo cp metadataserver.service /usr/lib/systemd/system/
$ sudo cp serverpicker.service /usr/lib/systemd/system/
$ sudo cp sqream*.service /usr/lib/systemd/system/
```

18. Verify that your service files have been copied into systemd:

```
$ ls -l /usr/lib/systemd/system/sqream*
$ ls -l /usr/lib/systemd/system/metadataserver.service
$ ls -l /usr/lib/systemd/system/serverpicker.service
$ sudo systemctl daemon-reload
```

19. Copy the license into the **/etc/license** directory:

```
$ cp license.enc /etc/sqream/
```

If you have an HDFS environment, see *Configuring an HDFS Environment for the User sqream*.

#### 2.1.2.1 Upgrading SQream Version

Upgrading your SQream version requires stopping all running services while you manually upgrade SQream.

**To upgrade your version of SQream:**

1. Stop all actively running SQream services.

**Note:** All SQream services must remain stopped while the upgrade is in process. Ensuring that SQream services remain stopped depends on the tool being used.

For an example of stopping actively running SQream services, see *Launching SQream with Monit*.

2. Verify that SQream has stopped listening on ports **500X**, **510X**, and **310X**:

```
$ sudo netstat -nltp    #to make sure sqream stopped listening on 500X, 510X and
↪310X ports.
```

3. Replace the old version `sqream-db-v2020.2`, with the new version `sqream-db-v2021.1`:

```
$ cd /home/sqream
$ mkdir tempfolder
$ mv sqream-db-v2021.1.tar.gz tempfolder/
$ tar -xf sqream-db-v2021.1.tar.gz
$ sudo mv sqream /usr/local/sqream-db-v2021.1
$ cd /usr/local
$ sudo chown -R sqream:sqream sqream-db-v2021.1
```

4. Remove the symbolic link:

```
$ sudo rm sqream
```

5. Create a new symbolic link named "sqream" pointing to the new version:

```
$ sudo ln -s sqream-db-v2021.1 sqream
```

6. Verify that the symbolic SQream link points to the real folder:

```
$ ls -l
```

The following is an example of the correct output:

```
$ sqream -> sqream-db-v2021.1
```

7. **Optional-** (for major versions) Upgrade your version of SQream storage cluster, as shown in the following example:

```
$ cat /etc/sqream/sqream1_config.json |grep cluster
$ ./upgrade_storage <cluster path>
```

The following is an example of the correct output:

```
    get_leveldb_version path{<cluster path>}
    current storage version 23
upgrade_v24
upgrade_storage to 24
    upgrade_storage to 24 - Done
    upgrade_v25
    upgrade_storage to 25
    upgrade_storage to 25 - Done
    upgrade_v26
    upgrade_storage to 26
    upgrade_storage to 26 - Done
    validate_leveldb
    ...
upgrade_v37
    upgrade_storage to 37
    upgrade_storage to 37 - Done
    validate_leveldb
storage has been upgraded successfully to version 37
```

8. Verify that the latest version has been installed:

```
$ ./sqream sql --username sqream --password sqream --host localhost --
↪databasename master -c "SELECT SHOW_VERSION();"
```

The following is an example of the correct output:

```
v2021.1
1 row
time: 0.050603s
```

For more information, see the upgrade_storage command line program.

For more information about installing Studio on a stand-alone server, see Installing Studio on a Stand-Alone Server.

### 2.1.3 Installing and Running SQream in a Docker Container

The **Installing and Running SQream in a Docker Container** page describes how to prepare your machine's environment for installing and running SQream in a Docker container.

This page describes the following:



#### 2.1.3.1 Setting Up a Host

##### 2.1.3.1.1 Operating System Requirements

SQream was tested and verified on the following versions of Linux:

- x86 CentOS/RHEL 7.6 - 7.9
- IBM RHEL 7.6

SQream recommends installing a clean OS on the host to avoid any installation issues.

**Warning:** Docker-based installation supports only single host deployment and cannot be used on a multi-node cluster. Installing Docker on a single host you will not be able to scale it to a multi-node cluster.

##### 2.1.3.1.2 Creating a Local User

To run SQream in a Docker container you must create a local user.

**To create a local user:**

1. Add a local user:

```
$ useradd -m -U <local user name>
```

2. Set the local user's password:

```
$ passwd <local user name>
```

3. Add the local user to the `wheel` group:

```
$ usermod -aG wheel <local user name>
```

   You can remove the local user from the `wheel` group when you have completed the installation.

4. Log out and log back in as the local user.

#### 2.1.3.1.3 Setting a Local Language

After creating a local user you must set a local language.

**To set a local language:**

1. Set the local language:

```
$ sudo localectl set-locale LANG=en_US.UTF-8
```

2. Set the time stamp (time and date) of the locale:

```
$ sudo timedatectl set-timezone Asia/Jerusalem
```

You can run the `timedatectl list-timezones` command to see your timezone.

#### 2.1.3.1.4 Adding the EPEL Repository

After setting a local language you must add the EPEL repository.

**To add the EPEL repository:**

1. As a root user, upgrade the **epel-release-latest-7.noarch.rpm** repository:

   1. RedHat (RHEL 7):

```
$ sudo rpm -Uvh http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.
↪rpm
```

   2. CentOS 7

```
$ sudo yum install epel-release
```

#### 2.1.3.1.5 Installing the Required NTP Packages

After adding the EPEL repository, you must install the required NTP packages.

You can install the required NTP packages by running the following command:

```
$ sudo yum install ntp  pciutils python36 kernel-devel-$(uname -r) kernel-headers-
↪$(uname -r) gcc
```

#### 2.1.3.1.6 Installing the Recommended Tools

After installin gthe required NTP packages you must install the recommended tools.

SQream recommends installing the following recommended tools:

```
$ sudo yum install bash-completion.noarch  vim-enhanced.x86_64 vim-common.x86_64 net-
↪tools iotop htop psmisc screen xfsprogs wget yum-utils deltarpm dos2unix
```

#### 2.1.3.1.7 Updating to the Current Version of the Operating System

After installing the recommended tools you must update to the current version of the operating system.

SQream recommends updating to the current version of the operating system. This is not recommended if the nvidia driver has **not been installed.**

#### 2.1.3.1.8 Configuring the NTP Package

After updating to the current version of the operating system you must configure the NTP package.

**To configure the NTP package:**

1. Add your local servers to the NTP configuration.

2. Configure the **ntpd** service to begin running when your machine is started:

```
$ sudo systemctl enable ntpd
$ sudo systemctl start ntpd
$ sudo ntpq -p
```

#### 2.1.3.1.9 Configuring the Performance Profile

After configuring the NTP package you must configure the performance profile.

**To configure the performance profile:**

1. *Optional* - Switch the active profile:

```
$ sudo tuned-adm profile throughput-performance
```

2. Change the multi-user's default run level:

```
$ sudo systemctl set-default multi-user.target
```

#### 2.1.3.1.10 Configuring Your Security Limits

After configuring the performance profile you must configure your security limits. Configuring your security limits refers to configuring the number of open files, processes, etc.

**To configure your security limits:**

1. Run the **bash** shell as a super-user:

```
$ sudo bash
```

2. Run the following command:

```
$ echo -e "sqream soft nproc 500000\nsqream hard nproc 500000\nsqream soft nofile␣
↪500000\nsqream hard nofile 500000\nsqream soft core unlimited\nsqream hard core␣
↪unlimited" >> /etc/security/limits.conf
```

3. Run the following command:

```
$ echo -e "vm.dirty_background_ratio = 5 \n vm.dirty_ratio = 10 \n vm.swappiness␣
↪= 10 \n vm.zone_reclaim_mode = 0 \n vm.vfs_cache_pressure = 200 \n"  >> /etc/
↪sysctl.conf
```

#### 2.1.3.1.11 Disabling Automatic Bug-Reporting Tools

After configuring your security limits you must disable the following automatic bug-reporting tools:

- ccpp.service
- oops.service
- pstoreoops.service
- vmcore.service
- xorg.service

You can abort the above but-reporting tools by running the following command:

```
$ for i in abrt-ccpp.service abrtd.service abrt-oops.service abrt-pstoreoops.service␣
↪abrt-vmcore.service abrt-xorg.service ; do sudo systemctl disable $i; sudo␣
↪systemctl stop $i; done
```

#### 2.1.3.1.12 Installing the Nvidia CUDA Driver

1. Verify that the Tesla NVIDIA card has been installed and is detected by the system:

   ```
   $ lspci | grep -i nvidia
   ```

   The correct output is a list of Nvidia graphic cards. If you do not receive this output, verify that an NVIDIA GPU card has been installed.

2. Verify that the open-source upstream Nvidia driver is running:

   ```
   $ lsmod | grep nouveau
   ```

   No output should be generated.

3. If you receive any output, do the following:

   1. Disable the open-source upstream Nvidia driver:

      ```
      $ sudo bash
      $ echo "blacklist nouveau" > /etc/modprobe.d/blacklist-nouveau.conf
      $ echo "options nouveau modeset=0"  >> /etc/modprobe.d/blacklist-nouveau.conf
      $ dracut --force
      $ modprobe --showconfig | grep nouveau
      ```

   2. Reboot the server and verify that the Nouveau model has not been loaded:

      ```
      $ lsmod | grep nouveau
      ```

4. Check if the Nvidia CUDA driver has already been installed:

   ```
   $ nvidia-smi
   ```

The following is an example of the correct output:

```
nvidia-smi
Wed Oct 30 14:05:42 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.00    Driver Version: 418.87.00    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000004:04:00.0 Off |                    0 |
| N/A   32C    P0    37W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000035:03:00.0 Off |                    0 |
| N/A   33C    P0    37W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

5. Verify that the installed CUDA version shown in the output above is `10.1`.

6. Do one of the following:

   1. If CUDA version 10.1 has already been installed, skip to Docktime Runtime (Community Edition).

   2. If CUDA version 10.1 has not been installed yet, continue with Step 7 below.

7. Do one of the following:

   - Install *CUDA Driver version 10.1 for x86_64*.

   - Install *CUDA driver version 10.1 for IBM Power9*.

### 2.1.3.1.12.1 Installing the CUDA Driver Version 10.1 for x86_64

**To install the CUDA driver version 10.1 for x86_64:**

1. Make the following target platform selections:

   - **Operating system**: Linux
   - **Architecture**: x86_64
   - **Distribution**: CentOS
   - **Version**: 7
   - **Installer type**: the relevant installer type

For installer type, SQream recommends selecting **runfile (local)**. The available selections shows only the supported platforms.

2. Download the base installer for Linux CentOS 7 x86_64:

```
wget http://developer.download.nvidia.com/compute/cuda/10.1/Prod/local_installers/
↪cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-1.0-1.x86_64.rpm
```

3. Install the base installer for Linux CentOS 7 x86_64 by running the following commands:

```
$ sudo yum localinstall cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-1.0-1.x86_
↪64.rpm
$ sudo yum clean all
$ sudo yum install nvidia-driver-latest-dkms
```

**Warning:** Verify that the output indicates that driver **418.87** will be installed.

4. Follow the command line prompts.

5. Enable the Nvidia service to start at boot and start it:

```
$ sudo systemctl enable nvidia-persistenced.service && sudo systemctl start␣
↪nvidia-persistenced.service
```

7. Reboot the server.

8. Verify that the Nvidia driver has been installed and shows all available GPU's:

```
$ nvidia-smi
```

The following is the correct output:

```
nvidia-smi
Wed Oct 30 14:05:42 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.00    Driver Version: 418.87.00    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000004:04:00.0 Off |                    0 |
| N/A   32C    P0    37W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000035:03:00.0 Off |                    0 |
| N/A   33C    P0    37W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

##### 2.1.3.1.12.2 Installing the CUDA Driver Version 10.1 for IBM Power9

**To install the CUDA driver version 10.1 for IBM Power9:**

1. Download the base installer for Linux CentOS 7 PPC64le:

```
wget http://developer.download.nvidia.com/compute/cuda/10.1/Prod/local_installers/
↪cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-1.0-1.ppc64le.rpm
```

2. Install the base installer for Linux CentOS 7 x86_64 by running the following commands:

```
$ sudo rpm -i cuda-repo-rhel7-10-1-local-10.1.243-418.87.00-1.0-1.ppc64le.rpm
$ sudo yum clean all
$ sudo yum  install nvidia-driver-latest-dkms
```

**Warning:** Verify that the output indicates that driver **418.87** will be installed.

3. Copy the file to the **/etc/udev/rules.d** directory.

4. If you are using RHEL 7 version (7.6 or later), comment out, remove, or change the hot-pluggable memory rule located in file copied to the **/etc/udev/rules.d** directory by running the following command:

```
$ sudo cp /lib/udev/rules.d/40-redhat.rules /etc/udev/rules.d
$ sudo sed -i 's/SUBSYSTEM!="memory",.*GOTO="memory_hotplug_end"/SUBSYSTEM=="*",␣
↪GOTO="memory_hotplug_end"/' /etc/udev/rules.d/40-redhat.rules
```

5. Enable the **nvidia-persisted.service** file:

```
$ sudo systemctl enable nvidia-persistenced.service
```

6. Reboot your system to initialize the above modifications.

7. Verify that the Nvidia driver and the **nvidia-persistenced.service** files are running:

```
$ nvidia smi
```

The following is the correct output:

```
nvidia-smi
Wed Oct 30 14:05:42 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.00    Driver Version: 418.87.00    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000004:04:00.0 Off |                    0 |
| N/A   32C    P0    37W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000035:03:00.0 Off |                    0 |
| N/A   33C    P0    37W / 300W |      0MiB / 16130MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

8. Verify that the **nvidia-persistenced** service is running:

```
$ systemctl status nvidia-persistenced
```

The following is the correct output:

```
root@gpudb ~]systemctl status nvidia-persistenced
  nvidia-persistenced.service - NVIDIA Persistence Daemon
   Loaded: loaded (/usr/lib/systemd/system/nvidia-persistenced.service; enabled;␣
↪vendor preset: disabled)
   Active: active (running) since Tue 2019-10-15 21:43:19 KST; 11min ago
  Process: 8257 ExecStart=/usr/bin/nvidia-persistenced --verbose (code=exited,␣
↪status=0/SUCCESS)
 Main PID: 8265 (nvidia-persiste)
    Tasks: 1
   Memory: 21.0M
   CGroup: /system.slice/nvidia-persistenced.service
     └─8265 /usr/bin/nvidia-persistenced --verbose
```

#### 2.1.3.2 Installing the Docker Engine (Community Edition)

After installing the Nvidia CUDA driver you must install the Docker engine.

This section describes how to install the Docker engine using the following processors:

- *Using x86_64 processor on CentOS*
- *Using x86_64 processor on Ubuntu*
- *Using IBM Power9 (PPC64le) processor*

##### 2.1.3.2.1 Installing the Docker Engine Using an x86_64 Processor on CentOS

The x86_64 processor supports installing the **Docker Community Edition (CE)** versions 18.03 and higher.

For more information on installing the Docker Engine CE on an x86_64 processor, see Install Docker Engine on CentOS

##### 2.1.3.2.2 Installing the Docker Engine Using an x86_64 Processor on Ubuntu

The x86_64 processor supports installing the **Docker Community Edition (CE)** versions 18.03 and higher.

For more information on installing the Docker Engine CE on an x86_64 processor, see Install Docker Engine on Ubuntu

##### 2.1.3.2.3 Installing the Docker Engine on an IBM Power9 Processor

The x86_64 processor only supports installing the **Docker Community Edition (CE)** version 18.03.

**To install the Docker Engine on an IBM Power9 processor:**

You can install the Docker Engine on an IBM Power9 processor by running the following command:

```
wget http://ftp.unicamp.br/pub/ppc64el/rhel/7_1/docker-ppc64el/container-selinux-2.9-
↪4.el7.noarch.rpm
wget http://ftp.unicamp.br/pub/ppc64el/rhel/7_1/docker-ppc64el/docker-ce-18.03.1.ce-1.
↪el7.centos.ppc64le.rpm
yum install -y container-selinux-2.9-4.el7.noarch.rpm docker-ce-18.03.1.ce-1.el7.
↪centos.ppc64le.rpm
```

For more information on installing the Docker Engine CE on an IBM Power9 processor, see Install Docker Engine on Ubuntu.

#### 2.1.3.3 Docker Post-Installation

After installing the Docker engine you must configure Docker on your local machine.

**To configure Docker on your local machine:**

1. Enable Docker to start on boot:

```
$ sudo systemctl enable docker && sudo systemctl start docker
```

2. Enable managing Docker as a non-root user:

```
$ sudo usermod -aG docker $USER
```

3. Log out and log back in via SSH. This causes Docker to re-evaluate your group membership.

4. Verify that you can run the following Docker command as a non-root user (without `sudo`):

```
$ docker run hello-world
```

If you can run the above Docker command as a non-root user, the following occur:

- Docker downloads a test image and runs it in a container.
- When the container runs, it prints an informational message and exits.

For more information on installing the Docker Post-Installation, see Docker Post-Installation.

#### 2.1.3.4 Installing the Nvidia Docker2 ToolKit

After configuring Docker on your local machine you must install the Nvidia Docker2 ToolKit. The NVIDIA Docker2 Toolkit lets you build and run GPU-accelerated Docker containers. The Toolkit includes a container runtime library and related utilities for automatically configuring containers to leverage NVIDIA GPU's.

This section describes the following:

- *Installing the NVIDIA Docker2 Toolkit on an x86_64 processor*
- *Installing the NVIDIA Docker2 Toolkit on a PPC64le processor*

#### 2.1.3.4.1 Installing the NVIDIA Docker2 Toolkit on an x86_64 Processor

This section describes the following:

- *Installing the NVIDIA Docker2 Toolkit on a CentOS operating system*
- *Installing the NVIDIA Docker2 Toolkit on an Ubuntu operating system*

##### 2.1.3.4.1.1 Installing the NVIDIA Docker2 Toolkit on a CentOS Operating System

**To install the NVIDIA Docker2 Toolkit on a CentOS operating system:**

1. Install the repository for your distribution:

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.
↪repo | \
sudo tee /etc/yum.repos.d/nvidia-docker.repo
```

2. Install the `nvidia-docker2` package and reload the Docker daemon configuration:

```
$ sudo yum install nvidia-docker2
$ sudo pkill -SIGHUP dockerd
```

3. Do one of the following:
   - If you received an error when installing the `nvidia-docker2` package, skip to *Step 4*.
   - If you successfully installed the `nvidia-docker2` package, skip to *Step 5*.

4. Do the following:
   1. Run the `sudo vi /etc/yum.repos.d/nvidia-docker.repo` command if the following error is displayed when installing the `nvidia-docker2` package:

   ```
   https://nvidia.github.io/nvidia-docker/centos7/ppc64le/repodata/repomd.xml:
   [Errno -1] repomd.xml signature could not be verified for nvidia-docker
   ```

   2. Change `repo_gpgcheck=1` to `repo_gpgcheck=0`.

5. Verify that the NVIDIA-Docker run has been installed correctly:

```
$ docker run --runtime=nvidia --rm nvidia/cuda:10.1-base nvidia-smi
```

For more information on installing the NVIDIA Docker2 Toolkit on a CentOS operating system, see Installing the NVIDIA Docker2 Toolkit on a CentOS operating system

##### 2.1.3.4.1.2 Installing the NVIDIA Docker2 Toolkit on an Ubuntu Operating System

**To install the NVIDIA Docker2 Toolkit on an Ubuntu operating system:**

1. Install the repository for your distribution:

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.
↪list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update
```

2. Install the `nvidia-docker2` package and reload the Docker daemon configuration:

```
$ sudo apt-get install nvidia-docker2
$ sudo pkill -SIGHUP dockerd
```

3. Do one of the following:

   - If you received an error when installing the `nvidia-docker2` package, skip to *Step 4*.
   - If you successfully installed the `nvidia-docker2` package, skip to *Step 5*.

4. Do the following:

   1. Run the `sudo vi /etc/yum.repos.d/nvidia-docker.repo` command if the following error is displayed when installing the `nvidia-docker2` package:

   ```
   https://nvidia.github.io/nvidia-docker/centos7/ppc64le/repodata/repomd.xml:
   [Errno -1] repomd.xml signature could not be verified for nvidia-docker
   ```

   2. Change `repo_gpgcheck=1` to `repo_gpgcheck=0`.

5. Verify that the NVIDIA-Docker run has been installed correctly:

```
$ docker run --runtime=nvidia --rm nvidia/cuda:10.1-base nvidia-smi
```

For more information on installing the NVIDIA Docker2 Toolkit on a CentOS operating system, see Installing the NVIDIA Docker2 Toolkit on an Ubuntu operating system

#### 2.1.3.4.2 Installing the NVIDIA Docker2 Toolkit on a PPC64le Processor

This section describes how to install the NVIDIA Docker2 Toolkit on an IBM RHEL operating system:

**To install the NVIDIA Docker2 Toolkit on an IBM RHEL operating system:**

1. Import the repository and install the `libnvidia-container` and the `nvidia-container-runtime` containers.

```
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
$ curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.
↪repo | \
  sudo tee /etc/yum.repos.d/nvidia-docker.repo
$ sudo yum install -y libnvidia-container*
```

2. Do one of the following:

   - If you received an error when installing the containers, skip to *Step 3*.
   - If you successfully installed the containers, skip to *Step 4*.

3. Do the following:

   1. Run the `sudo vi /etc/yum.repos.d/nvidia-docker.repo` command if the following error is displayed when installing the containers:

   ```
   https://nvidia.github.io/nvidia-docker/centos7/ppc64le/repodata/repomd.xml:
   [Errno -1] repomd.xml signature could not be verified for nvidia-docker
   ```

   2. Change `repo_gpgcheck=1` to `repo_gpgcheck=0`.

3. Install the `libnvidia-container` container.

```
$ sudo yum install -y libnvidia-container*
```

4. Install the `nvidia-container-runtime` container:

```
$ sudo yum install -y nvidia-container-runtime*
```

5. Add `nvidia runtime` to the Docker daemon:

```
$ sudo mkdir -p /etc/systemd/system/docker.service.d/
$ sudo vi /etc/systemd/system/docker.service.d/override.conf

$ [Service]
$ ExecStart=
$ ExecStart=/usr/bin/dockerd
```

6. Restart Docker:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

7. Verify that the NVIDIA-Docker run has been installed correctly:

```
$ docker run --runtime=nvidia --rm nvidia/cuda-ppc64le nvidia-smi
```

#### 2.1.3.4.3 Accessing the Hadoop and Kubernetes Configuration Files

The information this section is optional and is only relevant for Hadoop users. If you require Hadoop and Kubernetes (Krb5) connectivity, contact your IT department for access to the following configuration files:

- Hadoop configuration files:
  - core-site.xml
  - hdfs-site.xml
- Kubernetes files:
  - Configuration file - krb.conf
  - Kubernetes Hadoop client certificate - hdfs.keytab

Once you have the above files, you must copy them into the correct folders in your working directory.

For more information about the correct directory to copy the above files into, see the *Installing the SQream Software* section below.

For related information, see the following sections:

- *Configuring the Hadoop and Kubernetes Configuration Files*
- *Setting the Hadoop and Kubernetes Configuration Parameters*

#### 2.1.3.5 Installing the SQream Software

##### 2.1.3.5.1 Preparing Your Local Environment

After installing the Nvidia Docker2 toolKit you must prepare your local environment.

**Note:** You must install the SQream software under a *sqream* and not a *root* user.

The Linux user preparing the local environment must have **read/write** access to the following directories for the SQream software to correctly read and write the required resources:

- **Log directory** - default: /var/log/sqream/
- **Configuration directory** - default: /etc/sqream/
- **Cluster directory** - the location where SQream writes its DB system, such as */mnt/sqreamdb*
- **Ingest directory** - the location where the required data is loaded, such as */mnt/data_source/*

##### 2.1.3.5.2 Deploying the SQream Software

After preparing your local environment you must deploy the SQream software. Deploying the SQream software requires you to access and extract the required files and to place them in the correct directory.

**To deploy the SQream software:**

1. Contact the SQream Support team for access to the **sqream_installer-nnn-DBnnn-COnnn-EDnnn-<arch>.tar.gz** file.

The **sqream_installer-nnn-DBnnn-COnnn-EDnnn-<arch>.tar.gz** file includes the following parameter values:

- **sqream_installer-nnn** - sqream installer version
- **DBnnn** - SQreamDB version
- **COnnn** - SQream console version
- **EDnnn** - SQream editor version
- **arch** - server arch (applicable to X86.64 and ppc64le)

2. Extract the tarball file:

```
$ tar -xvf sqream_installer-1.1.5-DB2019.2.1-CO1.5.4-ED3.0.0-x86_64.tar.gz
```

   When the tarball file has been extracted, a new folder will be created. The new folder is automatically given the name of the tarball file:

```
drwxrwxr-x 9 sqream sqream 4096 Aug 11 11:51 sqream_istaller-1.1.5-DB2019.2.1-CO1.
↪5.4-ED3.0.0-x86_64/
-rw-rw-r-- 1 sqream sqream 3130398797 Aug 11 11:20 sqream_installer-1.1.5-DB2019.
↪2.1-CO1.5.4-ED3.0.0-x86_64.tar.gz
```

3. Change the directory to the new folder that you created in the previous step.
4. Verify that the folder you just created contains all of the required files.

```
$ ls -la
```

   The following is an example of the files included in the new folder:

```
drwxrwxr-x. 10 sqream sqream   198 Jun  3 17:57 .
drwx------. 25 sqream sqream  4096 Jun  7 18:11 ..
drwxrwxr-x.  2 sqream sqream   226 Jun  7 18:09 .docker
drwxrwxr-x.  2 sqream sqream    64 Jun  3 12:55 .hadoop
drwxrwxr-x.  2 sqream sqream  4096 May 31 14:18 .install
drwxrwxr-x.  2 sqream sqream    39 Jun  3 12:53 .krb5
drwxrwxr-x.  2 sqream sqream    22 May 31 14:18 license
drwxrwxr-x.  2 sqream sqream    82 May 31 14:18 .sqream
-rwxrwxr-x.  1 sqream sqream  1712 May 31 14:18 sqream-console
-rwxrwxr-x.  1 sqream sqream  4608 May 31 14:18 sqream-install
```

For information relevant to Hadoop users, see the following sections:

- *Accessing the Hadoop and Kubernetes Configuration Files*.
- *Configuring the Hadoop and Kubernetes Configuration Files*.
- *Setting the Hadoop and Kubernetes Configuration Parameters*.

#### 2.1.3.5.3 Configuring the Hadoop and Kubernetes Configuration Files

The information in this section is optional and is only relevant for Hadoop users. If you require Hadoop and Kubernetes (Krb5) connectivity, you must copy the Hadoop and Kubernetes files into the correct folders in your working directory as shown below:

- .hadoop/core-site.xml
- .hadoop/hdfs-site.xml
- .krb5/krb5.conf
- .krb5/hdfs.keytab

For related information, see the following sections:

- *Accessing the Hadoop and Kubernetes Configuration Files*.
- *Setting the Hadoop and Kubernetes Configuration Parameters*.

#### 2.1.3.5.4 Configuring the SQream Software

After deploying the SQream software, and optionally configuring the Hadoop and Kubernetes configuration files, you must configure the SQream software.

Configuring the SQream software requires you to do the following:

- Configure your local environment
- Understand the `sqream-install` flags
- Install your SQream license
- Validate your SQream icense
- Change your data ingest folder

#### 2.1.3.5.4.1 Configuring Your Local Environment

Once you've downloaded the SQream software, you can begin configuring your local environment. The following commands must be run (as **sudo**) from the same directory that you located your packages.

For example, you may have saved your packages in **/home/sqream/sqream-console-package/**.

The following table shows the flags that you can use to configure your local directory:

| Flag | Function | Note |
|---|---|---|
| **-i** | Loads all software from the hidden folder **.docker**. | Mandatory |
| **-k** | Loads all license packages from the **/license** directory. | Mandatory |
| **-f** | Overwrites existing folders. **Note** Using `-f` overwrites **all files** located in mounted directories. | Mandatory |
| **-c** | Defines the origin path for writing/reading SQream configuration files. The default location is `/etc/sqream/`. | If you are installing the Docker version on a server that already works with SQream, do not use the default path. |
| **-v** | The SQream cluster location. If a cluster does not exist yet, `-v` creates one. If a cluster already exists, `-v` mounts it. | Mandatory |
| **-l** | SQream system startup logs location, including startup logs and docker logs. The default location is `/var/log/sqream/`. | |
| **-d** | The directory containing customer data to be imported and/or copied to SQream. | |
| **-s** | Shows system settings. | |
| **-r** | Resets the system configuration. This value is run without any other variables. | Mandatory |
| **-h** | Help. Shows the available flags. | Mandatory |
| **-K** | Runs license validation | |
| **-e** | Used for inserting your RKrb5 server DNS name. For more information on setting your Kerberos configuration parameters, see *Setting the Hadoop and Kubernetes Configuration Parameters*. | |
| **-p** | Used for inserting your Kerberos user name. For more information on setting your Kerberos configuration parameters, see *Setting the Hadoop and Kubernetes Configuration Parameters*. | |

#### 2.1.3.5.4.2 Installing Your License

Once you've configured your local environment, you must install your license by copying it into the SQream installation package folder located in the **./license** folder:

```
$ sudo ./sqream-install -k
```

You do not need to extract this folder after uploading into the **./license**.

#### 2.1.3.5.4.3 Validating Your License

You can copy your license package into the SQream console folder located in the **/license** folder by running the following command:

```
$ sudo ./sqream-install -K
```

The following mandatory flags must be used in the first run:

```
$ sudo ./sqream-install -i -k -v <volume path>
```

The following is an example of the correct command syntax:

```
$ sudo ./sqream-install -i -k -c /etc/sqream -v /home/sqream/sqreamdb -l /var/log/
↪sqream -d /home/sqream/data_ingest
```

### 2.1.3.5.5 Setting the Hadoop and Kubernetes Connectivity Parameters

The information in this section is optional, and is only relevant for Hadoop users. If you require Hadoop and Kubernetes (Krb5) connectivity, you must set their connectivity parameters.

The following is the correct syntax when setting the Hadoop and Kubernetes connectivity parameters:

```
$ sudo ./sqream-install -p <Kerberos user name> -e  <Kerberos server DNS name>:
↪<Kerberos server IP>
```

The following is an example of setting the Hadoop and Kubernetes connectivity parameters:

```
$ sudo ./sqream-install -p <nn1@SQ.COM> -e  kdc.sq.com:<192.168.1.111>
```

For related information, see the following sections:

- *Accessing the Hadoop and Kubernetes Configuration Files*.
- *Configuring the Hadoop and Kubernetes Configuration Files*.

#### 2.1.3.5.5.1 Modifying Your Data Ingest Folder

Once you've validated your license, you can modify your data ingest folder after the first run by running the following command:

```
$ sudo ./sqream-install -d /home/sqream/data_in
```

#### 2.1.3.5.5.2 Configuring Your Network for Docker

Once you've modified your data ingest folder (if needed), you must validate that the server network and Docker network that you are setting up do not overlap.

**To configure your network for Docker:**

1. To verify that your server network and Docker network do not overlap, run the following command:

```
$ ifconfig | grep 172.
```

2. Do one of the following:

- If running the above command output no results, continue the installation process.
- If running the above command output results, run the following command:

```
$ ifconfig | grep 192.168.
```

#### 2.1.3.5.5.3 Checking and Verifying Your System Settings

Once you've configured your network for Docker, you can check and verify your system settings.

Running the following command shows you all the variables used by your SQream system:

```
$ ./sqream-install -s
```

The following is an example of the correct output:

```
SQREAM_CONSOLE_TAG=1.5.4
SQREAM_TAG=2019.2.1
SQREAM_EDITOR_TAG=3.0.0
license_worker_0=f0:cc:
license_worker_1=26:91:
license_worker_2=20:26:
license_worker_3=00:36:
SQREAM_VOLUME=/media/sqreamdb
SQREAM_DATA_INGEST=/media/sqreamdb/data_in
SQREAM_CONFIG_DIR=/etc/sqream/
LICENSE_VALID=true
SQREAM_LOG_DIR=/var/log/sqream/
SQREAM_USER=sqream
SQREAM_HOME=/home/sqream
SQREAM_ENV_PATH=/home/sqream/.sqream/env_file
PROCESSOR=x86_64
METADATA_PORT=3105
PICKER_PORT=3108
NUM_OF_GPUS=2
CUDA_VERSION=10.1
NVIDIA_SMI_PATH=/usr/bin/nvidia-smi
DOCKER_PATH=/usr/bin/docker
NVIDIA_DRIVER=418
SQREAM_MODE=single_host
```

### 2.1.3.6 Using the SQream Console

After configuring the SQream software and veriying your system settings you can begin using the SQream console.

#### 2.1.3.6.1 SQream Console - Basic Commands

The SQream console offers the following basic commands:

- *Starting your SQream console*
- *Starting Metadata and Picker*
- *Starting the running services*
- *Listing the running services*
- *Stopping the running services*
- *Using the SQream editor*
- *Using the SQream Client*

##### 2.1.3.6.1.1 Starting Your SQream Console

You can start your SQream console by running the following command:

```
$ ./sqream-console
```

##### 2.1.3.6.1.2 Starting the SQream Master

**To listen to metadata and picker:**

1. Start the metadata server (default port 3105) and picker (default port 3108) by running the following command:

   ```
   $ sqream master --start
   ```

   The following is the correct output:

   ```
   sqream-console> sqream master --start
   starting master server in single_host mode ...
   sqream_single_host_master is up and listening on ports: 3105,3108
   ```

2. *Optional* - Change the metadata and server picker ports by adding `-p <port number>` and `-m <port number>`:

   ```
   $ sqream-console>sqream master --start -p 4105 -m 43108
   $ starting master server in single_host mode ...
   $ sqream_single_host_master is up and listening on ports: 4105,4108
   ```

##### 2.1.3.6.1.3 Starting SQream Workers

When starting SQream workers, setting the `<number of workers>` value sets how many workers to start. Leaving the `<number of workers>` value unspecified runs all of the available resources.

```
$ sqream worker --start <number of workers>

The following is an example of expected output when setting the ``<number of workers>
↪`` value to ``2``:
```

(continues on next page)

(continued from previous page)

```
.. code-block::

   sqream-console>sqream worker --start 2
   started sqream_single_host_worker_0 on port 5000, allocated gpu: 0
   started sqream_single_host_worker_1 on port 5001, allocated gpu: 1
```

#### 2.1.3.6.1.4 Listing the Running Services

You can list running SQream services to look for container names and ID's by running the following command:

```
$ sqream master --list
```

The following is an example of the expected output:

```
sqream-console>sqream master --list
container name: sqream_single_host_worker_0, container id: c919e8fb78c8
container name: sqream_single_host_master, container id: ea7eef80e038--
```

#### 2.1.3.6.1.5 Stopping the Running Services

You can stop running services either for a single SQream worker, or all SQream services for both master and worker.

The following is the command for stopping a running service for a single SQream worker:

```
$ sqream worker --stop <full worker name>
```

The following is an example of expected output when stopping a running service for a single SQream worker:

```
sqream worker stop <full worker name>
stopped container sqream_single_host_worker_0, id: 892a8f1a58c5
```

You can stop all running SQream services (both master and worker) by running the following command:

```
$ sqream-console>sqream master --stop --all
```

The following is an example of expected output when stopping all running services:

```
sqream-console>sqream master --stop --all
stopped container sqream_single_host_worker_0, id: 892a8f1a58c5
stopped container sqream_single_host_master, id: 55cb7e38eb22
```

#### 2.1.3.6.1.6 Using SQream Studio

SQream Studio is an SQL statement editor.

**To start SQream Studio:**

1. Run the following command:

   ```
   $ sqream studio --start
   ```

The following is an example of the expected output:

```
SQream Acceleration Studio is available at http://192.168.1.62:8080
```

2. Click the `http://192.168.1.62:8080` link shown in the CLI.

**To stop SQream Studio:**

You can stop your SQream Studio by running the following command:

```
$ sqream studio --stop
```

The following is an example of the expected output:

```
sqream_admin    stopped
```

#### 2.1.3.6.1.7 Using the SQream Client

You can use the embedded SQream Client on the following nodes:

- Master node
- Worker node

When using the SQream Client on the Master node, the following default settings are used:

- **Default port**: 3108. You can change the default port using the `-p` variable.
- **Default database**: master. You can change the default database using the `-d` variable.

The following is an example:

```
$ sqream client --master -u sqream -w sqream
```

When using the SQream Client on a Worker node (or nodes), you should use the `-p` variable for Worker ports. The default database is `master`, but you can use the `-d` variable to change databases.

The following is an example:

```
$ sqream client --worker -p 5000 -u sqream -w sqream
```

### 2.1.3.6.2 Moving from Docker Installation to Standard On-Premises Installation

Because Docker creates all files and directories on the host at the **root** level, you must grant ownership of the SQream storage folder to the working directory user.

### 2.1.3.6.3 SQream Console - Advanced Commands

The SQream console offers the following advanced commands:

- *Controlling the spool size*
- *Splitting a GPU*
- *Splitting a GPU and setting the spool size*
- *Using a custom configuration file*
- *Clustering your Docker environment*

##### 2.1.3.6.3.1 Controlling the Spool Size

From the console you can define a spool size value.

The following example shows the spool size being set to `50`:

```
$ sqream-console>sqream worker --start 2 -m 50
```

If you don't define the SQream spool size, the SQream console automatically distributes the available RAM between all running workers.

##### 2.1.3.6.3.2 Splitting a GPU

You can start more than one sqreamd on a single GPU by splitting it.

The following example shows the GPU being split into **two** sqreamd's on the GPU in **slot 0**:

```
$ sqream-console>sqream worker --start 2 -g 0
```

##### 2.1.3.6.3.3 Splitting GPU and Setting the Spool Size

You can simultaneously split a GPU and set the spool size by appending the `-m` flag:

```
$ sqream-console>sqream worker --start 2 -g 0 -m 50
```

**Note:** The console does not validate whether the user-defined spool size is available. Before setting the spool size, verify that the requested resources are available.

##### 2.1.3.6.3.4 Using a Custom Configuration File

SQream lets you use your own external custom configuration json files. You must place these json files in the path mounted in the installation. SQream recommends placing the json file in the Configuration folder.

The SQream console does not validate the integrity of your external configuration files.

When using your custom configuration file, you can use the `-j` flag to define the full path to the Configuration file, as in the example below:

```
$ sqream-console>sqream worker --start 1 -j /etc/sqream/configfile.json
```

**Note:** To start more than one sqream daemon, you must provide files for each daemon, as in the example below:

```
$ sqream worker --start 2 -j /etc/sqream/configfile.json /etc/sqream/configfile2.json
```

**Note:** To split a specific GPU, you must also list the GPU flag, as in the example below:

```
$ sqream worker --start 2 -g 0 -j /etc/sqream/configfile.json /etc/sqream/configfile2.
↪json
```

#### 2.1.3.6.3.5 Clustering Your Docker Environment

SQream lets you connect to a remote Master node to start Docker in Distributed mode. If you have already connected to a Slave node server in Distributed mode, the **sqream Master** and **Client** commands are only available on the Master node.

```
$ --master-host
$ sqream-console>sqream worker --start 1 --master-host 192.168.0.1020
```

### 2.1.3.6.4 Checking the Status of SQream Services

SQream lets you check the status of SQream services from the following locations:

- *From the Sqream console*
- *From outside the Sqream console*

#### 2.1.3.6.4.1 Checking the Status of SQream Services from the SQream Console

From the SQream console, you can check the status of SQream services by running the following command:

```
$ sqream-console>sqream master --list
```

The following is an example of the expected output:

```
$ sqream-console>sqream master --list
$ checking 3 sqream services:
$ sqream_single_host_worker_1 up, listens on port: 5001 allocated gpu: 1
$ sqream_single_host_worker_0 up, listens on port: 5000 allocated gpu: 1
$ sqream_single_host_master up listens on ports: 3105,3108
```

#### 2.1.3.6.4.2 Checking the Status of SQream Services from Outside the SQream Console

From outside the Sqream Console, you can check the status of SQream services by running the following commands:

```
$ sqream-status
$ NAMES STATUS PORTS
$ sqream_single_host_worker_1 Up 3 minutes 0.0.0.0:5001->5001/tcp
$ sqream_single_host_worker_0 Up 3 minutes 0.0.0.0:5000->5000/tcp
$ sqream_single_host_master Up 3 minutes 0.0.0.0:3105->3105/tcp, 0.0.0.0:3108->3108/
↪tcp
$ sqream_editor_3.0.0 Up 3 hours (healthy) 0.0.0.0:3000->3000/tcp
```

#### 2.1.3.6.5 Upgrading Your SQream System

This section describes how to upgrade your SQream system.

**To upgrade your SQream system:**

1. Contact the SQream Support team for access to the new SQream package tarball file.

2. Set a maintenance window to enable stopping the system while upgrading it.

3. Extract the following tarball file received from the SQream Support team, under it with the same user and in the same folder that you used while Downloading the SQream Software.

   ```
   $ tar -xvf sqream_installer-2.0.5-DB2019.2.1-CO1.6.3-ED3.0.0-x86_64/
   ```

4. Navigate to the new folder created as a result of extracting the tarball file:

   ```
   $ cd sqream_installer-2.0.5-DB2019.2.1-CO1.6.3-ED3.0.0-x86_64/
   ```

5. Initiate the upgrade process:

   ```
   $ ./sqream-install -i
   ```

   Initiating the upgrade process checks if any SQream services are running. If any services are running, you will be prompted to stop them.

6. Do one of the following:

   - Select **Yes** to stop all running SQream workers (Master and Editor) and continue the upgrade process.
   - Select **No** to stop the upgrade process.

   SQream periodically upgrades the metadata structure. If an upgrade version includes a change to the metadata structure, you will be prompted with an approval request message. Your approval is required to finish the upgrade process.

   Because SQream supports only certain metadata versions, all SQream services must be upgraded at the same time.

7. When the upgrade is complete, load the SQream console and restart your services.

   For assistance, contact SQream Support.

### 2.1.4 Installing SQream with Kubernetes

**Kubernetes**, also known as **k8s**, is a portable open source platform that automates Linux container operations. Kubernetes supports outsourcing data centers to public cloud service providers or can be scaled for web hosting. SQream uses Kubernetes as an orchestration and recovery solution.

The **Installing SQream with Kubernetes** guide describes the following:

#### 2.1.4.1 Preparing the SQream Environment to Launch SQream Using Kubernetes

The **Preparing the SQream environment to Launch SQream Using Kubernetes** section describes the following:



##### 2.1.4.1.1 Overview

A minimum of three servers is required for preparing the SQream environment using Kubernetes.

Kubernetes uses clusters, which are sets of nodes running containterized applications. A cluster consists of at least two GPU nodes and one additional server without GPU to act as the quorum manager.

Each server must have the following IP addresses:

- An IP address located in the management network.
- An additional IP address from the same subnet to function as a floating IP.

All servers must be mounted in the same shared storage folder.

The following list shows the server host name format requirements:

- A maximum of 253 characters.
- Only lowercase alphanumeric characters, such as `-` or `.`.
- Starts and ends with alphanumeric characters.

Go back to *Preparing the SQream Environment to Launch SQream Using Kubernetes*

##### 2.1.4.1.2 Operating System Requirements

The required operating system is a version of x86 CentOS/RHEL between 7.6 and 7.9. Regarding PPC64le, the required version is RHEL 7.6.

Go back to *Preparing the SQream Environment to Launch SQream Using Kubernetes*

#### 2.1.4.1.3 Compute Server Specifications

Installing SQream with Kubernetes includes the following compute server specifications:

- **CPU:** 4 cores
- **RAM:** 16GB
- **HD:** 500GB

Go back to *Preparing the SQream Environment to Launch SQream Using Kubernetes*

### 2.1.4.2 Setting Up Your Hosts

SQream requires you to set up your hosts. Setting up your hosts requires the following:



#### 2.1.4.2.1 Configuring the Hosts File

**To configure the /etc/hosts file:**

1. Edit the **/etc/hosts** file:

```
$ sudo vim /etc/hosts
```

2. Call your local host:

```
$ 127.0.0.1         localhost
$ <server ip>       <server_name>
```

#### 2.1.4.2.2 Installing the Required Packages

The first step in setting up your hosts is to install the required packages.

**To install the required packages:**

1. Run the following command based on your operating system:

   - RHEL:

```
$ sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-
↪latest-7.noarch.rpm
```

   - CentOS:

```
$ sudo yum install epel-release
$ sudo yum install pciutils openssl-devel python36 python36-pip kernel-
↪devel-$(uname -r) kernel-headers-$(uname -r) gcc jq net-tools ntp
```

2. Verify that that the required packages were successfully installed. The following is the correct output:

```
ntpq --version
jq --version
python3 --version
pip3 --version
rpm -qa |grep kernel-devel-$(uname -r)
rpm -qa |grep kernel-headers-$(uname -r)
gcc --version
```

3. Enable the **ntpd (Network Time Protocol daemon)** program on all servers:

```
$ sudo systemctl start ntpd
$ sudo systemctl enable ntpd
$ sudo systemctl status ntpd
$ sudo ntpq -p
```

Go back to *Setting Up Your Hosts*

#### 2.1.4.2.3 Disabling the Linux UI

After installing the required packages, you must disable the Linux UI if it has been installed.

You can disable Linux by running the following command:

```
$ sudo systemctl set-default multi-user.target
```

Go back to *Setting Up Your Hosts*

#### 2.1.4.2.4 Disabling SELinux

After disabling the Linux UI you must disable SELinux.

**To disable SELinux:**

1. Run the following command:

```
$ sed -i -e s/enforcing/disabled/g /etc/selinux/config
$ sudo reboot
```

2. Reboot the system as a root user:

```
$ sudo reboot
```

Go back to *Setting Up Your Hosts*

#### 2.1.4.2.5 Disabling Your Firewall

After disabling SELinux, you must disable your firewall by running the following commands:

```
$ sudo systemctl stop firewalld
$ sudo systemctl disable firewalld
```

Go back to *Setting Up Your Hosts*

#### 2.1.4.2.6 Checking the CUDA Version

After completing all of the steps above, you must check the CUDA version.

**To check the CUDA version:**

1. Check the CUDA version:

```
$ nvidia-smi
```

The following is an example of the correct output:

```
$ +-----------------------------------------------------------------------------+
$ | NVIDIA-SMI 418.87.00    Driver Version: 418.87.00    CUDA Version: 10.1     |
$ |-------------------------------+----------------------+----------------------+
$ | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
$ | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
$ |===============================+======================+======================|
$ |   0  GeForce GTX 105...  Off  | 00000000:01:00.0 Off |                  N/A |
$ | 32%   38C    P0    N/A /  75W |      0MiB /  4039MiB |      0%      Default |
$ +-------------------------------+----------------------+----------------------+
$
$ +-----------------------------------------------------------------------------+
$ | Processes:                                                       GPU Memory |
$ |  GPU       PID   Type   Process name                             Usage      |
$ |=============================================================================|
$ |  No running processes found                                                 |
$ +-----------------------------------------------------------------------------+
```

In the above output, the CUDA version is **10.1**.

If the above output is not generated, CUDA has not been installed. To install CUDA, see Installing the CUDA driver.

Go back to *Setting Up Your Hosts*

### 2.1.4.3 Installing Your Kubernetes Cluster

After setting up your hosts, you must install your Kubernetes cluster. The Kubernetes and SQream software must be installed from the management host, and can be installed on any server in the cluster.

Installing your Kubernetes cluster requires the following:

- *Generating and Sharing SSH Keypairs Across All Existing Nodes*
- *Installing and Deploying a Kubernetes Cluster Using Kubespray*
- *Adjusting Kubespray Deployment Values*

#### 2.1.4.3.1 Generating and Sharing SSH Keypairs Across All Existing Nodes

You can generate and share SSH keypairs across all existing nodes. Sharing SSH keypairs across all nodes enables passwordless access from the management server to all nodes in the cluster. All nodes in the cluster require passwordless access.

**Note:** You must generate and share an SSH keypair across all nodes even if you are installing the Kubernetes cluster on a single host.

**To generate and share an SSH keypair:**

1. Switch to root user access:

```
$ sudo su -
```

2. Generate an RSA key pair:

```
$ ssh-keygen
```

The following is an example of the correct output:

```
$ ssh-keygen
$ Generating public/private rsa key pair.
$ Enter file in which to save the key (/root/.ssh/id_rsa):
$ Created directory '/root/.ssh'.
$ Enter passphrase (empty for no passphrase):
$ Enter same passphrase again:
$ Your identification has been saved in /root/.ssh/id_rsa.
$ Your public key has been saved in /root/.ssh/id_rsa.pub.
$ The key fingerprint is:
$ SHA256:xxxxxxxxxxxxxxdsdsdffggtt66gfgfg root@localhost.localdomain
$ The key's randomart image is:
$ +---[RSA 2048]----+
$ |             =*.  |
```

(continues on next page)

(continued from previous page)

```
$ |             .o    |
$ |             ..o o|
$ |      .      .oo +.|
$ |       = S =...o o|
$ |        B + *..o+.|
$ |       o * *..o .+|
$ |        o * oo.E.o|
$ |       . ..+..B.+o|
$ +----[SHA256]-----+
```

The generated file is `/root/.ssh/id_rsa.pub`.

3. Copy the public key to all servers in the cluster, including the one that you are running on.

```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub root@remote-host
```

4. Replace the `remote host` with your host IP address.

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.2 Installing and Deploying a Kubernetes Cluster Using Kubespray

SQream uses the Kubespray software package to install and deploy Kubernetes clusters.

**To install and deploy a Kubernetes cluster using Kubespray:**

1. Clone Kubernetes:

   1. Clone the **kubespray.git** repository:

   ```
   $ git clone https://github.com/kubernetes-incubator/kubespray.git
   ```

   2. Nagivate to the **kubespray** directory:

   ```
   $ cd kubespray
   ```

   3. Install the **requirements.txt** configuration file:

   ```
   $ pip3 install -r requirements.txt
   ```

2. Create your SQream inventory directory:

   1. Run the following command:

   ```
   $ cp -rp inventory/sample inventory/sqream
   ```

   2. Replace the **<cluster node IP>** with the defined cluster node IP address(es).

   ```
   $ declare -a IPS=(<host>, <cluster node IP address>)
   ```

   For example, the following replaces `192.168.0.93` with `192.168.0.92`:

   ```
   $ declare -a IPS=(host-93,192.168.0.93 host-92,192.168.0.92)
   ```

**Note the following:**

- Running a declare requires defining a pair (host name and cluster node IP address), as shown in the above example.

- You can define more than one pair.

3. When the reboot is complete, switch back to the root user:

```
$ sudo su -
```

4. Navigate to **root/kubespray**:

```
$ cd /root/kubespray
```

5. Copy `inventory/sample` as `inventory/sqream`:

```
$ cp -rfp inventory/sample inventory/sqream
```

6. Update the Ansible inventory file with the inventory builder:

```
$ declare -a IPS=(<hostname1>,<IP1> <hostname2>,<IP2> <hostname3>,<IP3>)
```

7. In the **kubespray hosts.yml** file, set the node IP's:

```
$ CONFIG_FILE=inventory/sqream/hosts.yml python3 contrib/inventory_builder/
↪inventory.py ${IPS[@]}
```

If you do not set a specific hostname in declare, the server hostnames will change to `node1`, `node2`, etc. To maintain specific hostnames, run declare as in the following example:

```
$ declare -a IPS=(eks-rhl-1,192.168.5.81 eks-rhl-2,192.168.5.82 eks-rhl-3,192.168.
↪5.83)
```

Note that the declare must contain pairs (hostname,ip).

8. Verify that the following have been done:
   - That the **hosts.yml** file is configured correctly.
   - That all children are included with their relevant nodes.

You can save your current server hostname by replacing <nodeX> with your server hostname.

9. Generate the content output of the **hosts.yml** file. Make sure to include the file's directory:

```
$ cat  inventory/sqream/hosts.yml
```

The hostname can be lowercase and contain `-` or `.` only, and must be aligned with the server's hostname.

The following is an example of the correct output. Each host and IP address that you provided in Step 2 should be displayed once:

```
$ all:
$   hosts:
$     node1:
$       ansible_host: 192.168.5.81
$       ip: 192.168.5.81
$       access_ip: 192.168.5.81
$     node2:
$       ansible_host: 192.168.5.82
$       ip: 192.168.5.82
$       access_ip: 192.168.5.82
$     node3:
$       ansible_host: 192.168.5.83
```

(continues on next page)

(continued from previous page)

```
$       ip: 192.168.5.83
$       access_ip: 192.168.5.83
$   children:
$     kube-master:
$       hosts:
$         node1:
$         node2:
$         node3:
$     kube-node:
$       hosts:
$         node1:
$         node2:
$         node3:
$     etcd:
$       hosts:
$         node1:
$         node2:
$         node3:
$     k8s-cluster:
$       children:
$         kube-master:
$         kube-node:
$     calico-rr:
$       hosts: {}
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.3 Adjusting Kubespray Deployment Values

After downloading and configuring Kubespray, you can adjust your Kubespray deployment values. A script is used to modify how the Kubernetes cluster is deployed, and you must set the cluster name variable before running this script.

**Note:** The script must be run from the **kubespray** folder.

**To adjust Kubespray deployment values:**

1. Add the following export to the local user's **~/.bashrc** file by replacing the <VIP IP> with the user's Virtual IP address:

```
$ export VIP_IP=<VIP IP>
```

2. Logout, log back in, and verify the following:

```
$ echo $VIP_IP
```

3. Make the following replacements to the **kubespray.settings.sh** file:

```
$ cat <<EOF > kubespray_settings.sh
$ sed -i "/cluster_name: cluster.local/c   \cluster_name: cluster.local.$cluster_
↪name" inventory/sqream/group_vars/k8s-cluster/k8s-cluster.yml
$ sed -i "/dashboard_enabled/c   \dashboard_enabled\: "false"" inventory/sqream/
↪group_vars/k8s-cluster/addons.yml
$ sed -i "/kube_version/c   \kube_version\: "v1.18.3"" inventory/sqream/group_
↪vars/k8s-cluster/k8s-cluster.yml
```

(continues on next page)

(continued from previous page)

```
$ sed -i "/metrics_server_enabled/c   \metrics_server_enabled\: "true"" inventory/
↪sample/group_vars/k8s-cluster/addons.yml
$ echo 'kube_apiserver_node_port_range: "3000-6000"' >> inventory/sqream/group_
↪vars/k8s-cluster/k8s-cluster.yml
$ echo 'kube_controller_node_monitor_grace_period: 20s' >> inventory/sqream/group_
↪vars/k8s-cluster/k8s-cluster.yml
$ echo 'kube_controller_node_monitor_period: 2s' >> inventory/sqream/group_vars/
↪k8s-cluster/k8s-cluster.yml
$ echo 'kube_controller_pod_eviction_timeout: 30s' >> inventory/sqream/group_vars/
↪k8s-cluster/k8s-cluster.yml
$ echo 'kubelet_status_update_frequency: 4s' >> inventory/sqream/group_vars/k8s-
↪cluster/k8s-cluster.yml
$ echo 'ansible ALL=(ALL) NOPASSWD: ALL' >> /etc/sudoers
$ EOF
```

**Note:** In most cases, the Docker data resides on the system disk. Because Docker requires a high volume of data (images, containers, volumes, etc.), you can change the default Docker data location to prevent the system disk from running out of space.

4. *Optional* - Change the default Docker data location:

```
$ sed -i "/docker_daemon_graph/c   \docker_daemon_graph\: "</path/to/desired/
↪location>"" inventory/sqream/group_vars/all/docker.yml
```

5. Make the **kubespray_settings.sh** file executable for your user:

```
$ chmod u+x kubespray_settings.sh && ./kubespray_settings.sh
```

6. Run the following script:

```
$ ./kubespray_settings.sh
```

7. Run a playbook on the **inventory/sqream/hosts.yml cluster.yml** file:

```
$ ansible-playbook -i inventory/sqream/hosts.yml cluster.yml -v
```

The Kubespray installation takes approximately 10 - 15 minutes.

The following is an example of the correct output:

```
$ PLAY RECAP
$␣
↪*************************************************************************************************
$ node-1             : ok=680  changed=133  unreachable=0    failed=0
$ node-2             : ok=583  changed=113  unreachable=0    failed=0
$ node-3             : ok=586  changed=115  unreachable=0    failed=0
$ localhost          : ok=1    changed=0    unreachable=0    failed=0
```

In the event that the output is incorrect, or a failure occurred during the installation, please contact a SQream customer support representative.

Go back to *Installing Your Kubernetes Cluster*.

#### 2.1.4.3.4 Checking Your Kubernetes Status

After adjusting your Kubespray deployment values, you must check your Kubernetes status.

**To check your Kuberetes status:**

1. Check the status of the node:

```
$ kubectl get nodes
```

The following is an example of the correct output:

```
$ NAME         STATUS   ROLES                    AGE   VERSION
$ eks-rhl-1    Ready    control-plane,master     29m   v1.21.1
$ eks-rhl-2    Ready    control-plane,master     29m   v1.21.1
$ eks-rhl-3    Ready    <none>                   28m   v1.21.1
```

2. Check the status of the pod:

```
$ kubectl get pods --all-namespaces
```

The following is an example of the correct output:

```
$ NAMESPACE                NAME                                       ␣
↪READY   STATUS    RESTARTS   AGE
$ kube-system              calico-kube-controllers-68dc8bf4d5-n9pbp     1/
↪1      Running   0          160m
$ kube-system              calico-node-26cn9                            1/
↪1      Running   1          160m
$ kube-system              calico-node-kjsgw                            1/
↪1      Running   1          160m
$ kube-system              calico-node-vqvc5                            1/
↪1      Running   1          160m
$ kube-system              coredns-58687784f9-54xsp                     1/
↪1      Running   0          160m
$ kube-system              coredns-58687784f9-g94xb                     1/
↪1      Running   0          159m
$ kube-system              dns-autoscaler-79599df498-hlw8k              1/
↪1      Running   0          159m
$ kube-system              kube-apiserver-k8s-host-1-134                1/
↪1      Running   0          162m
$ kube-system              kube-apiserver-k8s-host-194                  1/
↪1      Running   0          161m
$ kube-system              kube-apiserver-k8s-host-68                   1/
↪1      Running   0          161m
$ kube-system              kube-controller-manager-k8s-host-1-134       1/
↪1      Running   0          162m
$ kube-system              kube-controller-manager-k8s-host-194         1/
↪1      Running   0          161m
$ kube-system              kube-controller-manager-k8s-host-68          1/
↪1      Running   0          161m
$ kube-system              kube-proxy-5f42q                             1/
↪1      Running   0          161m
$ kube-system              kube-proxy-bbwvk                             1/
↪1      Running   0          161m
$ kube-system              kube-proxy-fgcfb                             1/
↪1      Running   0          161m
$ kube-system              kube-scheduler-k8s-host-1-134                1/
↪1      Running   0          161m
```

(continues on next page)

(continued from previous page)

```
$ kube-system              kube-scheduler-k8s-host-194                    1/
↪1      Running    0             161m
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.5 Adding a SQream Label to Your Kubernetes Cluster Nodes

After checking your Kubernetes status, you must add a SQream label on your Kubernetes cluster nodes.

**To add a SQream label on your Kubernetes cluster nodes:**

1. Get the cluster node list:

```
$ kubectl get nodes
```

The following is an example of the correct output:

```
$ NAME         STATUS   ROLES                   AGE   VERSION
$ eks-rhl-1    Ready    control-plane,master    29m   v1.21.1
$ eks-rhl-2    Ready    control-plane,master    29m   v1.21.1
$ eks-rhl-3    Ready    <none>                  28m   v1.21.1
```

2. Set the node label, change the `node-name` to the node NAME(s) in the above example:

```
$ kubectl label nodes <node-name> cluster=sqream
```

The following is an example of the correct output:

```
$ [root@edk-rhl-1 kubespray]# kubectl label nodes eks-rhl-1 cluster=sqream
$ node/eks-rhl-1 labeled
$ [root@edk-rhl-1 kubespray]# kubectl label nodes eks-rhl-2 cluster=sqream
$ node/eks-rhl-2 labeled
$ [root@edk-rhl-1 kubespray]# kubectl label nodes eks-rhl-3 cluster=sqream
$ node/eks-rhl-3 labeled
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.6 Copying Your Kubernetes Configuration API File to the Master Cluster Nodes

After adding a SQream label on your Kubernetes cluster nodes, you must copy your Kubernetes configuration API file to your Master cluster nodes.

When the Kubernetes cluster installation is complete, an API configuration file is automatically created in the **.kube** folder of the root user. This file enables the **kubectl** command access Kubernetes' internal API service. Following this step lets you run **kubectl** commands from any node in the cluster.

> **Warning:** You must perform this on the management server only!

**To copy your Kubernetes configuration API file to your Master cluster nodes:**

1. Create the **.kube** folder in the **local user** directory:

```
$ mkdir /home/<local user>/.kube
```

2. Copy the configuration file from the root user directory to the <local user> directory:

```
$ sudo cp /root/.kube/config /home/<local user>/.kube
```

3. Change the file owner from **root user** to the <local user>:

```
$  sudo chown <local user>.<local user> /home/<local user>/.kube/config
```

4. Create the **.kube** folder in the other nodes located in the <local user> directory:

```
$ ssh <local user>@<node name> mkdir .kube
```

5. Copy the configuration file from the management node to the other nodes:

```
$ scp /home/<local user>/.kube/config <local user>@<node name>:/home/<local user>/
↪.kube/
```

6. Under local user on each server you copied **.kube** to, run the following command:

```
$ sudo usermod -aG docker $USER
```

This grants the local user the necessary permissions to run Docker commands.

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.7 Creating an env_file in Your Home Directory

After copying your Kubernetes configuration API file to your Master cluster nodes, you must create an **env_file** in your home directory, and must set the VIP address as a variable.

> **Warning:** You must perform this on the management server only!

**To create an env_file for local users in the user's home directory:**

1. Set a variable that includes the VIP IP address:

```
$ export VIP_IP=<VIP IP>
```

**Note:** If you use Kerberos, replace the `KRB5_SERVER` value with the IP address of your Kerberos server.

2. Do one of the following:

   - For local users:

   ```
   $ mkdir /home/$USER/.sqream
   ```

3. Make the following replacements to the **kubespray.settings.sh** file, verifying that the `KRB5_SERVER` parameter is set to your server IP:

```
$ cat <<EOF > /home/$USER/.sqream/env_file
SQREAM_K8S_VIP=$VIP_IP
SQREAM_ADMIN_UI_PORT=8080
SQREAM_DASHBOARD_DATA_COLLECTOR_PORT=8100
SQREAM_DATABASE_NAME=master
SQREAM_K8S_ADMIN_UI=sqream-admin-ui
SQREAM_K8S_DASHBOARD_DATA_COLLECTOR=dashboard-data-collector
SQREAM_K8S_METADATA=sqream-metadata
SQREAM_K8S_NAMESPACE=sqream
SQREAM_K8S_PICKER=sqream-picker
SQREAM_K8S_PROMETHEUS=prometheus
SQREAM_K8S_REGISTRY_PORT=6000
SQREAM_METADATA_PORT=3105
SQREAM_PICKER_PORT=3108
SQREAM_PROMETHEUS_PORT=9090
SQREAM_SPOOL_MEMORY_RATIO=0.25
SQREAM_WORKER_0_PORT=5000
KRB5CCNAME=FILE:/tmp/tgt
KRB5_SERVER=kdc.sq.com:<server IP>1
KRB5_CONFIG_DIR=${        $ SQREAM_MOUNT_DIR}/krb5
KRB5_CONFIG_FILE=${KRB5_CONFIG_DIR}/krb5.conf
HADOOP_CONFIG_DIR=${        $ SQREAM_MOUNT_DIR}/hadoop
HADOOP_CORE_XML=${HADOOP_CONFIG_DIR}/core-site.xml
HADOOP_HDFS_XML=${HADOOP_CONFIG_DIR}/hdfs-site.xml
EOF
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.8 Creating a Base Kubernetes Namespace

After creating an env_file in the user's home directory, you must create a base Kubernetes namespace.

You can create a Kubernetes namespace by running the following command:

```
$ kubectl create namespace sqream-init
```

The following is an example of the correct output:

```
$ namespace/sqream-init created
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.9 Pushing the env_file File to the Kubernetes Configmap

After creating a base Kubernetes namespace, you must push the **env_file** to the Kubernetes configmap. You must push the **env_file** file to the Kubernetes **configmap** in the **sqream-init** namespace.

This is done by running the following command:

```
$ kubectl create configmap sqream-init -n sqream-init --from-env-file=/home/$USER/.
↪sqream/env_file
```

The following is an example of the correct output:

```
$ configmap/sqream-init created
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.10 Installing the NVIDIA Docker2 Toolkit

After pushing the **env_file** file to the Kubernetes configmap, you must install the NVIDIA Docker2 Toolkit. The **NVIDIA Docker2 Toolkit** lets users build and run GPU-accelerated Docker containers, and must be run only on GPU servers. The NVIDIA Docker2 Toolkit includes a container runtime library and utilities that automatically configure containers to leverage NVIDIA GPUs.

##### 2.1.4.3.10.1 Installing the NVIDIA Docker2 Toolkit on an x86_64 Bit Processor on CentOS

**To install the NVIDIA Docker2 Toolkit on an x86_64 bit processor on CentOS:**

1. Add the repository for your distribution:

```
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
$ curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.
↪repo | \
$ sudo tee /etc/yum.repos.d/nvidia-docker.repo
```

2. Install the **nvidia-docker2** package and reload the Docker daemon configuration:

```
$ sudo yum install nvidia-docker2
$ sudo pkill -SIGHUP dockerd
```

3. Verify that the **nvidia-docker2** package has been installed correctly:

```
$ docker run --runtime=nvidia --rm nvidia/cuda:10.1-base nvidia-smi
```

The following is an example of the correct output:

```
$ docker run --runtime=nvidia --rm nvidia/cuda:10.1-base nvidia-smi
$ Unable to find image 'nvidia/cuda:10.1-base' locally
$ 10.1-base: Pulling from nvidia/cuda
$ d519e2592276: Pull complete
$ d22d2dfcfa9c: Pull complete
$ b3afe92c540b: Pull complete
$ 13a10df09dc1: Pull complete
$ 4f0bc36a7e1d: Pull complete
$ cd710321007d: Pull complete
$ Digest: sha256:635629544b2a2be3781246fdddc55cc1a7d8b352e2ef205ba6122b8404a52123
$ Status: Downloaded newer image for nvidia/cuda:10.1-base
$ Sun Feb 14 13:27:58 2021
$ +-----------------------------------------------------------------------------+
$ | NVIDIA-SMI 418.87.00    Driver Version: 418.87.00    CUDA Version: 10.1     |
$ |-------------------------------+----------------------+----------------------+
$ | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
$ | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
$ |===============================+======================+======================|
$ |   0  GeForce GTX 105...  Off  | 00000000:01:00.0 Off |                  N/A |
$ | 32%   37C    P0    N/A /  75W |      0MiB /  4039MiB |      0%      Default |
$ +-------------------------------+----------------------+----------------------+
```

(continues on next page)

(continued from previous page)

```
$
$ +-----------------------------------------------------------------------------+
$ | Processes:                                                       GPU Memory |
$ |  GPU       PID   Type   Process name                             Usage      |
$ |=============================================================================|
$ |  No running processes found                                                 |
$ +-----------------------------------------------------------------------------+
```

For more information on installing the NVIDIA Docker2 Toolkit on an x86_64 Bit Processor on CentOS, see NVIDIA Docker Installation - CentOS distributions

#### 2.1.4.3.10.2 Installing the NVIDIA Docker2 Toolkit on an x86_64 Bit Processor on Ubuntu

**To install the NVIDIA Docker2 Toolkit on an x86_64 bit processor on Ubuntu:**

1. Add the repository for your distribution:

```
$ curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
$ sudo apt-key add -
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
$ curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.
↪list | \
$ sudo tee /etc/apt/sources.list.d/nvidia-docker.list
$ sudo apt-get update
```

2. Install the **nvidia-docker2** package and reload the Docker daemon configuration:

```
$ sudo apt-get install nvidia-docker2
$ sudo pkill -SIGHUP dockerd
```

3. Verify that the nvidia-docker2 package has been installed correctly:

```
$ docker run --runtime=nvidia --rm nvidia/cuda nvidia-smi
```

For more information on installing the NVIDIA Docker2 Toolkit on an x86_64 Bit Processor on Ubuntu, see NVIDIA Docker Installation - Ubuntu distributions

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.11 Modifying the Docker Daemon JSON File for GPU and Compute Nodes

After installing the NVIDIA Docker2 toolkit, you must modify the Docker daemon JSON file for GPU and Compute nodes.

#### 2.1.4.3.11.1 Modifying the Docker Daemon JSON File for GPU Nodes

**To modify the Docker daemon JSON file for GPU nodes:**

1. Enable GPU and set HTTP access to the local Kubernetes Docker registry.

**Note:** The Docker daemon JSON file must be modified on all GPU nodes.

**Note:** Contact your IT department for a virtual IP.

2. Replace the `VIP address` with your assigned VIP address.
3. Connect as a root user:

```
$  sudo -i
```

4. Set a variable that includes the VIP address:

```
$ export VIP_IP=<VIP IP>
```

5. Replace the <VIP IP> with the VIP address:

```
$ cat <<EOF > /etc/docker/daemon.json
$ {
$    "insecure-registries": ["$VIP_IP:6000"],
$     "default-runtime": "nvidia",
$     "runtimes": {
$         "nvidia": {
$             "path": "nvidia-container-runtime",
$             "runtimeArgs": []
$         }
$     }
$ }
$ EOF
```

6. Apply the changes and restart Docker:

```
$ systemctl daemon-reload && systemctl restart docker
```

7. Exit the root user:

```
$ exit
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.11.2 Modifying the Docker Daemon JSON File for Compute Nodes

You must follow this procedure only if you have a Compute node.

**To modify the Docker daemon JSON file for Compute nodes:**

1. Switch to a root user:

```
$  sudo -i
```

2. Set a variable that includes a VIP address.

**Note:** Contact your IT department for a virtual IP.

3. Replace the `VIP address` with your assigned VIP address.

```
$ cat <<EOF > /etc/docker/daemon.json
$ {
$    "insecure-registries": ["$VIP_IP:6000"]
$ }
$ EOF
```

4. Restart the services:

```
$ systemctl daemon-reload && systemctl restart docker
```

5. Exit the root user:

```
$ exit
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.12 Installing the Nvidia-device-plugin Daemonset

After modifying the Docker daemon JSON file for GPU or Compute Nodes, you must installing the Nvidia-device-plugin daemonset. The Nvidia-device-plugin daemonset is only relevant to GPU nodes.

**To install the Nvidia-device-plugin daemonset:**

1. Set `nvidia.com/gpu` to `true` on all GPU nodes:

```
$ kubectl label nodes <GPU node name> nvidia.com/gpu=true
```

2. Replace the *<GPU node name>* with your GPU node name:

   For a complete list of GPU node names, run the `kubectl get nodes` command.

   The following is an example of the correct output:

```
$ [root@eks-rhl-1 ~]# kubectl label nodes eks-rhl-1 nvidia.com/gpu=true
$ node/eks-rhl-1 labeled
$ [root@eks-rhl-1 ~]# kubectl label nodes eks-rhl-2 nvidia.com/gpu=true
$ node/eks-rhl-2 labeled
$ [root@eks-rhl-1 ~]# kubectl label nodes eks-rhl-3 nvidia.com/gpu=true
$ node/eks-rhl-3 labeled
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.13 Creating an Nvidia Device Plugin

After installing the Nvidia-device-plugin daemonset, you must create an Nvidia-device-plugin. You can create an Nvidia-device-plugin by running the following command

```
$  kubectl create -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/1.0.0-
↪beta6/nvidia-device-plugin.yml
```

If needed, you can check the status of the Nvidia-device-plugin-daemonset pod status:

```
$ kubectl get pods -n kube-system -o wide | grep nvidia-device-plugin
```

The following is an example of the correct output:

```
$ NAME                                       READY   STATUS    RESTARTS   AGE
$ nvidia-device-plugin-daemonset-fxfct       1/1     Running   0          6h1m
$ nvidia-device-plugin-daemonset-jdvxs       1/1     Running   0          6h1m
$ nvidia-device-plugin-daemonset-xpmsv       1/1     Running   0          6h1m
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.14 Checking GPU Resources Allocatable to GPU Nodes

After creating an Nvidia Device Plugin, you must check the GPU resources alloctable to the GPU nodes. Each GPU node has records, such as `nvidia.com/gpu:  <#>`. The # indicates the number of allocatable, or available, GPUs in each node.

You can output a description of allocatable resources by running the following command:

```
$ kubectl describe node | grep -i -A 7 -B 2 allocatable:
```

The following is an example of the correct output:

```
$ Allocatable:
$  cpu:                    3800m
$  ephemeral-storage:      94999346224
$  hugepages-1Gi:          0
$  hugepages-2Mi:          0
$  memory:                 15605496Ki
$  nvidia.com/gpu:         1
$  pods:                   110
```

Go back to *Installing Your Kubernetes Cluster*

#### 2.1.4.3.15 Preparing the WatchDog Monitor

SQream's deployment includes installing two watchdog services. These services monitor Kuberenetes management and the server's storage network.

You can enable the storage watchdogs by adding entries in the **/etc/hosts** file on each server:

```
$ <address 1> k8s-node1.storage
$ <address 2> k8s-node2.storage
$ <address 3> k8s-node3.storage
```

The following is an example of the correct syntax:

```
$ 10.0.0.1 k8s-node1.storage
$ 10.0.0.2 k8s-node2.storage
$ 10.0.0.3 k8s-node3.storage
```

Go back to *Installing Your Kubernetes Cluster*

### 2.1.4.4 Installing the SQream Software

Once you've prepared the SQream environment for launching it using Kubernetes, you can begin installing the SQream software.

The **Installing the SQream Software** section describes the following:



#### 2.1.4.4.1 Getting the SQream Package

The first step in installing the SQream software is getting the SQream package. Please contact the SQream Support team to get the **sqream_k8s-nnn-DBnnn-COnnn-SDnnn-<arch>.tar.gz** tarball file.

This file includes the following values:

- **sqream_k8s-<nnn>** - the SQream installer version.
- **DB<nnn>** - the SQreamDB version.
- **CO<nnn>** - the SQream console version.
- **SD<nnn>** - the SQream Acceleration Studio version.
- **arch** - the server architecture.

You can extract the contents of the tarball by running the following command:

```
$ tar -xvf sqream_k8s-1.0.15-DB2020.1.0.2-SD0.7.3-x86_64.tar.gz
$ cd sqream_k8s-1.0.15-DB2020.1.0.2-SD0.7.3-x86_64
$ ls
```

Extracting the contents of the tarball file generates a new folder with the same name as the tarball file.

The following shows the output of the extracted file:

```
drwxrwxr-x. 2 sqream sqream    22 Jan 27 11:39 license
lrwxrwxrwx. 1 sqream sqream    49 Jan 27 11:39 sqream -> .sqream/sqream-sql-v2020.3.1_
↪stable.x86_64/sqream
-rwxrwxr-x. 1 sqream sqream  9465 Jan 27 11:39 sqream-install
-rwxrwxr-x. 1 sqream sqream 12444 Jan 27 11:39 sqream-start
```

Go back to *Installing Your SQream Software*

#### 2.1.4.4.2 Setting Up and Configuring Hadoop

After getting the SQream package, you can set up and configure Hadoop by configuring the **keytab** and **krb5.conf** files.

**Note:** You only need to configure the **keytab** and **krb5.conf** files if you use Hadoop with Kerberos authentication.

**To set up and configure Hadoop:**

1. Contact IT for the **keytab** and **krb5.conf** files.
2. Copy both files into the respective empty **.hadoop/** and **.krb5/** directories:

```
$ cp hdfs.keytab krb5.conf .krb5/
$ cp core-site.xml hdfs-site.xml .hadoop/
```

The SQream installer automatically copies the above files during the installation process.

Go back to *Installing Your SQream Software*

#### 2.1.4.4.3 Starting a Local Docker Image Registry

After getting the SQream package, or (optionally) setting up and configuring Hadoop, you must start a local Docker image registry. Because Kubernetes is based on Docker, you must start the local Docker image registry on the host's shared folder. This allows all hosts to pull the SQream Docker images.

**To start a local Docker image registry:**

1. Create a Docker registry folder:

   ```
   $ mkdir <shared path>/docker-registry/
   ```

2. Set the `docker_path` for the Docker registry folder:

   ```
   $ export docker_path=<path>
   ```

3. Apply the **docker-registry** service to the cluster:

   ```
   $ cat .k8s/admin/docker_registry.yaml | envsubst | kubectl create -f -
   ```

   The following is an example of the correct output:

   ```
   namespace/sqream-docker-registry created
   configmap/sqream-docker-registry-config created
   deployment.apps/sqream-docker-registry created
   service/sqream-docker-registry created
   ```

4. Check the pod status of the **docker-registry** service:

   ```
   $ kubectl get pods -n sqream-docker-registry
   ```

The following is an example of the correct output:

```
 NAME                                       READY   STATUS     RESTARTS    AGE
sqream-docker-registry-655889fc57-hmg7h    1/1     Running    0           6h40m
```

Go back to *Installing Your SQream Software*

#### 2.1.4.4.4 Installing the Kubernetes Dashboard

After starting a local Docker image registry, you must install the Kubernetes dashboard. The Kubernetes dashboard lets you see the Kubernetes cluster, nodes, services, and pod status.

**To install the Kubernetes dashboard:**

1. Apply the **k8s-dashboard** service to the cluster:

```
$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/
↪aio/deploy/recommended.yaml
```

The following is an example of the correct output:

```
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

2. Grant the user external access to the Kubernetes dashboard:

```
$ cat .k8s/admin/kubernetes-dashboard-svc-metallb.yaml | envsubst | kubectl␣
↪create -f -
```

The following is an example of the correct output:

```
service/kubernetes-dashboard-nodeport created
```

3. Create the `cluster-admin-sa.yaml` file:

```
$ kubectl create -f .k8s/admin/cluster-admin-sa.yaml
```

The following is an example of the correct output:

```
clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-sa-cluster-admin␣
↪created
```

4. Check the pod status of the **K8s-dashboard** service:

```
$ kubectl get pods -n kubernetes-dashboard
```

The following is an example of the correct output:

```
NAME                                         READY   STATUS    RESTARTS   AGE
dashboard-metrics-scraper-6b4884c9d5-n8p57   1/1     Running   0          4m32s
kubernetes-dashboard-7b544877d5-qc8b4        1/1     Running   0          4m32s
```

5. Obtain the **k8s-dashboard** access token:

```
$ kubectl -n kube-system describe secrets cluster-admin-sa-token
```

The following is an example of the correct output:

```
Name:         cluster-admin-sa-token-rbl9p
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: cluster-admin-sa
              kubernetes.io/service-account.uid: 81866d6d-8ef3-4805-840d-
↪58618235f68d

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1025 bytes
namespace:  11 bytes
token:      ␣
↪eyJhbGciOiJSUzI1NiIsImtpZCI6IjRMV09qVzFabjhId09oamQzZGFFNmZBeEFzOHp3SlJOZWdtVm5lVTdtSW8ifQ.
↪eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3
↪mNhp8JMr5y3hQ44QrvRDCMueyjSHSrmqZcoV00ZC7iBzNUqh3n-fB99CvC_
↪GR15ys43jnfsz0tdsTy7VtSc9hm5ENBI-tQ_mwT1Zc7zJrEtgFiA0o_
↪eyfYZOARdhdyFEJg84bzkIxJFPKkBWb4iPWU1Xb7RibuMCjNTarZMZbqzKYfQEcMZWJ5UmfUqp-
↪HahZZR4BNbjSWybs7t6RWdcQZt6sO_rRCDrOeEJlqKKjx4-5jFZB8Du_
↪0kKmnw2YJmmSCEOXrpQCyXIiZJpX08HyDDYfFp8IGzm61arB8HDA9dN_
↪xoWvuz4Cj8klUtTzL9effJJPjHJlZXcEqQc9hE3jw
```

6. Navigate to `https://<VIP address>:5999`.
7. Select the **Token** radio button, paste the token from the previous command output, and click **Sign in**.

The Kubernetes dashboard is displayed.

Go back to *Installing Your SQream Software*

#### 2.1.4.4.5 Installing the SQream Prometheus Package

After installing the Kubernetes dashboard, you must install the SQream Prometheus package. To properly monitor the host and GPU statistics the **exporter service** must be installed on each Kubernetes cluster node.

This section describes how to install the following:

- **node_exporter** - collects host data, such as CPU memory usage.
- **nvidia_exporter** - collects GPU utilization data.

**Note:** The steps in this section must be done on **all** cluster nodes.

To install the **sqream-prometheus** package, you must do the following:

1. *Install the exporter service*
2. *Check the exporter service*

Go back to *Installing Your SQream Software*

##### 2.1.4.4.5.1 Installing the Exporter Service

**To install the exporter service:**

1. Create a user and group that will be used to run the exporter services:

```
$ sudo groupadd --system prometheus && sudo useradd -s /sbin/nologin --system -g␣
↪prometheus prometheus
```

2. Extract the **sqream_exporters_prometheus.0.1.tar.gz** file:

```
$ cd .prometheus
$ tar -xf sqream_exporters_prometheus.0.1.tar.gz
```

3. Copy the exporter software files to the **/usr/bin** directory:

```
$ cd  sqream_exporters_prometheus.0.1
$ sudo cp node_exporter/node_exporter /usr/bin/
$ sudo cp nvidia_exporter/nvidia_exporter /usr/bin/
```

4. Copy the exporters service file to the **/etc/systemd/system/** directory:

```
$ sudo cp services/node_exporter.service /etc/systemd/system/
$ sudo cp services/nvidia_exporter.service /etc/systemd/system/
```

5. Set the permission and group of the service files:

```
$ sudo chown prometheus:prometheus /usr/bin/node_exporter
$ sudo chmod u+x /usr/bin/node_exporter
$ sudo chown prometheus:prometheus /usr/bin/nvidia_exporter
$ sudo chmod u+x /usr/bin/nvidia_exporter
```

6. Reload the services:

```
$ sudo systemctl daemon-reload
```

7. Start both services and set them to start when the server is booted up:

   - Node_exporter:

   ```
   $ sudo systemctl start node_exporter && sudo systemctl enable node_exporter
   ```

   - Nvidia_exporter:

   ```
   $ sudo systemctl start nvidia_exporter && sudo systemctl enable nvidia_
   ↪exporter
   ```

##### 2.1.4.4.5.2 Checking the Exporter Status

After installing the **exporter** service, you must check its status.

You can check the exporter status by running the following command:

```
$ sudo systemctl status node_exporter && sudo systemctl status nvidia_exporter
```

Go back to *Installing Your SQream Software*

#### 2.1.4.5 Running the Sqream-install Service

The **Running the Sqream-install Service** section describes the following:



##### 2.1.4.5.1 Installing Your License

After install the SQream Prometheus package, you must install your license.

**To install your license:**

1. Copy your license package to the sqream **/license** folder.

**Note:** You do not need to untar the license package after copying it to the **/license** folder because the installer script does it automatically.

The following flags are **mandatory** during your first run:

```
$ sudo ./sqream-install -i -k -m <path to sqream cluster>
```

**Note:** If you cannot run the script with **sudo**, verify that you have the right permission (**rwx** for the user) on the relevant directories (config, log, volume, and data-in directories).

Go back to *Running the SQream_install Service*.

##### 2.1.4.5.2 Changing Your Data Ingest Folder

After installing your license, you must change your data ingest folder.

You can change your data ingest folder by running the following command:

```
$ sudo ./sqream-install -d /media/nfs/sqream/data_in
```

Go back to *Running the SQream_install Service*.

#### 2.1.4.5.3 Checking Your System Settings

After changing your data ingest folder, you must check your system settings.

The following command shows you all the variables that your SQream system is running with:

```
$ ./sqream-install -s
```

After optionally checking your system settings, you can use the **sqream-start** application to control your Kubernetes cluster.

Go back to *Running the SQream_install Service*.

#### 2.1.4.5.4 SQream Installation Command Reference

If needed, you can use the **sqream-install** flag reference for any needed flags by typing:

```
$ ./sqream-install --help
```

The following shows the **sqream–install** flag descriptions:

| Flag | Function | Note |
|---|---|---|
| **-i** | Loads all the software from the hidden **.docker** folder. | Mandatory |
| **-k** | Loads the license package from the **/license** directory. | Mandatory |
| **-m** | Sets the relative path for all SQream folders under the shared filesystem available from all nodes (sqreamdb, config, logs and data_in). No other flags are required if you use this flag (such as c, v, l or d). | Mandatory |
| **-c** | Sets the path where to write/read SQream configuration files from. The default is **/etc/sqream/**. | Optional |
| **-v** | Shows the location of the SQream cluster. `v` creates a cluster if none exist, and mounts it if does. | Optional |
| **-l** | Shows the location of the SQream system startup logs. The logs contain startup and Docker logs. The default is **/var/log/sqream/**. | Optional |
| **-d** | Shows the folder containing data that you want to import into or copy from SQream. | Optional |
| **-n** <Namespace> | Sets the Kubernetes namespace. The default is **sqream**. | Optional |
| **-N** <Namespace> | Deletes a specific Kubernetes namespace and sets the factory default namespace (sqream). | Optional |
| **-f** | Overwrite existing folders and all files located in mounted directories. | Optional |
| **-r** | Resets the system configuration. This flag is run without any other variables. | Optional |
| **-s** | Shows the system settings. | Optional |
| **-e** | Sets the Kubernetes cluster's virtual IP address. | Optional |
| **-h** | Help, shows all available flags. | Optional |

Go back to *Running the SQream_install Service*.

#### 2.1.4.5.5 Controlling Your Kubernetes Cluster Using SQream Flags

You can control your Kubernetes cluster using SQream flags.

The following command shows you the available Kubernetes cluster control options:

```
$ ./sqream-start -h
```

The following describes the **sqream-start** flags:

| Flag | Function | Note |
|---|---|---|
| **-s** | Starts the sqream services, starting metadata, server picker, and workers. The number of workers started is based on the number of available GPU's. | Mandatory |
| **-p** | Sets specific ports to the workers services. You must enter the starting port for the sqream-start application to allocate it based on the number of workers. | |
| **-j** | Uses an external .json configuration file. The file must be located in the **configuration** directory. | The workers must each be started individually. |
| **-m** | Allocates worker spool memory. | The workers must each be started individually. |
| **-a** | Starts the SQream Administration dashboard and specifies the listening port. | |
| **-d** | Deletes all running SQream services. | |
| **-h** | Shows all available flags. | Help |

Go back to *Running the SQream_install Service*.

### 2.1.4.6 Using the sqream-start Commands

In addition to controlling your Kubernetes cluster using SQream flags, you can control it using **sqream-start** commands.

The **Using the sqream-start Commands** section describes the following:

#### 2.1.4.6.1 Starting Your SQream Services

You can run the **sqream-start** command with the **-s** flag to start SQream services on all available GPU's:

```
$ sudo ./sqream-start -s
```

This command starts the SQream metadata, server picker, and sqream workers on all available GPU's in the cluster.

The following is an example of the correct output:

```
./sqream-start -s
Initializing network watchdogs on 3 hosts...
Network watchdogs are up and running

Initializing 3 worker data collectors ...
Worker data collectors are up and running

Starting Prometheus ...
Prometheus is available at 192.168.5.100:9090

Starting SQream master ...
SQream master is up and running

Starting up 3 SQream workers ...
All SQream workers are up and running, SQream-DB is available at 192.168.5.100:3108
All SQream workers are up and running, SQream-DB is available at 192.168.5.100:3108
```

Go back to *Using the SQream-start Commands*.

#### 2.1.4.6.2 Starting Your SQream Services in Split Mode

Starting SQream services in split mode refers to running multiple SQream workers on a single GPU. You can do this by running the **sqream-start** command with the **-s** and **-z** flags. In addition, you can define the amount of hosts to run the multiple workers on. In the example below, the command defines to run the multiple workers on three hosts.

**To start SQream services in split mode:**

1. Run the following command:

```
$ ./sqream-start -s -z 3
```

This command starts the SQream metadata, server picker, and sqream workers on a single GPU for three hosts:

The following is an example of the correct output:

```
Initializing network watchdogs on 3 hosts...
Network watchdogs are up and running

Initializing 3 worker data collectors ...
Worker data collectors are up and running

Starting Prometheus ...
Prometheus is available at 192.168.5.101:9090

Starting SQream master ...
SQream master is up and running
```

(continues on next page)

(continued from previous page)

```
Starting up 9 SQream workers over <#> available GPUs ...
All SQream workers are up and running, SQream-DB is available at 192.168.5.101:3108
```

2. Verify all pods are properly running in k8s cluster (**STATUS** column):

```
kubectl -n sqream get pods

NAME                                          READY   STATUS              RESTARTS   ␣
↪AGE
prometheus-bcf877867-kxhld                    1/1     Running             0          ␣
↪106s
sqream-metadata-fbcbc989f-6zlkx               1/1     Running             0          ␣
↪103s
sqream-picker-64b8c57ff5-ndfr9                1/1     Running             2          ␣
↪102s
sqream-split-workers-0-1-2-6bdbfbbb86-ml7kn   1/1     Running             0          ␣
↪57s
sqream-split-workers-3-4-5-5cb49d49d7-596n4   1/1     Running             0          ␣
↪57s
sqream-split-workers-6-7-8-6d598f4b68-2n9z5   1/1     Running             0          ␣
↪56s
sqream-workers-start-xj75g                    1/1     Running             0          ␣
↪58s
watchdog-network-management-6dnfh             1/1     Running             0          ␣
↪115s
watchdog-network-management-tfd46             1/1     Running             0          ␣
↪115s
watchdog-network-management-xct4d             1/1     Running             0          ␣
↪115s
watchdog-network-storage-lr6v4                1/1     Running             0          ␣
↪116s
watchdog-network-storage-s29h7                1/1     Running             0          ␣
↪116s
watchdog-network-storage-sx9mw                1/1     Running             0          ␣
↪116s
worker-data-collector-62rxs                   0/1     Init:0/1            0          ␣
↪54s
worker-data-collector-n8jsv                   0/1     Init:0/1            0          ␣
↪55s
worker-data-collector-zp8vf                   0/1     Init:0/1            0          ␣
↪54s
```

Go back to *Using the SQream-start Commands*.

#### 2.1.4.6.3 Starting the Sqream Studio UI

You can run the following command the to start the SQream Studio UI (Editor and Dashboard):

```
$ ./sqream-start -a
```

The following is an example of the correct output:

```
$ ./sqream-start -a
Please enter USERNAME:
sqream
```

(continues on next page)

(continued from previous page)

```
Please enter PASSWORD:
******
Please enter port value or press ENTER to keep 8080:

Starting up SQream Admin UI...
SQream admin ui is available at 192.168.5.100:8080
```

Go back to *Using the SQream-start Commands*.

#### 2.1.4.6.4 Stopping the SQream Services

You can run the following command to stop all SQream services:

```
$ ./sqream-start -d
```

The following is an example of the correct output:

```
$ ./sqream-start -d
$ Cleaning all SQream services in sqream namespace ...
$ All SQream service removed from sqream namespace
```

Go back to *Using the SQream-start Commands*.

#### 2.1.4.6.5 Advanced sqream-start Commands

##### 2.1.4.6.5.1 Controlling Your SQream Spool Size

If you do not specify the SQream spool size, the console automatically distributes the available RAM between all running workers.

You can define a specific spool size by running the following command:

```
$ ./sqream-start -s -m 4
```

##### 2.1.4.6.5.2 Using a Custom .json File

You have the option of using your own .json file for your own custom configurations. Your .json file must be placed within the path mounted in the installation. SQream recommends placing your .json file in the **configuration** folder.

The SQream console does not validate the integrity of external .json files.

You can use the following command (using the `j` flag) to set the full path of your .json file to the configuration file:

```
$ ./sqream-start -s -f <full path>.json
```

This command starts one worker with an external configuration file.

**Note:** The configuration file must be available in the shared configuration folder.

#### 2.1.4.6.5.3 Checking the Status of the SQream Services

You can show all running SQream services by running the following command:

```
$ kubectl get pods -n <namespace> -o wide
```

This command shows all running services in the cluster and which nodes they are running in.

Go back to *Using the SQream-start Commands*.

### 2.1.4.7 Upgrading Your SQream Version

The **Upgrading Your SQream Version** section describes the following:



#### 2.1.4.7.1 Before Upgrading Your System

Before upgrading your system you must do the following:

1. Contact SQream support for a new SQream package tarball file.
2. Set a maintenance window.

**Note:** You must stop the system while upgrading it.

#### 2.1.4.7.2 Upgrading Your System

After completing the steps in **Before Upgrading Your System** above, you can upgrade your system.

**To upgrade your system:**

1. Extract the contents of the tarball file that you received from SQream support. Make sure to extract the contents to the same directory as in *Getting the SQream Package* and for the same user:

   ```
   $ tar -xvf sqream_installer-2.0.5-DB2019.2.1-CO1.6.3-ED3.0.0-x86_64/
   $ cd sqream_installer-2.0.5-DB2019.2.1-CO1.6.3-ED3.0.0-x86_64/
   ```

2. Start the upgrade process run the following command:

   ```
   $ ./sqream-install -i
   ```

The upgrade process checks if the SQream services are running and will prompt you to stop them.

3. Do one of the following:

   - Stop the upgrade by writing `No`.
   - Continue the upgrade by writing `Yes`.

If you continue upgrading, all running SQream workers (master and editor) are stopped. When all services have been stopped, the new version is loaded.

---

**Note:** SQream periodically upgrades its metadata structure. If an upgrade version includes an upgraded metadata service, an approval request message is displayed. This approval is required to finish the upgrade process. Because SQream supports only specific metadata versions, all SQream services must be upgraded at the same time.

---

4. When SQream has successfully upgraded, load the SQream console and restart your services.

For questions, contact SQream Support.

### 2.1.5 Installing Monit

#### 2.1.5.1 Getting Started

Before installing SQream with Monit, verify that you have followed the required *recommended pre-installation configurations*.

The procedures in the **Installing Monit** guide must be performed on each SQream cluster node.

#### 2.1.5.2 Overview

Monit is a free open source supervision utility for managing and monitoring Unix and Linux. Monit lets you view system status directly from the command line or from a native HTTP web server. Monit can be used to conduct automatic maintenance and repair, such as executing meaningful causal actions in error situations.

SQream uses Monit as a watchdog utility, but you can use any other utility that provides the same or similar functionality.

The **Installing Monit** procedures describes how to install, configure, and start Monit.

You can install Monit in one of the following ways:

- *Installing Monit on CentOS*
- *Installing Monit on CentOS offline*
- *Installing Monit on Ubuntu*
- *Installing Monit on Ubuntu offline*

##### 2.1.5.2.1 Installing Monit on CentOS:

**To install Monit on CentOS:**

1. Install Monit as a superuser on CentOS:

```
$ sudo yum install monit
```

#### 2.1.5.2.2 Installing Monit on CentOS Offline:

Installing Monit on CentOS offline can be done in either of the following ways:

- *Building Monit from Source Code*
- *Building Monit from Pre-Built Binaries*

##### 2.1.5.2.2.1 Building Monit from Source Code

**To build Monit from source code:**

1. Copy the Monit package for the current version:

   ```
   $ tar zxvf monit-<x.y.z>.tar.gz
   ```

   The value `x.y.z` denotes the version numbers.

2. Navigate to the directory where you want to store the package:

   ```
   $ cd monit-x.y.z
   ```

3. Configure the files in the package:

   ```
   $ ./configure (use ./configure --help to view available options)
   ```

4. Build and install the package:

   ```
   $ make && make install
   ```

The following are the default storage directories:

- The Monit package: **/usr/local/bin/**
- The **monit.1 man-file**: **/usr/local/man/man1/**

5. **Optional** - To change the above default location(s), use the **–prefix** option to ./configure.

6. **Optional** - Create an RPM package for CentOS directly from the source code:

   ```
   $ rpmbuild -tb monit-x.y.z.tar.gz
   ```

##### 2.1.5.2.2.2 Building Monit from Pre-Built Binaries

**To build Monit from pre-built binaries:**

1. Copy the Monit package for the current version:

   ```
   $ tar zxvf monit-x.y.z-linux-x64.tar.gz
   ```

   The value `x.y.z` denotes the version numbers.

2. Navigate to the directory where you want to store the package:

3. Copy the **bin/monit** and **/usr/local/bin/** directories:

   ```
   $ cp bin/monit /usr/local/bin/
   ```

4. Copy the **conf/monitrc** and **/etc/** directories:

```
$ cp conf/monitrc /etc/
```

For examples of pre-built Monit binarties, see Download Precompiled Binaries.

*Back to top*

#### 2.1.5.2.3 Installing Monit on Ubuntu:

**To install Monit on Ubuntu:**

1. Install Monit as a superuser on Ubuntu:

   ```
   $ sudo apt-get install monit
   ```

*Back to top*

#### 2.1.5.2.4 Installing Monit on Ubuntu Offline:

You can install Monit on Ubuntu when you do not have an internet connection.

**To install Monit on Ubuntu offline:**

1. Compress the required file:

   ```
   $ tar zxvf monit-<x.y.z>-linux-x64.tar.gz
   ```

   **NOTICE:** <*x.y.z*> denotes the version number.

2. Navigate to the directory where you want to save the file:

   ```
   $ cd monit-x.y.z
   ```

3. Copy the **bin/monit** directory into the **/usr/local/bin/** directory:

   ```
   $ cp bin/monit /usr/local/bin/
   ```

4. Copy the **conf/monitrc** directory into the **/etc/** directory:

   ```
   $ cp conf/monitrc /etc/
   ```

*Back to top*

### 2.1.5.3 Configuring Monit

When the installation is complete, you can configure Monit. You configure Monit by modifying the Monit configuration file, called **monitrc**. This file contains blocks for each service that you want to monitor.

The following is an example of a service block:

```
$ #SQREAM1-START
$ check process sqream1 with pidfile /var/run/sqream1.pid
$ start program = "/usr/bin/systemctl start sqream1"
$ stop program = "/usr/bin/systemctl stop sqream1"
$ #SQREAM1-END
```

For example, if you have 16 services, you can configure this block by copying the entire block 15 times and modifying all service names as required, as shown below:

```
$ #SQREAM2-START
$ check process sqream2 with pidfile /var/run/sqream2.pid
$ start program = "/usr/bin/systemctl start sqream2"
$ stop program = "/usr/bin/systemctl stop sqream2"
$ #SQREAM2-END
```

For servers that don't run the **metadataserver** and **serverpicker** commands, you can use the block example above, but comment out the related commands, as shown below:

```
$ #METADATASERVER-START
$ #check process metadataserver with pidfile /var/run/metadataserver.pid
$ #start program = "/usr/bin/systemctl start metadataserver"
$ #stop program = "/usr/bin/systemctl stop metadataserver"
$ #METADATASERVER-END
```

**To configure Monit:**

1. Copy the required block for each required service.
2. Modify all service names in the block.
3. Copy the configured **monitrc** file to the **/etc/monit.d/** directory:

```
$ cp monitrc /etc/monit.d/
```

4. Set file permissions to **600** (full read and write access):

```
$ sudo chmod 600 /etc/monit.d/monitrc
```

5. Reload the system to activate the current configurations:

```
$ sudo systemctl daemon-reload
```

6. **Optional** - Navigate to the **/etc/sqream** directory and create a symbolic link to the **monitrc** file:

```
$ cd /etc/sqream
$ sudo ln -s /etc/monit.d/monitrc monitrc
```

#### 2.1.5.4 Starting Monit

After configuring Monit, you can start it.

**To start Monit:**

1. Start Monit as a super user:

```
$ sudo systemctl start monit
```

2. View Monit's service status:

```
$ sudo systemctl status monit
```

3. If Monit is functioning correctly, enable the Monit service to start on boot:

```
$ sudo systemctl enable monit
```

### 2.1.6 Launching SQream with Monit

This procedure describes how to launch SQream using Monit.

#### 2.1.6.1 Launching SQream

After doing the following, you can launch SQream according to the instructions on this page.

1. *Installing Monit*
2. *Installing SQream with Binary*

The following is an example of a working monitrc file configured to monitor the ***metadataserver** and **serverpicker** commands, and **four sqreamd services**. The **monitrc** configuration file is located in the **conf/monitrc** directory.

Note that the **monitrc** in the following example is configured for eight `sqreamd` services, but that only the first four are enabled:

```
$ set daemon  5              # check services at 30 seconds intervals
$ set logfile syslog
$
$ set httpd port 2812 and
$      use address localhost  # only accept connection from localhost
$      allow localhost        # allow localhost to connect to the server and
$      allow admin:monit      # require user 'admin' with password 'monit'
$
$  ##set mailserver smtp.gmail.com port 587
$  ##        using tlsv12
$  #METADATASERVER-START
$  check process metadataserver with pidfile /var/run/metadataserver.pid
$  start program = "/usr/bin/systemctl start metadataserver"
$  stop program = "/usr/bin/systemctl stop metadataserver"
$  #METADATASERVER-END
$  #      alert user@domain.com on {nonexist, timeout}
$  #                     with mail-format {
$  #                           from:     Monit@$HOST
$  #                           subject:  metadataserver $EVENT - $ACTION
$  #                           message:  This is an automate mail, sent from monit.
$  #                     }
$  #SERVERPICKER-START
$  check process serverpicker with pidfile /var/run/serverpicker.pid
$  start program = "/usr/bin/systemctl start serverpicker"
$  stop program = "/usr/bin/systemctl stop serverpicker"
$  #SERVERPICKER-END
$  #       alert user@domain.com on {nonexist, timeout}
$  #                                    with mail-format {
$  #                                          from:     Monit@$HOST
$  #                                          subject:  serverpicker $EVENT - $ACTION
$  #                                         message:  This is an automate mail, sent␣
↪from monit.
$  #
$  #
$  #SQREAM1-START
$  check process sqream1 with pidfile /var/run/sqream1.pid
```

(continues on next page)

(continued from previous page)

```
$   start program = "/usr/bin/systemctl start sqream1"
$   stop program = "/usr/bin/systemctl stop sqream1"
$   #SQREAM1-END
$   #         alert user@domain.com on {nonexist, timeout}
$   #                with mail-format {
$   #                      from:     Monit@$HOST
$   #                      subject:  sqream1 $EVENT - $ACTION
$   #                      message:  This is an automate mail, sent from monit.
$   #               }
$   #SQREAM2-START
$   check process sqream2 with pidfile /var/run/sqream2.pid
$   start program = "/usr/bin/systemctl start sqream2"
$   #SQREAM2-END
$   #        alert user@domain.com on {nonexist, timeout}
$   #                with mail-format {
$   #                      from:     Monit@$HOST
$   #                      subject:  sqream1 $EVENT - $ACTION
$   #                      message:  This is an automate mail, sent from monit.
$   #               }
$   #SQREAM3-START
$   check process sqream3 with pidfile /var/run/sqream3.pid
$   start program = "/usr/bin/systemctl start sqream3"
$   stop program = "/usr/bin/systemctl stop sqream3"
$   #SQREAM3-END
$   #        alert user@domain.com on {nonexist, timeout}
$   #                with mail-format {
$   #                      from:     Monit@$HOST
$   #                      subject:  sqream2 $EVENT - $ACTION
$   #                      message:  This is an automate mail, sent from monit.
$   #               }
$   #SQREAM4-START
$   check process sqream4 with pidfile /var/run/sqream4.pid
$   start program = "/usr/bin/systemctl start sqream4"
$   stop program = "/usr/bin/systemctl stop sqream4"
$   #SQREAM4-END
$   #        alert user@domain.com on {nonexist, timeout}
$   #                       with mail-format {
$   #                             from:     Monit@$HOST
$   #                             subject:  sqream2 $EVENT - $ACTION
$   #                             message:  This is an automate mail, sent from monit.
$   #                      }
$   #
$   #SQREAM5-START
$   #check process sqream5 with pidfile /var/run/sqream5.pid
$   #start program = "/usr/bin/systemctl start sqream5"
$   #stop program = "/usr/bin/systemctl stop sqream5"
$   #SQREAM5-END
$   #        alert user@domain.com on {nonexist, timeout}
$   #                       with mail-format {
$   #                             from:     Monit@$HOST
$   #                             subject:  sqream2 $EVENT - $ACTION
$   #                             message:  This is an automate mail, sent from monit.
$   #                      }
$   #
$   #SQREAM6-START
$   #check process sqream6 with pidfile /var/run/sqream6.pid
$   #start program = "/usr/bin/systemctl start sqream6"
```

(continues on next page)

(continued from previous page)

```
$   #stop program = "/usr/bin/systemctl stop sqream6"
$   #SQREAM6-END
$   #       alert user@domain.com on {nonexist, timeout}
$   #                      with mail-format {
$   #                            from:     Monit@$HOST
$   #                            subject:  sqream2 $EVENT - $ACTION
$   #                            message:  This is an automate mail, sent from monit.
$   #                    }
$   #
$   #SQREAM7-START
$   #check process sqream7 with pidfile /var/run/sqream7.pid
$   #start program = "/usr/bin/systemctl start sqream7"
$   #stop program = "/usr/bin/systemctl stop sqream7"
$   #SQREAM7-END
$   #                      with mail-format {
$   #                            from:     Monit@$HOST
$   #                            subject:  sqream2 $EVENT - $ACTION
$   #                            message:  This is an automate mail, sent from monit.
$   #                    }
$   #
$   #SQREAM8-START
$   #check process sqream8 with pidfile /var/run/sqream8.pid
$   #start program = "/usr/bin/systemctl start sqream8"
$   #stop program = "/usr/bin/systemctl stop sqream8"
$   #SQREAM8-END
$   #       alert user@domain.com on {nonexist, timeout}
$   #                      with mail-format {
$   #                            from:     Monit@$HOST
$   #                            subject:  sqream2 $EVENT - $ACTION
$   #                            message:  This is an automate mail, sent from monit.
$   #                    }
```

### 2.1.6.2 Monit Usage Examples

This section shows examples of two methods for stopping the **sqream3** service use Monit's command syntax:

- *Stopping Monit and SQream separately*
- *Stopping SQream using a Monit command*

#### 2.1.6.2.1 Stopping Monit and SQream Separately

You can stop the Monit service and SQream separately as follows:

```
$ sudo systemctl stop monit
$ sudo systemctl stop sqream3
```

You can restart Monit as follows:

```
$ sudo systemctl start monit
```

Restarting Monit automatically restarts the SQream services.

#### 2.1.6.2.2 Stopping SQream Using a Monit Command

You can stop SQream using a Monit command as follows:

```
$ sudo monit stop sqream3
```

This command stops SQream only (and not Monit).

You can restart SQream as follows:

```
$ sudo monit start sqream3
```

#### 2.1.6.2.3 Monit Command Line Options

The **Monit Command Line Options** section describes some of the most commonly used Monit command options.

You can show the command line options by running:

```
$ monit --help
```

```
$ start all             - Start all services
$ start <name>          - Only start the named service
$ stop all              - Stop all services
$ stop <name>           - Stop the named service
$ restart all           - Stop and start all services
$ restart <name>        - Only restart the named service
$ monitor all           - Enable monitoring of all services
$ monitor <name>        - Only enable monitoring of the named service
$ unmonitor all         - Disable monitoring of all services
$ unmonitor <name>      - Only disable monitoring of the named service
$ reload                - Reinitialize monit
$ status [name]         - Print full status information for service(s)
$ summary [name]        - Print short status information for service(s)
$ report [up|down|..]   - Report state of services. See manual for options
$ quit                  - Kill the monit daemon process
$ validate              - Check all services and start if not running
$ procmatch <pattern>   - Test process matching pattern
```

### 2.1.6.3 Using Monit While Upgrading Your Version of SQream

While upgrading your version of SQream, you can use Monit to avoid conflicts (such as service start). This is done by pausing or stopping all running services while you manually upgrade SQream. When you finish successfully upgrading SQream, you can use Monit to restart all SQream services

**To use Monit while upgrading your version of SQream:**

1. Stop all actively running SQream services:

   ```
   $ sudo monit stop all
   ```

2. Verify that SQream has stopped listening on ports **500X**, **510X**, and **310X**:

   ```
   $ sudo netstat -nltp    #to make sure sqream stopped listening on 500X, 510X and␣
   ↪310X ports.
   ```

The example below shows the old version `sqream-db-v2020.2` being replaced with the new version `sqream-db-v2025.200`.

```
$ cd /home/sqream
$ mkdir tempfolder
$ mv sqream-db-v2025.200.tar.gz tempfolder/
$ tar -xf sqream-db-v2025.200.tar.gz
$ sudo mv sqream /usr/local/sqream-db-v2025.200
$ cd /usr/local
$ sudo chown -R sqream:sqream sqream-db-v2025.200
$ sudo rm sqream   #This only should remove symlink
$ sudo ln -s sqream-db-v2025.200 sqream   #this will create new symlink named
↪"sqream" pointing to new version
$ ls -l
```

The symbolic SQream link should point to the real folder:

```
$ sqream -> sqream-db-v2025.200
```

4. Restart the SQream services:

```
$ sudo monit start all
```

5. Verify that the latest version has been installed:

```
$ SELECT SHOW_VERSION();
```

The correct version is output.

6. Restart the UI:

```
$ pm2 start all
```

## 2.2 Installing SQream Studio

The **Installing SQream Studio** page incudes the following installation guides:

### 2.2.1 Installing Prometheus Exporter

The **Installing Prometheus Exporters** guide includes the following sections:

#### 2.2.1.1 Overview

The **Prometheus** exporter is an open-source systems monitoring and alerting toolkit. It is used for collecting metrics from an operating system and exporting them to a graphic user interface.

The Installing Prometheus Exporters guide describes how to installing the following exporters:

- The **Node_exporter** - the basic exporter used for displaying server metrics, such as CPU and memory.
- The **Nvidia_exporter** - shows Nvidia GPU metrics.
- The **process_exporter** - shows data belonging to the server's running processes.

For information about more exporters, see Exporters and Integration

#### 2.2.1.2 Adding a User and Group

Adding a user and group determines who can run processes.

You can add users with the following command:

```
$ sudo groupadd --system prometheus
```

You can add groups with the following command:

```
$ sudo useradd -s /sbin/nologin --system -g prometheus prometheus
```

#### 2.2.1.3 Cloning the Prometheus GIT Project

After adding a user and group you must clone the Prometheus GIT project.

You can clone the Prometheus GIT project with the following command:

```
$ git clone http://gitlab.sq.l/IT/promethues.git prometheus
```

**Note:** If you experience difficulties cloning the Prometheus GIT project or receive an error, contact your IT department.

The following shows the result of cloning your Prometheus GIT project:

```
$ prometheus/
$ ├── node_exporter
$ │   └── node_exporter
$ ├── nvidia_exporter
$ │   └── nvidia_exporter
$ ├── process_exporter
$ │   └── process-exporter_0.5.0_linux_amd64.rpm
$ ├── README.md
$ └── services
$     ├── node_exporter.service
$     └── nvidia_exporter.service
```

#### 2.2.1.4 Installing the Node Exporter and NVIDIA Exporter

After cloning the Prometheus GIT project you must install the **node_exporter** and **NVIDIA_exporter**.

**To install the node_exporter and NVIDIA_exporter:**

1. Navigate to the cloned folder:

```
$ cd prometheus
```

2. Copy **node_exporter** and **nvidia_exporter** to **/usr/bin/**.

```
$ sudo cp node_exporter/node_exporter /usr/bin/
$ sudo cp nvidia_exporter/nvidia_exporter /usr/bin/
```

3. Copy the **services** files to the services folder:

```
$ sudo cp services/node_exporter.service /etc/systemd/system/
$ sudo cp services/nvidia_exporter.service /etc/systemd/system/
```

4. Reload the services so that they can be run:

```
$ sudo systemctl daemon-reload
```

5. Set the permissions and group for both service files:

```
$ sudo chown prometheus:prometheus /usr/bin/node_exporter
$ sudo chmod u+x /usr/bin/node_exporter
$ sudo chown prometheus:prometheus /usr/bin/nvidia_exporter
$ sudo chmod u+x /usr/bin/nvidia_exporter
```

6. Start both services:

```
$ sudo systemctl start node_exporter && sudo systemctl enable node_exporter
```

7. Set both services to start automatically when the server is booted up:

```
$ sudo systemctl start nvidia_exporter && sudo systemctl enable nvidia_exporter
```

8. Verify that the server's status is **active (running)**:

```
$ sudo systemctl status node_exporter && sudo systemctl status nvidia_exporter
```

The following is the correct output:

```
$ ● node_exporter.service - Node Exporter
$    Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor␣
↪preset: disabled)
$    Active: active (running) since Wed 2019-12-11 12:28:31 IST; 1 months 5 days␣
↪ago
$  Main PID: 28378 (node_exporter)
$    CGroup: /system.slice/node_exporter.service
$
$ ● nvidia_exporter.service - Nvidia Exporter
$    Loaded: loaded (/etc/systemd/system/nvidia_exporter.service; enabled; vendor␣
↪preset: disabled)
$    Active: active (running) since Wed 2020-01-22 13:40:11 IST; 31min ago
$  Main PID: 1886 (nvidia_exporter)
```

(continues on next page)

(continued from previous page)

```
$    CGroup: /system.slice/nvidia_exporter.service
$            └─1886 /usr/bin/nvidia_exporter
```

#### 2.2.1.5 Installing the Process Exporter

After installing the **node_exporter** and **Nvidia_exporter** you must install the **process_exporter**.

**To install the process_exporter:**

1. Do one of the following:
   - For **CentOS**, run `sudo rpm -i process_exporter/process-exporter_0.5.0_linux_amd64.rpm`.
   - For **Ubuntu**, run `sudo dpkg -i process_exporter/process-exporter_0.6.0_linux_amd64.deb`.
2. Verify that the process_exporter is running:

```
$ sudo systemctl status process-exporter
```

3. Set the process_exporter to start automatically when the server is booted up:

```
$ sudo systemctl enable process-exporter
```

#### 2.2.1.6 Opening the Firewall Ports

After installing the **process_exporter** you must open the firewall ports for the following services:

- **node_exporter** - port: 9100
- **nvidia_exporter** - port: 9445
- **process-exporter** - port: 9256

**Note:** This procedure is only relevant if your firwall is running.

**To open the firewall ports:**

1. Run the following command:

```
$ sudo firewall-cmd --zone=public --add-port=<PORT NUMBER>/tcp --permanent
```

2. Reload the firewall:

```
$ sudo firewall-cmd --reload
```

3. Verify that the changes have taken effect.

### 2.2.2 Installing Prometheus Using Binary Packages

The **Installing Prometheus Using Binary Packages** guide includes the following sections:



#### 2.2.2.1 Overview

Prometheus is an application used for event monitoring and alerting.

#### 2.2.2.2 Installing Prometheus

You must install Prometheus before installing the Dashboard Data Collector.

**To install Prometheus:**

1. Verify the following:
   1. That you have **sudo** access to your Linux server.
   2. That your server has access to the internet (for downloading the Prometheus binary package).
   3. That your firewall rules are opened for accessing Prometheus Port 9090.
2. Navigate to the Prometheus Download page and download the **prometheus-2.32.0-rc.1.linux-amd64.tar.gz** package.
3. Do the following:
   1. Download the source using the `curl` command:

      ```
      $ curl -LO url -LO https://github.com/prometheus/prometheus/releases/download/
      ↪v2.22.0/prometheus-2.22.0.linux-amd64.tar.gz
      ```

   2. Extract the file contents:

      ```
      $ tar -xvf prometheus-2.22.0.linux-amd64.tar.gz
      ```

   3. Rename the extracted folder **prometheus-files**:

      ```
      $ mv prometheus-2.22.0.linux-amd64 prometheus-files
      ```

4. Create a Prometheus user:

   ```
   $ sudo useradd --no-create-home --shell /bin/false prometheus
   ```

5. Create your required directories:

```
$ sudo mkdir /etc/prometheus
$ sudo mkdir /var/lib/prometheus
```

6. Set the Prometheus user as the owner of your required directories:

```
$ sudo chown prometheus:prometheus /etc/prometheus
$ sudo chown prometheus:prometheus /var/lib/prometheus
```

7. Copy the Prometheus and Promtool binary packages from the **prometheus-files** folder to **/usr/local/bin**:

```
$ sudo cp prometheus-files/prometheus /usr/local/bin/
$ sudo cp prometheus-files/promtool /usr/local/bin/
```

8. Change the ownership to the prometheus user:

```
$ sudo chown prometheus:prometheus /usr/local/bin/prometheus
$ sudo chown prometheus:prometheus /usr/local/bin/promtool
```

9. Move the **consoles** and **consoles_libraries** directories from **prometheus-files** folder to **/etc/prometheus** folder:

```
$ sudo cp -r prometheus-files/consoles /etc/prometheus
    $ sudo cp -r prometheus-files/console_libraries /etc/prometheus
```

10. Change the ownership to the prometheus user:

```
$ sudo chown -R prometheus:prometheus /etc/prometheus/consoles
$ sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
```

For more information on installing the Dashboard Data Collector, see Installing the Dashboard Data Collector.

Back to *Installing Prometheus Using Binary Packages*

### 2.2.2.3 Configuring Your Prometheus Settings

After installing Prometheus you must configure your Prometheus settings. You must perform all Prometheus configurations in the **/etc/prometheus/prometheus.yml** file.

**To configure your Prometheus settings:**

1. Create your **prometheus.yml** file:

```
$ sudo vi /etc/prometheus/prometheus.yml
```

2. Copy the contents below into your prometheus.yml file:

```
$ #node_exporter port : 9100
$ #nvidia_exporter port: 9445
$ #process-exporter port: 9256
$
$ global:
$   scrape_interval: 10s
$
$ scrape_configs:
$   - job_name: 'prometheus'
$     scrape_interval: 5s
$     static_configs:
$       - targets:
```

(continues on next page)

(continued from previous page)

```
$          - <prometheus server IP>:9090
$   - job_name: 'processes'
$     scrape_interval: 5s
$     static_configs:
$       - targets:
$         - <process exporters iP>:9256
$         - <another process exporters iP>:9256
$   - job_name: 'nvidia'
$     scrape_interval: 5s
$     static_configs:
$       - targets:
$         - <nvidia exporter IP>:9445
$         - <another nvidia exporter IP>:9445
$   - job_name: 'nodes'
$     scrape_interval: 5s
$     static_configs:
$       - targets:
$         - <node exporter IP>:9100
$         - <another node exporter IP>:9100
```

3. Change the ownership of the file to the prometheus user:

```
$ sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml
```

Back to *Installing Prometheus Using Binary Packages*

### 2.2.2.4 Configuring Your Prometheus Service File

After configuring your Prometheus settings you must configure your Prometheus service file.

**To configure your Prometheus service file**:

1. Create your **prometheus.yml** file:

```
$ sudo vi /etc/systemd/system/prometheus.service
```

2. Copy the contents below into your prometheus service file:

```
$ [Unit]
$ Description=Prometheus
$ Wants=network-online.target
$ After=network-online.target
$
$ [Service]
$ User=prometheus
$ Group=prometheus
$ Type=simple
$ ExecStart=/usr/local/bin/prometheus \
$     --config.file /etc/prometheus/prometheus.yml \
$     --storage.tsdb.path /var/lib/prometheus/ \
$     --web.console.templates=/etc/prometheus/consoles \
$     --web.console.libraries=/etc/prometheus/console_libraries
$
$ [Install]
$ WantedBy=multi-user.target
```

3. Register the prometheus service by reloading the **systemd** service:

```
$ sudo systemctl daemon-reload
```

4. Start the prometheus service:

```
$ sudo systemctl start prometheus
```

5. Check the status of the prometheus service:

```
$ sudo systemctl status prometheus
```

If the status is `active (running)`, you have configured your Prometheus service file correctly.

Back to *Installing Prometheus Using Binary Packages*

### 2.2.2.5 Accessing the Prometheus User Interface

After configuring your prometheus service file, you can access the Prometheus user interface.

You can access the Prometheus user interface by running the following command:

```
$ http://<prometheus-ip>:9090/graph
```

The Prometheus user interface is displayed.

From the **Query** tab you can query metrics, as shown below:

Back to *Installing Prometheus Using Binary Packages*

## 2.2.3 Installing the Dashboard Data Collector

### 2.2.3.1 Installing the Dashboard Data Collector

After accessing the Prometheus user interface, you can install the **Dashboard Data Collector**. You must install the Dashboard Data Collector to enable the Dashboard in Studio.

**Note:** Before installing the Dashboard Data collector, verify that Prometheus has been installed and configured for the cluster.

How to install Prometheus from tarball - **Comment - this needs to be its own page.**

**To install the Dashboard Data Collector:**

1. Store the Data Collector Package obtained from SQream Artifactory.

2. Extract and rename the package:

```
$ tar -xvf dashboard-data-collector-0.5.2.tar.gz
$ mv package dashboard-data-collector
```

3. Change your directory to the location of the package folder:

```
$ cd dashboard-data-collector
```

4. Set up the data collection by modifying the SQream and Data Collector IPs, ports, user name, and password according to the cluster:

```
$ npm run setup -- \
$         --host=127.0.0.1 \
$         --port=3108 \
$         --database=master \
$         --is-cluster=true \
$         --service=sqream \
$         --dashboard-user=sqream \
$         --dashboard-password=sqream \
$         --prometheus-url=http://127.0.0.1:9090/api/v1/query
```

5. Debug the Data Collector: (**Comment** - *using the npm project manager*).

```
$ npm start
```

A json file is generated in the log, as shown below:

```
$ {
$   "machines": [
$     {
$       "machineId": "dd4af489615",
$       "name": "Server 0",
$       "location": "192.168.4.94",
$       "totalMemory": 31.19140625,
$       "gpus": [
$         {
$           "gpuId": "GPU-b17575ec-eeba-3e0e-99cd-963967e5ee3f",
$           "machineId": "dd4af489615",
$           "name": "GPU 0",
$           "totalMemory": 3.9453125
$         }
$       ],
$       "workers": [
$         {
$           "workerId": "sqream_01",
$           "gpuId": "",
$           "name": "sqream_01"
$         }
$       ],
$       "storageWrite": 0,
$       "storageRead": 0,
$       "freeStorage": 0
$     },
$     {
$       "machineId": "704ec607174",
$       "name": "Server 1",
$       "location": "192.168.4.95",
$       "totalMemory": 31.19140625,
```

(continues on next page)

(continued from previous page)

```
$       "gpus": [
$         {
$           "gpuId": "GPU-8777c14f-7611-517a-e9c7-f42eeb21700b",
$           "machineId": "704ec607174",
$           "name": "GPU 0",
$           "totalMemory": 3.9453125
$         }
$       ],
$       "workers": [
$         {
$           "workerId": "sqream_02",
$           "gpuId": "",
$           "name": "sqream_02"
$         }
$       ],
$       "storageWrite": 0,
$       "storageRead": 0,
$       "freeStorage": 0
$     }
$   ],
$   "clusterStatus": true,
$   "storageStatus": {
$     "dataStorage": 49.9755859375,
$     "totalDiskUsage": 52.49829018075231,
$     "storageDetails": {
$       "data": 0,
$       "freeData": 23.7392578125,
$       "tempData": 0,
$       "deletedData": 0,
$       "other": 26.236328125
$     },
$     "avgThroughput": {
$       "read": 0,
$       "write": 0
$     },
$     "location": "/"
$   },
$   "queues": [
$     {
$       "queueId": "sqream",
$       "name": "sqream",
$       "workerIds": [
$         "sqream_01",
$         "sqream_02"
$       ]
$     }
$   ],
$   "queries": [],
$   "collected": true,
$   "lastCollect": "2021-11-17T12:46:31.601Z"
$ }
```

**Note:** Verify that all machines and workers are correctly registered.

6. Press **CTRL + C** to stop `npm start` (**Comment** - *It may be better to refer to it as the npm project manager*).

7. Start the Data Collector with the `pm2` service:

```
$ pm2 start ./index.js --name=dashboard-data-collector
```

8. Add the following parameter to the SQream Studio setup defined in *Step 4* in **Installing Studio** below.

```
--data-collector-url=http://127.0.0.1:8100/api/dashboard/data
```

Back to *Installing Studio on a Stand-Alone Server*

### 2.2.4 Installing Studio on a Stand-Alone Server

The **Installing Studio on a Stand-Alone Server** guide describes how to install SQream Studio on a stand-alone server. A stand-alone server is a server that does not run SQream based on binary files, Docker, or Kubernetes.

The Installing Studio on a Stand-Alone Server guide includes the following sections:



#### 2.2.4.1 Installing NodeJS Version 12 on the Server

Before installing Studio you must install NodeJS version 12 on the server.

**To install NodeJS version 12 on the server:**

1. Check if a version of NodeJS older than version *12.<x.x>* has been installed on the target server.

```
$ node -v
```

The following is the output if a version of NodeJS has already been installed on the target server:

```
bash: /usr/bin/node: No such file or directory
```

2. If a version of NodeJS older than *12.<x.x>* has been installed, remove it as follows:

   - On CentOS:

```
$ sudo yum remove -y nodejs
```

   - On Ubuntu:

```
$ sudo apt remove -y nodejs
```

3. If you have not installed NodeJS version 12, run the following commands:

   - On CentOS:

   ```
   $ curl -sL https://rpm.nodesource.com/setup_12.x | sudo bash -
   $ sudo yum clean all && sudo yum makecache fast
   $ sudo yum install -y nodejs
   ```

   - On Ubuntu:

   ```
   $ curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
   $ sudo apt-get install -y nodejs
   ```

   The following output is displayed if your installation has completed successfully:

```
Transaction Summary
==============================================================================================
Install  1 Package

Total download size: 22 M
Installed size: 67 M
Downloading packages:
warning: /var/cache/yum/x86_64/7/nodesource/packages/nodejs-12.22.1-
↪1nodesource.x86_64.rpm: Header V4 RSA/SHA512 Signature, key ID 34fa74dd:␣
↪NOKEY
Public key for nodejs-12.22.1-1nodesource.x86_64.rpm is not installed
nodejs-12.22.1-1nodesource.x86_64.rpm                                       ␣
↪                          |  22 MB  00:00:02
Retrieving key from file:///etc/pki/rpm-gpg/NODESOURCE-GPG-SIGNING-KEY-EL
Importing GPG key 0x34FA74DD:
 Userid     : "NodeSource <gpg-rpm@nodesource.com>"
 Fingerprint: 2e55 207a 95d9 944b 0cc9 3261 5ddb e8d4 34fa 74dd
 Package    : nodesource-release-el7-1.noarch (installed)
 From       : /etc/pki/rpm-gpg/NODESOURCE-GPG-SIGNING-KEY-EL
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Warning: RPMDB altered outside of yum.
  Installing : 2:nodejs-12.22.1-1nodesource.x86_64                          ␣
↪                                            1/1
  Verifying  : 2:nodejs-12.22.1-1nodesource.x86_64                          ␣
↪                                            1/1

Installed:
  nodejs.x86_64 2:12.22.1-1nodesource

Complete!
```

4. Confirm the Node version.

```
$ node -v
```

   The following is an example of the correct output:

```
v12.22.1
```

5. Install Prometheus using binary packages.

   For more information on installing Prometheus using binary packages, see *Installing Prometheus Using Binary Packages*.

Back to *Installing Studio on a Stand-Alone Server*

#### 2.2.4.2 Installing Studio

After installing the Dashboard Data Collector, you can install Studio.

**To install Studio:**

1. Copy the SQream Studio package from SQream Artifactory into the target server. For access to the Sqream Studio package, contact SQream Support.

2. Extract the package:

```
$ tar -xvf sqream-acceleration-studio-<version number>.x86_64.tar.gz
```

3. Navigate to the new package folder.

```
$ cd sqream-admin
```

4. Build the configuration file to set up Sqream Studio. You can use IP address **127.0.0.1** on a single server.

```
$ npm run setup -- -y --host=<SQreamD IP> --port=3108 --data-collector-url=http://
↪<data collector IP address>:8100/api/dashboard/data
```

   The above command creates the **sqream-admin-config.json** configuration file in the **sqream-admin** folder and shows the following output:

```
Config generated successfully. Run `npm start` to start the app.
```

   For more information about the available set-up arguments, see Set-Up Arguments.

5. To access Studio over a secure connection, in your configuration file do the following:

   1. Change your `port` value to **3109**.

   2. Change your `ssl` flag value to **true**.

      The following is an example of the correctly modified configuration file:

```
{
  "debugSqream": false,
  "webHost": "localhost",
  "webPort": 8080,
  "webSslPort": 8443,
  "logsDirectory": "",
  "clusterType": "standalone",
  "dataCollectorUrl": "",
  "connections": [
    {
```

(continues on next page)

(continued from previous page)

```
        "host": "127.0.0.1",
        "port":3109,
        "isCluster": true,
        "name": "default",
        "service": "sqream",
        "ssl":true,
        "networkTimeout": 60000,
        "connectionTimeout": 3000
      }
    ]
}
```

5. If you have installed Studio on a server where SQream is already installed, move the **sqream-admin-config.json** file to **/etc/sqream/**:

   ```
   $ mv sqream-admin-config.json /etc/sqream
   ```

Back to *Installing Studio on a Stand-Alone Server*

#### 2.2.4.3 Starting Studio Manually

You can start Studio manually by running the following command:

```
$ cd /home/sqream/sqream-admin
$ NODE_ENV=production pm2 start ./server/build/main.js --name=sqream-studio -- start
```

The following output is displayed:

```
[PM2] Starting /home/sqream/sqream-admin/server/build/main.js in fork_mode (1␣
↪instance)
[PM2] Done.
┌─────┬──────────────────┬─────────────┬─────────┬─────────┬──────────┬────────┬──────┬──────────┬──────────┬──────────┐
│ id  │ name             │ namespace   │ version │ mode    │ pid      │ uptime │ ↺    │ ␣
↪│ status    │ cpu      │ mem      │ user     │ watching │
├─────┼──────────────────┼─────────────┼─────────┼─────────┼──────────┼────────┼──────┼──────────┼──────────┼──────────┤
│ 0   │ sqream-studio    │ default     │ 0.1.0   │ fork    │ 11540    │ 0s     │ 0    │ ␣
↪│ online    │ 0%       │ 15.6mb   │ sqream   │ disabled │
└─────┴──────────────────┴─────────────┴─────────┴─────────┴──────────┴────────┴──────┴──────────┴──────────┴──────────┘
```

#### 2.2.4.4 Starting Studio as a Service

Sqream uses the **Process Manager (PM2)** to maintain Studio.

**To start Studio as a service:**

1. Run the following command:

   ```
   $ sudo npm install -g pm2
   ```

2. Verify that the PM2 has been installed successfully.

   ```
   $ pm2 list
   ```

   The following is the output:

```
┌────┬────────────────────┬───────────┬─────────┬─────────┬──────────┬────────┬␣
↪──┬──────────┬──────────┬──────────┬──────────┬──────────┐
│ id │ name               │ namespace │ version │ mode    │ pid      │ uptime │␣
↪↺ │ status   │ cpu      │ mem      │ user     │ watching │
├────┼────────────────────┼───────────┼─────────┼─────────┼──────────┼────────┼␣
↪──┼──────────┼──────────┼──────────┼──────────┼──────────┤
│ 0  │ sqream-studio      │ default   │ 0.1.0   │ fork    │ 11540    │ 2m     │␣
↪0 │ online   │ 0%       │ 31.5mb   │ sqream   │ disabled │
└────┴────────────────────┴───────────┴─────────┴─────────┴──────────┴────────┴␣
↪──┴──────────┴──────────┴──────────┴──────────┴──────────┘
```

2. Start the service with PM2:

   - If the **sqream-admin-config.json** file is located in **/etc/sqream/**, run the following command:

     ```
     $ cd /home/sqream/sqream-admin
     $ NODE_ENV=production pm2 start ./server/build/main.js --name=sqream-studio --
     ↪ start --config-location=/etc/sqream/sqream-admin-config.json
     ```

   - If the **sqream-admin-config.json** file is not located in **/etc/sqream/**, run the following command:

     ```
     $ cd /home/sqream/sqream-admin
     $ NODE_ENV=production pm2 start ./server/build/main.js --name=sqream-studio --
     ↪ start
     ```

3. Verify that Studio is running.

   ```
   $ netstat -nltp
   ```

4. Verify that SQream_studio is listening on port 8080, as shown below:

   ```
   (Not all processes could be identified, non-owned process info
    will not be shown, you would have to be root to see it all.)
   Active Internet connections (only servers)
   Proto Recv-Q Send-Q Local Address           Foreign Address         State       ␣
   ↪PID/Program name
   tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
   tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      -
   tcp6       0      0 :::8080                 :::*                    LISTEN      ␣
   ↪11540/sqream-studio
   tcp6       0      0 :::22                   :::*                    LISTEN      -
   tcp6       0      0 ::1:25                  :::*                    LISTEN      -
   ```

5. Verify the following:

   1. That you can access Studio from your browser (`http://<IP_Address>:8080`).
   2. That you can log in to SQream.

6. Save the configuration to run on boot.

   ```
   $ pm2 startup
   ```

   The following is an example of the output:

   ```
   $ sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -
   ↪u sqream --hp /home/sqream
   ```

7. Copy and paste the output above and run it.

8. Save the configuration.

```
$ pm2 save
```

Back to *Installing Studio on a Stand-Alone Server*

#### 2.2.4.5 Accessing Studio

The Studio page is available on port 8080: `http://<server ip>:8080`.

If port 8080 is blocked by the server firewall, you can unblock it by running the following command:

```
$ firewall-cmd --zone=public --add-port=8080/tcp --permanent
$ firewall-cmd --reload
```

Back to *Installing Studio on a Stand-Alone Server*

#### 2.2.4.6 Maintaining Studio with the Process Manager (PM2)

Sqream uses the **Process Manager (PM2)** to maintain Studio.

You can use PM2 to do one of the following:

- To check the PM2 service status: `pm2 list`
- To restart the PM2 service: `pm2 reload sqream-studio`
- To see the PM2 service logs: `pm2 logs sqream-studio`

Back to *Installing Studio on a Stand-Alone Server*

#### 2.2.4.7 Upgrading Studio

To upgrade Studio you need to stop the version that you currently have.

**To stop the current version of Studio:**

1. List the process name:

   ```
   $ pm2 list
   ```

   The process name is displayed.

   ```
   <process name>
   ```

2. Run the following command with the process name:

   ```
   $ pm2 stop <process name>
   ```

3. If only one process is running, run the following command:

   ```
   $ pm2 stop all
   ```

4. Change the name of the current **sqream-admin** folder to the old version.

```
$ mv sqream-admin sqream-admin-<old_version>
```

5. Extract the new Studio version.

```
$ tar -xf sqream-acceleration-studio-<version>tar.gz
```

6. Rebuild the configuration file. You can use IP address **127.0.0.1** on a single server.

```
$ npm run setup -- -y --host=<SQreamD IP> --port=3108
```

   The above command creates the **sqream-admin-config.json** configuration file in the **sqream_admin** folder.

7. Copy the **sqream-admin-config.json** configuration file to **/etc/sqream/** to overwrite the old configuration file.
8. Start PM2.

```
$ pm2 start all
```

Back to *Installing Studio on a Stand-Alone Server*

#### 2.2.4.8 Installing Studio in a Docker Container

This guide explains how to install SQream Studio in a Docker container and includes the following sections:



##### 2.2.4.8.1 Installing Studio

If you have already installed Docker, you can install Studio in a Docker container.

**To install Studio:**

1. Copy the downloaded image onto the target server.
2. Load the Docker image.

```
$ docker load -i <docker_image_file>
```

3. If the downloaded image is called **sqream-acceleration-studio-5.1.3.x86_64.docker18.0.3.tar,** run the following command:

```
$ docker load -i sqream-acceleration-studio-5.1.3.x86_64.docker18.0.3.tar
```

4. Start the Docker container.

```
$ docker run -d --restart=unless-stopped -p <external port>:8080 -e␣
↪runtime=docker -e SQREAM_K8S_PICKER=<SQream host IP or VIP> -e SQREAM_PICKER_
↪PORT=<SQream picker port> -e SQREAM_DATABASE_NAME=<SQream database name> -e␣
↪SQREAM_ADMIN_UI_PORT=8080 --name=sqream-admin-ui <docker_image_name>
```

The following is an example of the command above:

```
$ docker run -d --name sqream-studio  -p 8080:8080 -e runtime=docker -e SQREAM_
↪K8S_PICKER=192.168.0.183 -e SQREAM_PICKER_PORT=3108 -e SQREAM_DATABASE_
↪NAME=master -e SQREAM_ADMIN_UI_PORT=8080 sqream-acceleration-studio:5.1.3
```

Back to *Installing Studio in a Docker Container*

#### 2.2.4.8.2 Accessing Studio

You can access Studio from Port 8080: `http://<server ip>:8080`.

If you want to use Studio over a secure connection (https), you must use the parameter values shown in the following table:

| Parameter | Default Value | Description |
|---|---|---|
| `--web-ssl-port` | 8443 | |
| `--web-ssl-key-path` | None | The path of SSL key PEM file for enabling https. Leave empty to disable. |
| `--web-ssl-cert-path` | None | The path of SSL certificate PEM file for enabling https. Leave empty to disable. |

You can configure the above parameters using the following syntax:

```
$ npm run setup -- -y --host=127.0.0.1 --port=3108
```

Back to *Installing Studio in a Docker Container*

#### 2.2.4.8.3 Using Docker Container Commands

When installing Studio in Docker, you can run the following commands:

- View Docker container logs:

```
$ docker logs -f sqream-admin-ui
```

- Restart the Docker container:

```
$ docker restart sqream-admin-ui
```

- Kill the Docker container:

```
$ docker rm -f sqream-admin-ui
```

Back to *Installing Studio in a Docker Container*

#### 2.2.4.8.4 Setting Up Argument Configurations

When creating the **sqream-admin-config.json** configuration file, you can add `-y` to create the configuration file in non-interactive mode. Configuration files created in non-interactive mode use all the parameter defaults not provided in the command.

The following table shows the available arguments:

| Parameter | Default Value | Description |
|---|---|---|
| `--web--host` | 8443 | |
| `--web-port` | 8080 | |
| `--web-ssl-port` | 8443 | |
| `--web-ssl-key-path` | None | The path of the SSL Key PEM file for enabling https. Leave empty to disable. |
| `--web-ssl-cert-path` | None | The path of the SSL Certificate PEM file for enabling https. Leave empty to disable. |
| `--debug-sqream (flag)` | false | |
| `--host` | 127.0.0.1 | |
| `--port` | 3108 | |
| `is-cluster (flag)` | true | |
| `--service` | sqream | |
| `--ssl (flag)` | false | Enables the SQream SSL connection. |
| `--name` | default | |
| `--data-collector-url` | localhost:8100/api/dashboard/data | Enables the Dashboard. Leaving this blank disables the Dashboard. Using a mock URL uses mock data. |
| `--cluster-type` | standalone (`standalone` or `k8s`) | |
| `--config-location` | ./sqream-admin-config.json | |
| `--network-timeout` | 60000 (60 seconds) | |
| `--access-key` | None | If defined, UI access is blocked unless `?ui-access=<access key>` is included in the URL. |

Back to *Installing Studio in a Docker Container*

Back to *Installing Studio on a Stand-Alone Server*

## 2.2.5 Installing an NGINX Proxy Over a Secure Connection

Configuring your NGINX server to use a strong encryption for client connections provides you with secure servers requests, preventing outside parties from gaining access to your traffic.

The **Installing an NGINX Proxy Over a Secure Connection** page describes the following:



### 2.2.5.1 Overview

The Node.js platform that SQream uses with our Studio user interface is susceptible to web exposure. This page describes how to implement HTTPS access on your proxy server to establish a secure connection.

**TLS (Transport Layer Security)**, and its predecessor **SSL (Secure Sockets Layer)**, are standard web protocols used for wrapping normal traffic in a protected, encrypted wrapper. This technology prevents the interception of server-client traffic. It also uses a certificate system for helping users verify the identity of sites they visit. The **Installing an NGINX Proxy Over a Secure Connection** guide describes how to set up a self-signed SSL certificate for use with an NGINX web server on a CentOS 7 server.

**Note:** A self-signed certificate encrypts communication between your server and any clients. However, because it is not signed by trusted certificate authorities included with web browsers, you cannot use the certificate to automatically validate the identity of your server.

A self-signed certificate may be appropriate if your domain name is not associated with your server, and in cases where your encrypted web interface is not user-facing. If you do have a domain name, using a CA-signed certificate is generally preferrable.

For more information on setting up a free trusted certificate, see How To Secure Nginx with Let's Encrypt on CentOS 7.

#### 2.2.5.2 Prerequisites

The following prerequisites are required for installing an NGINX proxy over a secure connection:

- Super user privileges
- A domain name to create a certificate for

#### 2.2.5.3 Installing NGINX and Adjusting the Firewall

After verifying that you have the above preriquisites, you must verify that the NGINX web server has been installed on your machine.

Though NGINX is not available in the default CentOS repositories, it is available from the **EPEL (Extra Packages for Enterprise Linux)** repository.

**To install NGINX and adjust the firewall:**

1. Enable the EPEL repository to enable server access to the NGINX package:

   ```
   $ sudo yum install epel-release
   ```

2. Install NGINX:

   ```
   $ sudo yum install nginx
   ```

3. Start the NGINX service:

   ```
   $ sudo systemctl start nginx
   ```

4. Verify that the service is running:

   ```
   $ systemctl status nginx
   ```

   The following is an example of the correct output:

   ```
   Output● nginx.service - The nginx HTTP and reverse proxy server
      Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor␣
   ↪preset: disabled)
      Active: active (running) since Fri 2017-01-06 17:27:50 UTC; 28s ago

   . . .

   Jan 06 17:27:50 centos-512mb-nyc3-01 systemd[1]: Started The nginx HTTP and␣
   ↪reverse proxy server.
   ```

5. Enable NGINX to start when your server boots up:

   ```
   $ sudo systemctl enable nginx
   ```

6. Verify that access to **ports 80 and 443** are not blocked by a firewall.

7. Do one of the following:

   - If you are not using a firewall, skip to *Creating Your SSL Certificate*.

- If you have a running firewall, open ports 80 and 443:

```
$ sudo firewall-cmd --add-service=http
$ sudo firewall-cmd --add-service=https
$ sudo firewall-cmd --runtime-to-permanent
```

8. If you have a running **iptables firewall**, for a basic rule set, add HTTP and HTTPS access:

```
$ sudo iptables -I INPUT -p tcp -m tcp --dport 80 -j ACCEPT
$ sudo iptables -I INPUT -p tcp -m tcp --dport 443 -j ACCEPT
```

**Note:** The commands in Step 8 above are highly dependent on your current rule set.

9. Verify that you can access the default NGINX page from a web browser.

#### 2.2.5.4 Creating Your SSL Certificate

After installing NGINX and adjusting your firewall, you must create your SSL certificate.

TLS/SSL combines public certificates with private keys. The SSL key, kept private on your server, is used to encrypt content sent to clients, while the SSL certificate is publicly shared with anyone requesting content. In addition, the SSL certificate can be used to decrypt the content signed by the associated SSL key. Your public certificate is located in the **/etc/ssl/certs** directory on your server.

This section describes how to create your **/etc/ssl/private directory**, used for storing your private key file. Because the privacy of this key is essential for security, the permissions must be locked down to prevent unauthorized access:

**To create your SSL certificate:**

1. Set the following permissions to **private**:

```
$ sudo mkdir /etc/ssl/private
$ sudo chmod 700 /etc/ssl/private
```

2. Create a self-signed key and certificate pair with OpenSSL with the following command:

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/
↪private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
```

The following list describes the elements in the command above:

- **openssl** - The basic command line tool used for creating and managing OpenSSL certificates, keys, and other files.

- **req** - A subcommand for using the X.509 **Certificate Signing Request (CSR)** management. A public key infrastructure standard, SSL and TLS adhere X.509 key and certificate management regulations.

- **-x509** - Used for modifying the previous subcommand by overriding the default functionality of generating a certificate signing request with making a self-signed certificate.

- **-nodes** - Sets **OpenSSL** to skip the option of securing our certificate with a passphrase, letting NGINX read the file without user intervention when the server is activated. If you don't use **-nodes** you must enter your passphrase after every restart.
- **-days 365** - Sets the certificate's validation duration to one year.
- **-newkey rsa:2048** - Simultaneously generates a new certificate and new key. Because the key required to sign the certificate was not created in the previous step, it must be created along with the certificate. The **rsa:2048** generates an RSA 2048 bits long.
- **-keyout** - Determines the location of the generated private key file.
- **-out** - Determines the location of the certificate.

After creating a self-signed key and certificate pair with OpenSSL, a series of prompts about your server is presented to correctly embed the information you provided in the certificate.

3. Provide the information requested by the prompts.

   The most important piece of information is the **Common Name**, which is either the server **FQDN** or **your** name. You must enter the domain name associated with your server or your server's public IP address.

   The following is an example of a filled out set of prompts:

```
OutputCountry Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:New York City
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Bouncy Castles, Inc.
Organizational Unit Name (eg, section) []:Ministry of Water Slides
Common Name (e.g. server FQDN or YOUR name) []:server_IP_address
Email Address []:admin@your_domain.com
```

   Both files you create are stored in their own subdirectories of the **/etc/ssl** directory.

   Although SQream uses OpenSSL, in addition we recommend creating a strong **Diffie-Hellman** group, used for negotiating **Perfect Forward Secrecy** with clients.

4. Create a strong Diffie-Hellman group:

```
$ sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

   Creating a Diffie-Hellman group takes a few minutes, which is stored as the **dhparam.pem** file in the **/etc/ssl/certs** directory. This file can use in the configuration.

#### 2.2.5.5 Configuring NGINX to use SSL

After creating your SSL certificate, you must configure NGINX to use SSL.

The default CentOS NGINX configuration is fairly unstructured, with the default HTTP server block located in the main configuration file. NGINX checks for files ending in **.conf** in the **/etc/nginx/conf.d** directory for additional configuration.

SQream creates a new file in the **/etc/nginx/conf.d** directory to configure a server block. This block serves content using the certificate files we generated. In addition, the default server block can be optionally configured to redirect HTTP requests to HTTPS.

**Note:** The example on this page uses the IP address **127.0.0.1**, which you should replace with your machine's IP address.

**To configure NGINX to use SSL:**

1. Create and open a file called **ssl.conf** in the **/etc/nginx/conf.d** directory:

```
$ sudo vi /etc/nginx/conf.d/ssl.conf
```

2. In the file you created in Step 1 above, open a server block:

   1. Listen to **port 443**, which is the TLS/SSL default port.

   2. Set the `server_name` to the server's domain name or IP address you used as the Common Name when generating your certificate.

   3. Use the `ssl_certificate`, `ssl_certificate_key`, and `ssl_dhparam` directives to set the location of the SSL files you generated, as shown in the **/etc/nginx/conf.d/ssl.conf** file below:

```
    upstream ui {
        server 127.0.0.1:8080;
    }
server {
    listen 443 http2 ssl;
    listen [::]:443 http2 ssl;

    server_name nginx.sq.l;

    ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;

root /usr/share/nginx/html;

#    location / {
#    }

  location / {
        proxy_pass http://ui;
        proxy_set_header           X-Forwarded-Proto https;
        proxy_set_header           X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header           X-Real-IP       $remote_addr;
        proxy_set_header           Host $host;
                add_header                 Front-End-Https   on;
        add_header                 X-Cache-Status $upstream_cache_status;
```

(continues on next page)

(continued from previous page)

```
        proxy_cache                 off;
        proxy_cache_revalidate      off;
        proxy_cache_min_uses        1;
        proxy_cache_valid           200 302 1h;
        proxy_cache_valid           404 3s;
        proxy_cache_use_stale       error timeout invalid_header updating http_500␣
↪http_502 http_503 http_504;
        proxy_no_cache              $cookie_nocache $arg_nocache $arg_comment
↪$http_pragma $http_authorization;
        proxy_redirect              default;
        proxy_max_temp_file_size    0;
        proxy_connect_timeout       90;
        proxy_send_timeout          90;
        proxy_read_timeout          90;
        proxy_buffer_size           4k;
        proxy_buffering             on;
        proxy_buffers               4 32k;
        proxy_busy_buffers_size     64k;
        proxy_temp_file_write_size  64k;
        proxy_intercept_errors      on;

        proxy_set_header            Upgrade $http_upgrade;
        proxy_set_header            Connection "upgrade";
    }

    error_page 404 /404.html;
    location = /404.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

4. Open and modify the **nginx.conf** file located in the **/etc/nginx/conf.d** directory as follows:

```
$ sudo vi /etc/nginx/conf.d/nginx.conf
```

```
server {
    listen       80;
    listen       [::]:80;
    server_name  _;
    root         /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    error_page 404 /404.html;
    location = /404.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

#### 2.2.5.6 Redirecting Studio Access from HTTP to HTTPS

After configuring NGINX to use SSL, you must redirect Studio access from HTTP to HTTPS.

According to your current configuration, NGINX responds with encrypted content for requests on port 443, but with **unencrypted** content for requests on **port 80**. This means that our site offers encryption, but does not enforce its usage. This may be fine for some use cases, but it is usually better to require encryption. This is especially important when confidential data like passwords may be transferred between the browser and the server.

The default NGINX configuration file allows us to easily add directives to the default port 80 server block by adding files in the /etc/nginx/default.d directory.

**To create a redirect from HTTP to HTTPS:**

1. Create a new file called **ssl-redirect.conf** and open it for editing:

```
$ sudo vi /etc/nginx/default.d/ssl-redirect.conf
```

2. Copy and paste this line:

```
$ return 301 https://$host$request_uri:8080/;
```

#### 2.2.5.7 Activating Your NGINX Configuration

After redirecting from HTTP to HTTPs, you must restart NGINX to activate your new configuration.

**To activate your NGINX configuration:**

1. Verify that your files contain no syntax errors:

```
$ sudo nginx -t
```

   The following output is generated if your files contain no syntax errors:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

2. Restart NGINX to activate your configuration:

```
$ sudo systemctl restart nginx
```

#### 2.2.5.8 Verifying that NGINX is Running

After activating your NGINX configuration, you must verify that NGINX is running correctly.

**To verify that NGINX is running correctly:**

1. Check that the service is up and running:

```
$ systemctl status nginx
```

   The following is an example of the correct output:

```
Output● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor␣
↪preset: disabled)
   Active: active (running) since Fri 2017-01-06 17:27:50 UTC; 28s ago
```

(continues on next page)

(continued from previous page)

```
. . .

Jan 06 17:27:50 centos-512mb-nyc3-01 systemd[1]: Started The nginx HTTP and␣
↪reverse proxy server.
```

2. Run the following command:

```
$ sudo netstat -nltp |grep nginx
```

The following is an example of the correct output:

```
[sqream@dorb-pc etc]$ sudo netstat -nltp |grep nginx
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      ␣
↪15486/nginx: master
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN      ␣
↪15486/nginx: master
tcp6       0      0 :::80                   :::*                    LISTEN      ␣
↪15486/nginx: master
tcp6       0      0 :::443                  :::*                    LISTEN      ␣
↪15486/nginx: master
```

CHAPTER

THREE

# DATA INGESTION SOURCES

The **Data Ingestion Sources** provides information about the following:

## 3.1 Inserting Data Overview

The **Inserting Data Overview** page provides basic information useful when ingesting data into SQream from a variety of sources and locations, and describes the following:



### 3.1.1 Getting Started

SQream supports ingesting data using the following methods:

- Executing the `INSERT` statement using a client driver.

- Executing the `COPY FROM` statement or ingesting data from foreign tables:
  - Local filesystem and locally mounted network filesystems
  - Inserting Data using the Amazon S3 object storage service
  - Inserting Data using an HDFS data storage system

SQream supports loading files from the following formats:

- Text - CSV, TSV, and PSV
- Parquet
- ORC

For more information, see the following:

- Using the `INSERT` statement - insert
- Using client drivers - *Client drivers*
- Using the `COPY FROM` statement - copy_from

- Using the Amazon S3 object storage service - *Inserting Data Using Amazon S3*
- Using the HDFS data storage system - *Using SQream in an HDFS Environment*
- Loading data from foreign tables - *Foreign Tables*

### 3.1.2 Data Loading Considerations

The **Data Loading Considerations** section describes the following:



#### 3.1.2.1 Verifying Data and Performance after Loading

Like many RDBMSs, SQream recommends its own set of best practices for table design and query optimization. When using SQream, verify the following:

- That your data is structured as you expect (row counts, data types, formatting, content).
- That your query performance is adequate.
- That you followed the table design best practices (*Optimization and Best Practices*).
- That you've tested and verified that your applications work (such as Tableau).
- That your data types have not been not over-provisioned.

#### 3.1.2.2 File Soure Location when Loading

While you are loading data, you can use the `COPY FROM` command to let statements run on any worker. If you are running multiple nodes, verify that all nodes can see the source the same. Loading data from a local file that is only on one node and not on shared storage may cause it to fail. If required, you can also control which node a statement runs on using the Workload Manager).

For more information, see the following:

- copy_from
- *Workload Manager*

#### 3.1.2.3 Supported Load Methods

You can use the `COPY FROM` syntax to load CSV files.

**Note:** The `COPY FROM` cannot be used for loading data from Parquet and ORC files.

You can use foreign tables to load text files, Parquet, and ORC files, and to transform your data before generating a full table, as described in the following table:

| Method/File Type | Text (CSV) | Parquet | ORC | Streaming Data |
|---|---|---|---|---|
| COPY FROM | Supported | Not supported | Not supported | Not supported |
| Foreign tables | Supported | Supported | Supported | Not supported |
| INSERT | Not supported | Not supported | Not supported | Supported (Python, JDBC, Node.JS) |

For more information, see the following:

- COPY FROM
- *Foreign tables*
- INSERT

#### 3.1.2.4 Unsupported Data Types

SQream does not support certain features that are supported by other databases, such as `ARRAY`, `BLOB`, `ENUM`, and `SET`. You must convert these data types before loading them. For example, you can store `ENUM` as `TEXT`.

#### 3.1.2.5 Handing Extended Errors

While you can use foreign tables to load CSVs, the `COPY FROM` statement provides more fine-grained error handling options and extended support for non-standard CSVs with multi-character delimiters, alternate timestamp formats, and more.

For more information, see *foreign tables*.

#### 3.1.2.6 Best Practices for CSV

Text files, such as CSV, rarely conform to RFC 4180 , so you may need to make the following modifications:

- Use `OFFSET 2` for files containing header rows.
- You can capture failed rows in a log file for later analysis, or skip them. See capturing_rejected_rows for information on skipping rejected rows.
- You can modify record delimiters (new lines) using the *RECORD DELIMITER* syntax.
- If the date formats deviate from ISO 8601, refer to the copy_date_parsers section for overriding the default parsing.
- *(Optional)* You can quote fields in a CSV using double-quotes (`"`).

**Note:** You must quote any field containing a new line or another double-quote character.

- If a field is quoted, you must double quote any double quote, similar to the **string literals quoting rules**. For example, to encode `What are "birds"?`, the field should appear as `"What are ""birds""?"`. For more information, see string literals quoting rules.
- Field delimiters do not have to be a displayable ASCII character. For all supported field delimiters, see field_delimiters.

### 3.1.2.7 Best Practices for Parquet

The following list shows the best practices when inserting data from Parquet files:

- You must load Parquet files through *Foreign Tables*. Note that the destination table structure must be identical to the number of columns between the source files.
- Parquet files support **predicate pushdown**. When a query is issued over Parquet files, SQream uses row-group metadata to determine which row-groups in a file must be read for a particular query and the row indexes can narrow the search to a particular set of rows.

#### 3.1.2.7.1 Supported Types and Behavior Notes

Unlike the ORC format, the column types should match the data types exactly, as shown in the table below:

| SQream DB type → Parquet source | `BOOL` | `TINYINT` | `SMALLINT` | `INT` | `BIGINT` | `REAL` | `DOUBLE` | Text[Page 154, 1] | `DATE` | `DATETIME` |
|---|---|---|---|---|---|---|---|---|---|---|
| `BOOLEAN` | Supported | | | | | | | | | |
| `INT16` | | | Supported | | | | | | | |
| `INT32` | | | | Supported | | | | | | |
| `INT64` | | | | | Supported | | | | | |
| `FLOAT` | | | | | | Supported | | | | |
| `DOUBLE` | | | | | | | Supported | | | |
| `BYTE_ARRAY`[2] | | | | | | | | Supported | | |
| `INT96`[3] | | | | | | | | | | Supported[4] |

If a Parquet file has an unsupported type, such as `enum`, `uuid`, `time`, `json`, `bson`, `lists`, `maps`, but the table does not reference this data (i.e., the data does not appear in the SELECT query), the statement will succeed. If the table **does** reference a column, an error will be displayed explaining that the type is not supported, but the column may be omitted.

[1] Text values include `TEXT`, `VARCHAR`, and `NVARCHAR`

[2] With UTF8 annotation

[3] With `TIMESTAMP_NANOS` or `TIMESTAMP_MILLIS` annotation

[4] Any microseconds will be rounded down to milliseconds.

#### 3.1.2.8 Best Practices for ORC

The following list shows the best practices when inserting data from ORC files:

- You must load ORC files through *Foreign Tables*. Note that the destination table structure must be identical to the number of columns between the source files.
- ORC files support **predicate pushdown**. When a query is issued over ORC files, SQream uses ORC metadata to determine which stripes in a file need to be read for a particular query and the row indexes can narrow the search to a particular set of 10,000 rows.

##### 3.1.2.8.1 Type Support and Behavior Notes

You must load ORC files through foreign table. Note that the destination table structure must be identical to the number of columns between the source files.

For more information, see *Foreign Tables*.

The types should match to some extent within the same "class", as shown in the following table:

| SQream DB Type → ORC Source | BOOL | TINYINT | SMALLINT | INT | BIGINT | REAL | DOUBLE | Text[Page 154, 1] | DATE | DATETIME |
|---|---|---|---|---|---|---|---|---|---|---|
| `boolean` | Supported | Supported[5] | Supported[?] | Supported[?] | Supported[?] | | | | | |
| `tinyint` | ○[6] | Supported | Supported | Supported | Supported | | | | | |
| `smallint` | ○[?] | ○[7] | Supported | Supported | Supported | | | | | |
| `int` | ○[?] | ○[?] | ○[?] | Supported | Supported | | | | | |
| `bigint` | ○[?] | ○[?] | ○[?] | ○[?] | Supported | | | | | |
| `float` | | | | | | Supported | Supported | | | |
| `double` | | | | | | Supported | Supported | | | |
| `string` / `char` / `varchar` | | | | | | | | Supported | | |
| `date` | | | | | | | | | Supported | Supported |
| `timestamp`, `timestamp` with timezone | | | | | | | | | | Supported |

- If an ORC file has an unsupported type like `binary`, `list`, `map`, and `union`, but the data is not referenced in the table (it does not appear in the SELECT query), the statement will succeed. If the column is referenced, an error will be thrown to the user, explaining that the type is not supported, but the column may be ommited.

[5] Boolean values are cast to 0, 1
[6] Will succeed if all values are 0, 1
[7] Will succeed if all values fit the destination type

### 3.1.3 Further Reading and Migration Guides

For more information, see the following:

- copy_from
- insert
- *Foreign Tables*

## 3.2 Inserting Data from Avro

The **Inserting Data from Avro** page describes inserting data from Avro into SQream and includes the following:



### 3.2.1 Overview

**Avro** is a well-known data serialization system that relies on schemas. Due to its flexibility as an efficient data storage method, SQream supports the Avro binary data format as an alternative to JSON. Avro files are represented using the **Object Container File** format, in which the Avro schema is encoded alongside binary data. Multiple files loaded in the same transaction are serialized using the same schema. If they are not serialized using the same schema, an error message is displayed. SQream uses the **.avro** extension for ingested Avro files.

### 3.2.2 Making Avro Files Accessible to Workers

To give workers access to files every node must have the same view of the storage being used.

The following apply for Avro files to be accessible to workers:

- For files hosted on NFS, ensure that the mount is accessible from all servers.
- For HDFS, ensure that SQream servers have access to the HDFS name node with the correct **user-id**. For more information, see *Using SQream in an HDFS Environment*.
- For S3, ensure network access to the S3 endpoint. For more information, see *Inserting Data Using Amazon S3*.

For more information about restricted worker access, see *Workload Manager*.

### 3.2.3 Preparing Your Table

You can build your table structure on both local and foreign tables:



#### 3.2.3.1 Creating a Table

Before loading data, you must build the `CREATE TABLE` to correspond with the file structure of the inserted table.

The example in this section is based on the source `nba.avro` table shown below:

Table 1: nba.avro

| Name | Team | Number | Position | Age | Height | Weight | College | Salary |
|---|---|---|---|---|---|---|---|---|
| Avery Bradley | Boston Celtics | 0.0 | PG | 25.0 | 6-2 | 180.0 | Texas | 7730337.0 |
| Jae Crowder | Boston Celtics | 99.0 | SF | 25.0 | 6-6 | 235.0 | Marquette | 6796117.0 |
| John Holland | Boston Celtics | 30.0 | SG | 27.0 | 6-5 | 205.0 | Boston University | |
| R.J. Hunter | Boston Celtics | 28.0 | SG | 22.0 | 6-5 | 185.0 | Georgia State | 1148640.0 |
| Jonas Jerebko | Boston Celtics | 8.0 | PF | 29.0 | 6-10 | 231.0 | | 5000000.0 |
| Amir Johnson | Boston Celtics | 90.0 | PF | 29.0 | 6-9 | 240.0 | | 12000000.0 |
| Jordan Mickey | Boston Celtics | 55.0 | PF | 21.0 | 6-8 | 235.0 | LSU | 1170960.0 |
| Kelly Olynyk | Boston Celtics | 41.0 | C | 25.0 | 7-0 | 238.0 | Gonzaga | 2165160.0 |
| Terry Rozier | Boston Celtics | 12.0 | PG | 22.0 | 6-2 | 190.0 | Louisville | 1824360.0 |

The following example shows the correct file structure used to create the `CREATE TABLE` statement based on the **nba.avro** table:

```
CREATE TABLE ext_nba
(

     Name         TEXT(40),
     Team         TEXT(40),
     Number       BIGINT,
     Position     TEXT(2),
     Age          BIGINT,
     Height       TEXT(4),
     Weight       BIGINT,
     College      TEXT(40),
     Salary       FLOAT
 )
```

(continues on next page)

(continued from previous page)

```
WRAPPER avro_fdw
OPTIONS
(
  LOCATION =  's3://sqream-demo-data/nba.avro'
);
```

**Tip:** An exact match must exist between the SQream and Avro types. For unsupported column types, you can set the type to any type and exclude it from subsequent queries.

**Note:** The **nba.avro** file is stored on S3 at `s3://sqream-demo-data/nba.avro`.

#### 3.2.3.2 Creating a Foreign Table

Before loading data, you must build the `CREATE FOREIGN TABLE` to correspond with the file structure of the inserted table.

The example in this section is based on the source `nba.avro` table shown below:

Table 2: nba.avro

| Name | Team | Number | Position | Age | Height | Weight | College | Salary |
|---|---|---|---|---|---|---|---|---|
| Avery Bradley | Boston Celtics | 0.0 | PG | 25.0 | 6-2 | 180.0 | Texas | 7730337.0 |
| Jae Crowder | Boston Celtics | 99.0 | SF | 25.0 | 6-6 | 235.0 | Marquette | 6796117.0 |
| John Holland | Boston Celtics | 30.0 | SG | 27.0 | 6-5 | 205.0 | Boston University | |
| R.J. Hunter | Boston Celtics | 28.0 | SG | 22.0 | 6-5 | 185.0 | Georgia State | 1148640.0 |
| Jonas Jerebko | Boston Celtics | 8.0 | PF | 29.0 | 6-10 | 231.0 | | 5000000.0 |
| Amir Johnson | Boston Celtics | 90.0 | PF | 29.0 | 6-9 | 240.0 | | 12000000.0 |
| Jordan Mickey | Boston Celtics | 55.0 | PF | 21.0 | 6-8 | 235.0 | LSU | 1170960.0 |
| Kelly Olynyk | Boston Celtics | 41.0 | C | 25.0 | 7-0 | 238.0 | Gonzaga | 2165160.0 |
| Terry Rozier | Boston Celtics | 12.0 | PG | 22.0 | 6-2 | 190.0 | Louisville | 1824360.0 |

The following example shows the correct file structure used to create the `CREATE FOREIGN TABLE` statement based on the **nba.avro** table:

```
CREATE FOREIGN TABLE ext_nba
(

     Name       TEXT(40),
     Team       TEXT(40),
```

(continues on next page)

(continued from previous page)

```
    Number     BIGINT,
    Position   TEXT(2),
    Age        BIGINT,
    Height     TEXT(4),
    Weight     BIGINT,
    College    TEXT(40),
    Salary     FLOAT
)
WRAPPER avro_fdw
OPTIONS
(
  LOCATION =  's3://sqream-demo-data/nba.avro'
);
```

**Tip:** An exact match must exist between the SQream and Avro types. For unsupported column types, you can set the type to any type and exclude it from subsequent queries.

**Note:** The **nba.avro** file is stored on S3 at `s3://sqream-demo-data/nba.avro`.

**Note:** The examples in the sections above are identical except for the syntax used to create the tables.

## 3.2.4 Mapping Between SQream and Avro Data Types

Mapping between SQream and Avro data types depends on the Avro data type:



### 3.2.4.1 Primitive Data Types

The following table shows the supported **Primitive** data types:

| Avro Type | SQream Type | | | |
|---|---|---|---|---|
| | Number | Date/Datetime | String | Boolean |
| `null` | Supported | Supported | Supported | Supported |
| `boolean` | | | Supported | Supported |
| `int` | Supported | | Supported | |
| `long` | Supported | | Supported | |
| `float` | Supported | | Supported | |
| `double` | Supported | | Supported | |
| `bytes` | | | | |
| `string` | | Supported | Supported | |

#### 3.2.4.2 Complex Data Types

The following table shows the supported **Complex** data types:

| Avro Type | SQream Type | | | |
|---|---|---|---|---|
| | Number | Date/Datetime | String | Boolean |
| `record` | | | | |
| `enum` | | | Supported | |
| `array` | | | | |
| `map` | | | | |
| `union` | Supported | Supported | Supported | Supported |
| `fixed` | | | | |

#### 3.2.4.3 Logical Data Types

The following table shows the supported **Logical** data types:

| Avro Type | SQream Type | | | |
|---|---|---|---|---|
| | Number | Date/Datetime | String | Boolean |
| `decimal` | Supported | | Supported | |
| `uuid` | | | Supported | |
| `date` | | Supported | Supported | |
| `time-millis` | | | | |
| `time-micros` | | | | |
| `timestamp-millis` | | Supported | Supported | |
| `timestamp-micros` | | Supported | Supported | |
| `local-timestamp-millis` | | | | |
| `local-timestamp-micros` | | | | |
| `duration` | | | | |

**Note:** Number types include **tinyint**, **smallint**, **int**, **bigint**, **real** and **float**, and **numeric**. String types include **text**.

### 3.2.5 Mapping Objects to Rows

When mapping objects to rows, each Avro object or message must contain one `record` type object corresponding to a single row in SQream. The `record` fields are associated by name to their target table columns. Additional unmapped fields will be ignored. Note that using the JSONPath option overrides this.

### 3.2.6 Ingesting Data into SQream

This section includes the following:

#### 3.2.6.1 Syntax

Before ingesting data into SQream from an Avro file, you must create a table using the following syntax:

```
COPY [schema name.]table_name
  FROM WRAPPER fdw_name
;
```

After creating a table you can ingest data from an Avro file into SQream using the following syntax:

```
avro_fdw
```

#### 3.2.6.2 Example

The following is an example of creating a table:

```
COPY t
  FROM WRAPPER fdw_name
  OPTIONS
  (
    [ copy_from_option [, ...] ]
  )
;
```

The following is an example of loading data from an Avro file into SQream:

```
WRAPPER avro_fdw
OPTIONS
(
  LOCATION =  's3://sqream-demo-data/nba.avro'
);
```

For more examples, see *Additional Examples*.

### 3.2.7 Parameters

The following table shows the Avro parameter:

| Parameter | Description |
|---|---|
| schema_name | The schema name for the table. Defaults to `public` if not specified. |

### 3.2.8 Best Practices

Because external tables do not automatically verify the file integrity or structure, SQream recommends manually verifying your table output when ingesting Avro files into SQream. This lets you determine if your table output is identical to your originally inserted table.

The following is an example of the output based on the **nba.avro** table:

```
t=> SELECT * FROM ext_nba LIMIT 10;
Name           | Team             | Number | Position | Age | Height | Weight | College ␣
↪          | Salary
--------------+-----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
```

(continues on next page)

(continued from previous page)

```
Avery Bradley | Boston Celtics |     0 | PG        |  25 | 6-2     |     180 | Texas    ␣
↪         |  7730337
Jae Crowder   | Boston Celtics |    99 | SF        |  25 | 6-6     |     235 |␣
↪Marquette         |  6796117
John Holland  | Boston Celtics |    30 | SG        |  27 | 6-5     |     205 | Boston␣
↪University |
R.J. Hunter   | Boston Celtics |    28 | SG        |  22 | 6-5     |     185 | Georgia␣
↪State     |  1148640
Jonas Jerebko | Boston Celtics |     8 | PF        |  29 | 6-10    |     231 |          ␣
↪         |  5000000
Amir Johnson  | Boston Celtics |    90 | PF        |  29 | 6-9     |     240 |          ␣
↪         | 12000000
Jordan Mickey | Boston Celtics |    55 | PF        |  21 | 6-8     |     235 | LSU      ␣
↪         |  1170960
Kelly Olynyk  | Boston Celtics |    41 | C         |  25 | 7-0     |     238 | Gonzaga  ␣
↪         |  2165160
Terry Rozier  | Boston Celtics |    12 | PG        |  22 | 6-2     |     190 |␣
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |    36 | PG        |  22 | 6-4     |     220 | Oklahoma␣
↪State    |  3431040
```

**Note:** If your table output has errors, verify that the structure of the Avro files correctly corresponds to the external table structure that you created.

### 3.2.9 Additional Examples

This section includes the following additional examples of loading data into SQream:

- *Omitting Unsupported Column Types*
- *Modifying Data Before Loading*
- *Loading a Table from a Directory of Avro Files on HDFS*
- *Loading a Table from a Directory of Avro Files on S3*

#### 3.2.9.1 Omitting Unsupported Column Types

When loading data, you can omit columns using the `NULL as` argument. You can use this argument to omit unsupported columns from queries that access external tables. By omitting them, these columns will not be called and will avoid generating a "type mismatch" error.

In the example below, the `Position` column is not supported due its type.

```
CREATE TABLE nba AS
   SELECT Name, Team, Number, NULL as Position, Age, Height, Weight, College, Salary␣
↪FROM ext_nba;
```

#### 3.2.9.2 Modifying Data Before Loading

One of the main reasons for staging data using the EXTERNAL TABLE argument is to examine and modify table contents before loading it into SQream.

For example, we can replace pounds with kilograms using the create_table_as statement

In the example below, the Position column is set to the default NULL.

```
CREATE TABLE nba AS
   SELECT name, team, number, NULL as Position, age, height, (weight / 2.205) as␣
↪weight, college, salary
           FROM ext_nba
           ORDER BY weight;
```

#### 3.2.9.3 Loading a Table from a Directory of Avro Files on HDFS

The following is an example of loading a table from a directory of Avro files on HDFS:

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER avro_fdw
OPTIONS
  (
     LOCATION =  'hdfs://hadoop-nn.piedpiper.com/rhendricks/users/*.avro'
  );

CREATE TABLE users AS SELECT * FROM ext_users;
```

For more configuration option examples, see the CREATE FOREIGN TABLE parameters.

#### 3.2.9.4 Loading a Table from a Directory of Avro Files on S3

The following is an example of loading a table from a directory of Avro files on S3:

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER avro_fdw
OPTIONS
  ( LOCATION = 's3://pp-secret-bucket/users/*.avro',
    AWS_ID = 'our_aws_id',
    AWS_SECRET = 'our_aws_secret'
   );

CREATE TABLE users AS SELECT * FROM ext_users;
```

## 3.3 Inserting Data from a CSV File

This guide covers inserting data from CSV files into SQream DB using the copy_from method.

**In this topic:**



### 3.3.1 1. Prepare CSVs

Prepare the source CSVs, with the following requirements:

- Files should be a valid CSV. By default, SQream DB's CSV parser can handle RFC 4180 standard CSVs , but can also be modified to support non-standard CSVs (with multi-character delimiters, unquoted fields, etc).
- Files are UTF-8 or ASCII encoded
- Field delimiter is an ASCII character or characters
- Record delimiter, also known as a new line separator, is a Unix-style newline (`\n`), DOS-style newline (`\r\n`), or Mac style newline (`\r`).
- Fields are optionally enclosed by double-quotes, or mandatory quoted if they contain one of the following characters:
  - The record delimiter or field delimiter

  - A double quote character
  - A newline

- If a field is quoted, any double quote that appears must be double-quoted (similar to the string literals quoting rules. For example, to encode `What are "birds"?`, the field should appear as `"What are ""birds""?"`.

  Other modes of escaping are not supported (e.g. `1,"What are \"birds\"?"` is not a valid way of escaping CSV values).

- `NULL` values can be marked in two ways in the CSV:

  - An explicit null marker. For example, `col1,\N,col3`
  - An empty field delimited by the field delimiter. For example, `col1,,col3`

> **Note:** If a text field is quoted but contains no content (`""`) it is considered an empty text field. It is not considered `NULL`.

### 3.3.2 2. Place CSVs where SQream DB workers can access

During data load, the copy_from command can run on any worker (unless explicitly speficied with the *Workload Manager*). It is important that every node has the same view of the storage being used - meaning, every SQream DB worker should have access to the files.

- For files hosted on NFS, ensure that the mount is accessible from all servers.
- For HDFS, ensure that SQream DB servers can access the HDFS name node with the correct user-id. See our *Using SQream in an HDFS Environment* guide for more information.
- For S3, ensure network access to the S3 endpoint. See our *Inserting Data Using Amazon S3* guide for more information.

### 3.3.3 3. Figure out the table structure

Prior to loading data, you will need to write out the table structure, so that it matches the file structure.

For example, to import the data from `nba.csv`, we will first look at the file:

Table 3: nba.csv

| Name | Team | Number | Position | Age | Height | Weight | College | Salary |
|---|---|---|---|---|---|---|---|---|
| Avery Bradley | Boston Celtics | 0.0 | PG | 25.0 | 6-2 | 180.0 | Texas | 7730337.0 |
| Jae Crowder | Boston Celtics | 99.0 | SF | 25.0 | 6-6 | 235.0 | Marquette | 6796117.0 |
| John Holland | Boston Celtics | 30.0 | SG | 27.0 | 6-5 | 205.0 | Boston University | |
| R.J. Hunter | Boston Celtics | 28.0 | SG | 22.0 | 6-5 | 185.0 | Georgia State | 1148640.0 |
| Jonas Jerebko | Boston Celtics | 8.0 | PF | 29.0 | 6-10 | 231.0 | | 5000000.0 |
| Amir Johnson | Boston Celtics | 90.0 | PF | 29.0 | 6-9 | 240.0 | | 12000000.0 |
| Jordan Mickey | Boston Celtics | 55.0 | PF | 21.0 | 6-8 | 235.0 | LSU | 1170960.0 |
| Kelly Olynyk | Boston Celtics | 41.0 | C | 25.0 | 7-0 | 238.0 | Gonzaga | 2165160.0 |
| Terry Rozier | Boston Celtics | 12.0 | PG | 22.0 | 6-2 | 190.0 | Louisville | 1824360.0 |

- The file format in this case is CSV, and it is stored as an S3 object.
- The first row of the file is a header containing column names.
- The record delimiter was a DOS newline (`\r\n`).
- The file is stored on S3, at `s3://sqream-demo-data/nba.csv`.

We will make note of the file structure to create a matching `CREATE TABLE` statement.

```
CREATE TABLE nba
(
   Name text(40),
   Team text(40),
   Number tinyint,
   Position text(2),
   Age tinyint,
   Height text(4),
   Weight real,
   College text(40),
   Salary float
 );
```

### 3.3.4 4. Bulk load the data with COPY FROM

The CSV is a standard CSV, but with two differences from SQream DB defaults:

- The record delimiter is not a Unix newline (`\n`), but a Windows newline (`\r\n`)
- The first row of the file is a header containing column names, which we'll want to skip.

```
COPY nba
   FROM 's3://sqream-demo-data/nba.csv'
```

(continues on next page)

(continued from previous page)

```
    WITH RECORD DELIMITER '\r\n'
         OFFSET 2;
```

Repeat steps 3 and 4 for every CSV file you want to import.

## 3.3.5 Loading different types of CSV files

copy_from contains several configuration options. See more in the COPY FROM elements section.

### 3.3.5.1 Loading a standard CSV file from a local filesystem

```
COPY table_name FROM '/home/rhendricks/file.csv';
```

### 3.3.5.2 Loading a PSV (pipe separated value) file

```
COPY table_name FROM '/home/rhendricks/file.psv' WITH DELIMITER '|';
```

### 3.3.5.3 Loading a TSV (tab separated value) file

```
COPY table_name FROM '/home/rhendricks/file.tsv' WITH DELIMITER '\t';
```

### 3.3.5.4 Loading a text file with non-printable delimiter

In the file below, the separator is `DC1`, which is represented by ASCII 17 decimal or 021 octal.

```
COPY table_name FROM 'file.txt' WITH DELIMITER E'\021';
```

### 3.3.5.5 Loading a text file with multi-character delimiters

In the file below, the separator is `'|`.

```
COPY table_name FROM 'file.txt' WITH DELIMITER '''|';
```

### 3.3.5.6 Loading files with a header row

Use `OFFSET` to skip rows.

**Note:** When loading multiple files (e.g. with wildcards), this setting affects each file separately.

```
COPY  table_name FROM 'filename.psv' WITH DELIMITER '|' OFFSET  2;
```

#### 3.3.5.7 Loading files formatted for Windows (\r\n)

```
COPY table_name FROM 'filename.psv' WITH DELIMITER '|' RECORD DELIMITER '\r\n';
```

#### 3.3.5.8 Loading a file from a public S3 bucket

**Note:** The bucket must be publicly available and objects can be listed

```
COPY nba FROM 's3://sqream-demo-data/nba.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n';
```

#### 3.3.5.9 Loading files from an authenticated S3 bucket

```
COPY nba FROM 's3://secret-bucket/*.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n' AWS_ID
↪'12345678' AWS_SECRET 'super_secretive_secret';
```

#### 3.3.5.10 Loading files from an HDFS storage

```
COPY nba FROM 'hdfs://hadoop-nn.piedpiper.com/rhendricks/*.csv' WITH OFFSET 2 RECORD␣
↪DELIMITER '\r\n';
```

#### 3.3.5.11 Saving rejected rows to a file

See capturing_rejected_rows for more information about the error handling capabilities of `COPY FROM`.

```
COPY  table_name FROM 'filename.psv'  WITH DELIMITER '|'
                                      ERROR_LOG  '/temp/load_error.log' -- Save error␣
↪log
                                      ERROR_VERBOSITY 0; -- Only save rejected rows
```

#### 3.3.5.12 Stopping the load if a certain amount of rows were rejected

```
COPY  table_name  FROM  'filename.csv'  WITH  delimiter  '|'
                                        ERROR_LOG  '/temp/load_err.log' -- Save␣
↪error log
                                        OFFSET 2 -- skip header row
                                        LIMIT  100 -- Only load 100 rows
                                        STOP AFTER 5 ERRORS; -- Stop the load if 5␣
↪errors reached
```

#### 3.3.5.13 Load CSV files from a set of directories

Use glob patterns (wildcards) to load multiple files to one table.

```
COPY table_name  from  '/path/to/files/2019_08_*/*.csv';
```

#### 3.3.5.14 Rearrange destination columns

When the source of the files does not match the table structure, tell the `COPY` command what the order of columns should be

```
COPY table_name (fifth, first, third) FROM '/path/to/files/*.csv';
```

**Note:** Any column not specified will revert to its default value or `NULL` value if nullable

#### 3.3.5.15 Loading non-standard dates

If files contain dates not formatted as `ISO8601`, tell `COPY` how to parse the column. After parsing, the date will appear as `ISO8601` inside SQream DB.

In this example, `date_col1` and `date_col2` in the table are non-standard. `date_col3` is mentioned explicitly, but can be left out. Any column that is not specified is assumed to be `ISO8601`.

```
COPY table_name FROM '/path/to/files/*.csv' WITH PARSERS 'date_col1=YMD,date_col2=MDY,
↪date_col3=default';
```

**Tip:** The full list of supported date formats can be found under the Supported date formats section of the copy_from reference.

## 3.4 Inserting Data from a Parquet File

This guide covers inserting data from Parquet files into SQream DB using FOREIGN TABLE.

**In this topic:**

- *Loading a table from a directory of Parquet files on HDFS*
- *Loading a table from a bucket of files on S3*

### 3.4.1 1. Prepare the files

Prepare the source Parquet files, with the following requirements:

| SQream DB type → Parquet source | BOOL | TINYINT | SMALLINT | INT | BIGINT | REAL | DOUBLE | TEXT[1] | DATE | DATETIME |
|---|---|---|---|---|---|---|---|---|---|---|
| BOOLEAN | Supported | | | | | | | | | |
| INT16 | | | Supported | | | | | | | |
| INT32 | | | | Supported | | | | | | |
| INT64 | | | | | Supported | | | | | |
| FLOAT | | | | | | Supported | | | | |
| DOUBLE | | | | | | | Supported | | | |
| BYTE_ARRAY / FIXED_LEN_BYTE_ARRAY[2] | | | | | | | | Supported | | |
| INT96[3] | | | | | | | | | | Supported[4] |

- If a Parquet file has an unsupported type, such as `enum`, `uuid`, `time`, `json`, `bson`, `lists`, `maps`, but the data is not referenced in the table (it does not appear in the SELECT query), the statement will succeed. If the column is referenced, an error will be thrown to the user, explaining that the type is not Supported, but the column may be ommited. This can be worked around. See more information in the examples.

### 3.4.2 2. Place Parquet files where SQream DB workers can access them

Any worker may try to access files (unless explicitly speficied with the *Workload Manager*). It is important that every node has the same view of the storage being used - meaning, every SQream DB worker should have access to the files.

- For files hosted on NFS, ensure that the mount is accessible from all servers.
- For HDFS, ensure that SQream DB servers can access the HDFS name node with the correct user-id. See our *Using SQream in an HDFS Environment* guide for more information.
- For S3, ensure network access to the S3 endpoint. See our *Inserting Data Using Amazon S3* guide for more information.

[1] Text values include `TEXT`

[2] With UTF8 annotation

[3] With `TIMESTAMP_NANOS` or `TIMESTAMP_MILLIS` annotation

[4] Any microseconds will be rounded down to milliseconds.

### 3.4.3 3. Figure out the table structure

Prior to loading data, you will need to write out the table structure, so that it matches the file structure.

For example, to import the data from `nba.parquet`, we will first look at the source table:

Table 4: nba.parquet

| Name | Team | Number | Position | Age | Height | Weight | College | Salary |
|---|---|---|---|---|---|---|---|---|
| Avery Bradley | Boston Celtics | 0.0 | PG | 25.0 | 6-2 | 180.0 | Texas | 7730337.0 |
| Jae Crowder | Boston Celtics | 99.0 | SF | 25.0 | 6-6 | 235.0 | Marquette | 6796117.0 |
| John Holland | Boston Celtics | 30.0 | SG | 27.0 | 6-5 | 205.0 | Boston University | |
| R.J. Hunter | Boston Celtics | 28.0 | SG | 22.0 | 6-5 | 185.0 | Georgia State | 1148640.0 |
| Jonas Jerebko | Boston Celtics | 8.0 | PF | 29.0 | 6-10 | 231.0 | | 5000000.0 |
| Amir Johnson | Boston Celtics | 90.0 | PF | 29.0 | 6-9 | 240.0 | | 12000000.0 |
| Jordan Mickey | Boston Celtics | 55.0 | PF | 21.0 | 6-8 | 235.0 | LSU | 1170960.0 |
| Kelly Olynyk | Boston Celtics | 41.0 | C | 25.0 | 7-0 | 238.0 | Gonzaga | 2165160.0 |
| Terry Rozier | Boston Celtics | 12.0 | PG | 22.0 | 6-2 | 190.0 | Louisville | 1824360.0 |

- The file is stored on S3, at `s3://sqream-demo-data/nba.parquet`.

We will make note of the file structure to create a matching `CREATE EXTERNAL TABLE` statement.

```
CREATE FOREIGN TABLE ext_nba
(
     Name       TEXT(40),
     Team       TEXT(40),
     Number     BIGINT,
     Position   TEXT(2),
     Age        BIGINT,
     Height     TEXT(4),
     Weight     BIGINT,
     College    TEXT(40),
     Salary     FLOAT
 )
 WRAPPER parquet_fdw
 OPTIONS
 (
   LOCATION =  's3://sqream-demo-data/nba.parquet'
 );
```

**Tip:** Types in SQream DB must match Parquet types exactly.

If the column type isn't Supported, a possible workaround is to set it to any arbitrary type and then exclude it from subsequent queries.

## 3.4.4 4. Verify table contents

External tables do not verify file integrity or structure, so verify that the table definition matches up and contains the correct data.

```
t=> SELECT * FROM ext_nba LIMIT 10;
Name          | Team           | Number | Position | Age | Height | Weight | College ␣
↪          | Salary
--------------+----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas    ␣
↪          |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |␣
↪Marquette         |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston␣
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia␣
↪State      |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |          ␣
↪          |  5000000
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |          ␣
↪          | 12000000
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU      ␣
↪          |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga  ␣
↪          |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |␣
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma␣
↪State     |  3431040
```

If any errors show up at this stage, verify the structure of the Parquet files and match them to the external table structure you created.

## 3.4.5 5. Copying data into SQream DB

To load the data into SQream DB, use the create_table_as statement:

```
CREATE TABLE nba AS
   SELECT * FROM ext_nba;
```

### 3.4.5.1 Working around unSupported column types

Suppose you only want to load some of the columns - for example, if one of the columns isn't Supported.

By ommitting unSupported columns from queries that access the `EXTERNAL TABLE`, they will never be called, and will not cause a "type mismatch" error.

For this example, assume that the `Position` column isn't Supported because of its type.

```
CREATE TABLE nba AS
   SELECT Name, Team, Number, NULL as Position, Age, Height, Weight, College, Salary␣
↪FROM ext_nba;

-- We ommitted the unSupported column `Position` from this query, and replaced it␣
↪with a default ``NULL`` value, to maintain the same table structure.
```

#### 3.4.5.2 Modifying data during the copy process

One of the main reasons for staging data with EXTERNAL TABLE is to examine the contents and modify them before loading them.

Assume we are unhappy with weight being in pounds, because we want to use kilograms instead. We can apply the transformation as part of the create_table_as statement.

Similar to the previous example, we will also set the Position column as a default NULL.

```
CREATE TABLE nba AS
   SELECT name, team, number, NULL as position, age, height, (weight / 2.205) as␣
↪weight, college, salary
           FROM ext_nba
           ORDER BY weight;
```

### 3.4.6 Further Parquet loading examples

create_foreign_table contains several configuration options. See more in the CREATE FOREIGN TABLE parameters section.

#### 3.4.6.1 Loading a table from a directory of Parquet files on HDFS

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER parquet_fdw
OPTIONS
  (
     LOCATION =  'hdfs://hadoop-nn.piedpiper.com/rhendricks/users/*.parquet'
  );

CREATE TABLE users AS SELECT * FROM ext_users;
```

#### 3.4.6.2 Loading a table from a bucket of files on S3

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER parquet_fdw
OPTIONS
  ( LOCATION = 's3://pp-secret-bucket/users/*.parquet',
    AWS_ID = 'our_aws_id',
    AWS_SECRET = 'our_aws_secret'
   );

CREATE TABLE users AS SELECT * FROM ext_users;
```

## 3.5 Inserting Data from an ORC File

This guide covers inserting data from ORC files into SQream DB using FOREIGN TABLE.

### 3.5.1 1. Prepare the files

Prepare the source ORC files, with the following requirements:

| SQream DB type → ORC source | BOOL | TINYINT | SMALLINT | INT | BIGINT | REAL | DOUBLE | TEXT[1] | DATE | DATETIME |
|---|---|---|---|---|---|---|---|---|---|---|
| boolean | Supported | Supported[2] | Supported[?] | Supported[?] | Supported[?] | | | | | |
| tinyint | ○[3] | Supported | Supported | Supported | Supported | | | | | |
| smallint | ○[?] | ○[4] | Supported | Supported | Supported | | | | | |
| int | ○[?] | ○[?] | ○[?] | Supported | Supported | | | | | |
| bigint | ○[?] | ○[?] | ○[?] | ○[?] | Supported | | | | | |
| float | | | | | | Supported | Supported | | | |
| double | | | | | | Supported | Supported | | | |
| string / char / text | | | | | | | | Supported | | |
| date | | | | | | | | | Supported | Supported |
| timestamp, timestamp with timezone | | | | | | | | | | Supported |

- If an ORC file has an unsupported type like `binary`, `list`, `map`, and `union`, but the data is not referenced in the table (it does not appear in the SELECT query), the statement will succeed. If the column is referenced, an error will be thrown to the user, explaining that the type is not supported, but the column may be ommited. This can be worked around. See more information in the examples.

[1] Text values include `TEXT`
[2] Boolean values are cast to 0, 1
[3] Will succeed if all values are 0, 1
[4] Will succeed if all values fit the destination type

### 3.5.2 2. Place ORC files where SQream DB workers can access them

Any worker may try to access files (unless explicitly speficied with the *Workload Manager*). It is important that every node has the same view of the storage being used - meaning, every SQream DB worker should have access to the files.

- For files hosted on NFS, ensure that the mount is accessible from all servers.
- For HDFS, ensure that SQream DB servers can access the HDFS name node with the correct user-id. See our *Using SQream in an HDFS Environment* guide for more information.
- For S3, ensure network access to the S3 endpoint. See our *Inserting Data Using Amazon S3* guide for more information.

### 3.5.3 3. Figure out the table structure

Prior to loading data, you will need to write out the table structure, so that it matches the file structure.

For example, to import the data from `nba.orc`, we will first look at the source table:

Table 5: nba.orc

| Name | Team | Number | Position | Age | Height | Weight | College | Salary |
|---|---|---|---|---|---|---|---|---|
| Avery Bradley | Boston Celtics | 0.0 | PG | 25.0 | 6-2 | 180.0 | Texas | 7730337.0 |
| Jae Crowder | Boston Celtics | 99.0 | SF | 25.0 | 6-6 | 235.0 | Marquette | 6796117.0 |
| John Holland | Boston Celtics | 30.0 | SG | 27.0 | 6-5 | 205.0 | Boston University | |
| R.J. Hunter | Boston Celtics | 28.0 | SG | 22.0 | 6-5 | 185.0 | Georgia State | 1148640.0 |
| Jonas Jerebko | Boston Celtics | 8.0 | PF | 29.0 | 6-10 | 231.0 | | 5000000.0 |
| Amir Johnson | Boston Celtics | 90.0 | PF | 29.0 | 6-9 | 240.0 | | 12000000.0 |
| Jordan Mickey | Boston Celtics | 55.0 | PF | 21.0 | 6-8 | 235.0 | LSU | 1170960.0 |
| Kelly Olynyk | Boston Celtics | 41.0 | C | 25.0 | 7-0 | 238.0 | Gonzaga | 2165160.0 |
| Terry Rozier | Boston Celtics | 12.0 | PG | 22.0 | 6-2 | 190.0 | Louisville | 1824360.0 |

- The file is stored on S3, at `s3://sqream-demo-data/nba.orc`.

We will make note of the file structure to create a matching `CREATE FOREIGN TABLE` statement.

```
CREATE FOREIGN TABLE ext_nba
(
     Name        TEXT(40),
     Team        TEXT(40),
     Number      BIGINT,
     Position    TEXT(2),
     Age         BIGINT,
     Height      TEXT(4),
     Weight      BIGINT,
     College     TEXT(40),
```

(continues on next page)

(continued from previous page)

```
    Salary      FLOAT
)
  WRAPPER orc_fdw
  OPTIONS
    (
       LOCATION = 's3://sqream-demo-data/nba.orc'
    );
```

**Tip:** Types in SQream DB must match ORC types according to the table above.

If the column type isn't supported, a possible workaround is to set it to any arbitrary type and then exclude it from subsequent queries.

## 3.5.4 4. Verify table contents

External tables do not verify file integrity or structure, so verify that the table definition matches up and contains the correct data.

```
t=> SELECT * FROM ext_nba LIMIT 10;
Name          | Team           | Number | Position | Age | Height | Weight | College ␣
↪         | Salary
--------------+----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas    ␣
↪         |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |␣
↪Marquette         |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston␣
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia␣
↪State     |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |          ␣
↪         |  5000000
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |          ␣
↪         | 12000000
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU      ␣
↪         |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga  ␣
↪         |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |␣
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma␣
↪State     |  3431040
```

If any errors show up at this stage, verify the structure of the ORC files and match them to the external table structure you created.

### 3.5.5 5. Copying data into SQream DB

To load the data into SQream DB, use the create_table_as statement:

```
CREATE TABLE nba AS
   SELECT * FROM ext_nba;
```

#### 3.5.5.1 Working around unsupported column types

Suppose you only want to load some of the columns - for example, if one of the columns isn't supported.

By ommitting unsupported columns from queries that access the `EXTERNAL TABLE`, they will never be called, and will not cause a "type mismatch" error.

For this example, assume that the `Position` column isn't supported because of its type.

```
CREATE TABLE nba AS
   SELECT Name, Team, Number, NULL as Position, Age, Height, Weight, College, Salary␣
↪FROM ext_nba;

-- We ommitted the unsupported column `Position` from this query, and replaced it␣
↪with a default ``NULL`` value, to maintain the same table structure.
```

#### 3.5.5.2 Modifying data during the copy process

One of the main reasons for staging data with `EXTERNAL TABLE` is to examine the contents and modify them before loading them.

Assume we are unhappy with weight being in pounds, because we want to use kilograms instead. We can apply the transformation as part of the create_table_as statement.

Similar to the previous example, we will also set the `Position` column as a default `NULL`.

```
CREATE TABLE nba AS
   SELECT name, team, number, NULL as position, age, height, (weight / 2.205) as␣
↪weight, college, salary
           FROM ext_nba
           ORDER BY weight;
```

### 3.5.6 Further ORC loading examples

create_foreign_table contains several configuration options. See more in the CREATE FOREIGN TABLE parameters section.

#### 3.5.6.1 Loading a table from a directory of ORC files on HDFS

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER orc_fdw
  OPTIONS
    (
      LOCATION = 'hdfs://hadoop-nn.piedpiper.com/rhendricks/users/*.ORC'
    );

CREATE TABLE users AS SELECT * FROM ext_users;
```

#### 3.5.6.2 Loading a table from a bucket of files on S3

```
CREATE FOREIGN TABLE ext_users
  (id INT NOT NULL, name TEXT(30) NOT NULL, email TEXT(50) NOT NULL)
WRAPPER orc_fdw
OPTIONS
  (  LOCATION = 's3://pp-secret-bucket/users/*.ORC',
     AWS_ID = 'our_aws_id',
     AWS_SECRET = 'our_aws_secret'
   )
;

CREATE TABLE users AS SELECT * FROM ext_users;
```

For information about database tools and interfaces that SQream supports, see Third Party Tools.

CHAPTER

# FOUR

# CONNECTING TO SQREAM

SQream supports the most common database tools and interfaces, giving you direct access through a variety of drivers, connectors, and visualiztion tools and utilities. The tools described on this page have been tested and approved for use with SQream. Most third party tools that work through JDBC, ODBC, and Python should work.

This section provides information about the following third party tools:

## 4.1 Client Platforms

These topics explain how to install and connect a variety of third party tools.

Browse the articles below, in the sidebar, or use the search to find the information you need.

### 4.1.1 Overview

SQream DB is designed to work with most common database tools and interfaces, allowing you direct access through a variety of drivers, connectors, tools, vizualisers, and utilities.

The tools listed have been tested and approved for use with SQream DB. Most 3$^{rd}$ party tools that work through JDBC, ODBC, and Python should work.

If you are looking for a tool that is not listed, SQream and our partners can help. Go to SQream Support or contact your SQream account manager for more information.

#### 4.1.1.1 Connect to SQream Using Informatica Cloud Services

##### 4.1.1.1.1 Overview

The **Connecting to SQream Using Informatica Cloud Services** page is quick start guide for connecting to SQream using Informatica cloud services.

It describes the following:

- *Supported SQream Driver Versions*

#### 4.1.1.1.1.1 Establishing a Connection between SQream and Informatica

The **Establishing a Connection between SQream and Informatica** page describes how to establish a connection between SQream and the Informatica data integration Cloud.

**To establish a connection between SQream and the Informatica data integration Cloud:**

1. Go to the Informatica Cloud homepage.
2. Do one of the following:
   1. Log in using your credentials.
   2. Log in using your SAML Identity Provider.
3. From the **Services** window, select **Administrator** or click **Show all services** to show all services.

   The SQream dashboard is displayed.
4. In the menu on the left, click **Runtime Environments**.

   The **Runtime Environments** panel is displayed.
5. Click **Download Secure Agent**.
6. When the **Download the Secure Agent** panel is displayed, do the following:
   1. Select a platform (Windows 64 or Linux 64).
   2. Click **Copy** and save the token on your local hard drive.

      The token is used in combination with your user name to authorize the agent to access your account.
7. Click **Download**.

   The installation begins.
8. When the **Informatica Cloud Secure Agent Setup** panel is displayed, click **Next**.
9. Provide your **User Name** and **Install Token** and click **Register**.
10. From the Runtime Environments panel, click **New Runtime Environment**.

    The **New Secure Agent Group** window is displayed.

11. On the New Secure Agent Group window, click **OK** to connect your Runtime Environment with the running agent.

    **Note:** If you do not download Secure Agent, you will not be able to connect your Runtime Environment with the running agent and continue establishing a connection between SQream and the Informatica data integration Cloud.

##### 4.1.1.1.1.2 Establishing a Connection In Your Environment

The **Establishing a Connection In Your Environment** describes the following:



##### 4.1.1.1.1.3 Establishing an ODBC DSN Connection In Your Environment

After establishing a connection between SQream and Informatica you can establish an ODBC DSN connection in your environment.

**To establish an ODBC connection in your environment:**

1. Click **Add**.

2. Click **Configure**.

   **Note:** Verify that **Use Server Picker** is selected.

3. Click **Test**.

4. Verify that the connection has tested successfully.

5. Click **Save**.

6. Click **Actions** > **Publish**.

##### 4.1.1.1.1.4 Establishing a JDBC Connection In Your Environment

After establishing a connection between SQream and Informatica you can establish a JDBC connection in your environment.

**To establish a JDBC connection in your environment:**

1. Create a new DB connection by clicking **Connections** > **New Connection**.

   The **New Connection** window is displayed.

2. In the **JDBC_IC Connection Properties** section, in the **JDBC Connection URL** field, establish a JDBC connection by providing the correct connection string.

   For connection string examples, see Connection Strings.

3. Click **Test**.

4. Verify that the connection has tested successfully.

5. Click **Save**.

6. Click **Actions** > **Publish**.

##### 4.1.1.1.1.5 Supported SQream Driver Versions

SQream supports the following SQream driver versions:

- **JDBC** - Version 4.3.4 and above.

- **ODBC** - Version 4.0.0 and above.

### 4.1.1.2 MicroStrategy

#### 4.1.1.2.1 Overview

This document is a Quick Start Guide that describes how to install MicroStrategy and connect a datasource to the MicroStrategy dasbhoard for analysis.

The **Connecting to SQream Using MicroStrategy** page describes the following:



##### 4.1.1.2.1.1 What is MicroStrategy?

MicroStrategy is a Business Intelligence software offering a wide variety of data analytics capabilities. SQream uses the MicroStrategy connector for reading and loading data into SQream.

MicroStrategy provides the following:

- Data discovery
- Advanced analytics
- Data visualization

- Embedded BI
- Banded reports and statements

For more information about Microstrategy, see MicroStrategy.

*Back to Overview*

##### 4.1.1.2.1.2 Connecting a Data Source

1. Activate the **MicroStrategy Desktop** app. The app displays the Dossiers panel to the right.

2. Download the most current version of the SQream JDBC driver.

3. Click **Dossiers** and **New Dossier**. The **Untitled Dossier** panel is displayed.

4. Click **New Data**.

5. From the **Data Sources** panel, select **Databases** to access data from tables. The **Select Import Options** panel is displayed.

6. Select one of the following:
   - Build a Query
   - Type a Query
   - Select Tables

7. Click **Next**.

8. In the Data Source panel, do the following:
   1. From the **Database** dropdown menu, select **Generic**. The **Host Name**, **Port Number**, and **Database Name** fields are removed from the panel.

   2. In the **Version** dropdown menu, verify that **Generic DBMS** is selected.

   3. Click **Show Connection String**.

   4. Select the **Edit connection string** checkbox.

   5. From the **Driver** dropdown menu, select a driver for one of the following connectors:

- **JDBC** - The SQream driver is not integrated with MicroStrategy and does not appear in the dropdown menu. However, to proceed, you must select an item, and in the next step you must specify the path to the SQream driver that you installed on your machine.
- **ODBC** - SQreamDB ODBC

6. In the **Connection String** text box, type the relevant connection string and path to the JDBC jar file using the following syntax:

```
$ jdbc:Sqream://<host and port>/<database name>;user=<username>;password=
↪<password>sqream;[<optional parameters>; ...]
```

The following example shows the correct syntax for the JDBC connector:

```
jdbc;MSTR_JDBC_JAR_FOLDER=C:\path\to\jdbc\folder;DRIVER=<driver>;URL=
↪{jdbc:Sqream://<host and port>/<database name>;user=<username>;password=
↪<password>;[<optional parameters>; ...];}
```

The following example shows the correct syntax for the ODBC connector:

```
odbc:Driver={SqreamODBCDriver};DSN={SQreamDB ODBC};Server=<Host>;Port=<Port>;
↪Database=<database name>;User=<username>;Password=<password>;Cluster=
↪<boolean>;
```

For more information about the available **connection parameters** and other examples, see *Connection Parameters*.

7. In the **User** and **Password** fields, fill out your user name and password.

8. In the **Data Source Name** field, type **SQreamDB**.

9. Click **Save**. The SQreamDB that you picked in the Data Source panel is displayed.

9. In the **Namespace** menu, select a namespace. The tables files are displayed.

10. Drag and drop the tables into the panel on the right in your required order.

11. **Recommended** - Click **Prepare Data** to customize your data for analysis.

12. Click **Finish**.

13. From the **Data Access Mode** dialog box, select one of the following:
    - Connect Live
    - Import as an In-memory Dataset

Your populated dashboard is displayed and is ready for data discovery and analytics.

*Back to Overview*

#### 4.1.1.2.1.3 Supported SQream Drivers

The following list shows the supported SQream drivers and versions:

- **JDBC** - Version 4.3.3 and higher.
- **ODBC** - Version 4.0.0.

*Back to Overview*

### 4.1.1.3 Connecting to SQream Using Pentaho Data Integration

#### 4.1.1.3.1 Overview

This document is a Quick Start Guide that describes how to install Pentaho, create a transformation, and define your output.

The Connecting to SQream Using Pentaho page describes the following:

- *Installing Pentaho*
- *Installing and setting up the JDBC driver*
- *Creating a transformation*
- *Defining your output*
- *Importing your data*

##### 4.1.1.3.1.1 Installing Pentaho

To install PDI, see the Pentaho Community Edition (CE) Installation Guide.

The **Pentaho Community Edition (CE) Installation Guide** describes how to do the following:

- Downloading the PDI software.
- Installing the **JRE (Java Runtime Environment)** and **JDK (Java Development Kit)**.
- Setting up the JRE and JDK environment variables for PDI.

*Back to Overview*

##### 4.1.1.3.1.2 Installing and Setting Up the JDBC Driver

After installing Pentaho you must install and set up the JDBC driver. This section explains how to set up the JDBC driver using Pentaho. These instructions use Spoon, the graphical transformation and job designer associated with the PDI suite.

You can install the driver by copying and pasting the SQream JDBC .jar file into your **<directory>/design-tools/data-integration/lib** directory.

**NOTE:** Contact your SQream license account manager for the JDBC .jar file.

*Back to Overview*

#### 4.1.1.3.1.3 Creating a Transformation

After installing Pentaho you can create a transformation.

**To create a transformation:**

1. Use the CLI to open the PDI client for your operating system (Windows):

```
$ spoon.bat
```

2. Open the spoon.bat file from its folder location.
3. In the **View** tab, right-click **Transformations** and click **New**.

   A new transformation tab is created.
4. In the **Design** tab, click **Input** to show its file contents.
5. Drag and drop the **CSV file input** item to the new transformation tab that you created.
6. Double-click **CSV file input**. The **CSV file input** panel is displayed.
7. In the **Step name** field, type a name.
8. To the right of the **Filename** field, click **Browse**.
9. Select the file that you want to read from and click **OK**.
10. In the CSV file input window, click **Get Fields**.
11. In the **Sample data** window, enter the number of lines you want to sample and click **OK**. The default setting is **100**.

    The tool reads the file and suggests the field name and type.
12. In the CSV file input window, click **Preview**.
13. In the **Preview size** window, enter the number of rows you want to preview and click **OK**. The default setting is **1000**.
14. Verify that the preview data is correct and click **Close**.
15. Click **OK** in the **CSV file input** window.

*Back to Overview*

#### 4.1.1.3.1.4 Defining Your Output

After creating your transformation you must define your output.

**To define your output:**

1. In the **Design** tab, click **Output**.

   The Output folder is opened.
2. Drag and drop **Table output** item to the Transformation window.
3. Double-click **Table output** to open the **Table output** dialog box.
4. From the **Table output** dialog box, type a **Step name** and click **New** to create a new connection. Your **steps** are the building blocks of a transformation, such as file input or a table output.

   The **Database Connection** window is displayed with the **General** tab selected by default.

5. Enter or select the following information in the Database Connection window and click **Test**.

   The following table shows and describes the information that you need to fill out in the Database Connection window:

| No. | Element Name | Description |
|---|---|---|
| 1 | Connection name | Enter a name that uniquely describes your connection, such as **sample-data**. |
| 2 | Connection type | Select **Generic database**. |
| 3 | Access | Select **Native (JDBC)**. |
| 4 | Custom connection URL | Insert **jdbc:Sqream://<host:port>/<database name>;user=<username>;password=<password>;[<optional parameters>; ...];**. The IP is a node in your SQream cluster and is the name or schema of the database you want to connect to. Verify that you have not used any leading or trailing spaces. |
| 5 | Custom driver class name | Insert **com.sqream.jdbc.SQDriver**. Verify that you have not used any leading or trailing spaces. |
| 6 | Username | Your SQreamdb username. If you leave this blank, you will be prompted to provide it when you connect. |
| 7 | Password | Your password. If you leave this blank, you will be prompted to provide it when you connect. |

   The following message is displayed:

6. Click **OK** in the window above, in the Database Connection window, and Table Output window.

*Back to Overview*

#### 4.1.1.3.1.5 Importing Data

After defining your output you can begin importing your data.

For more information about backing up users, permissions, or schedules, see Backup and Restore Pentaho Repositories

**To import data:**

1. Double-click the **Table output** connection that you just created.
2. To the right of the **Target schema** field, click **Browse** and select a schema name.
3. Click **OK**. The selected schema name is displayed in the **Target schema** field.
4. Create a new hop connection between the **CSV file input** and **Table output** steps:
   1. On the CSV file input step item, click the **new hop connection** icon.
   2. Drag an arrow from the **CSV file input** step item to the **Table output** step item.
   3. Release the mouse button. The following options are displayed.
   4. Select **Main output of step**.

5. Double-click **Table output** to open the **Table output** dialog box.
6. In the **Target table** field, define a target table name.
7. Click **SQL** to open the **Simple SQL editor.**
8. In the **Simple SQL editor**, click **Execute**.

   The system processes and displays the results of the SQL statements.
9. Close all open dialog boxes.
10. Click the play button to execute the transformation.

11. Click **Run**.

    The **Execution Results** are displayed.

*Back to Overview*

#### 4.1.1.4 Connect to SQream Using PHP

##### 4.1.1.4.1 Overview

PHP is an open source scripting language that executes scripts on servers. The **Connect to PHP** page explains how to connect to a SQream cluster, and describes the following:

- *Installing PHP*
- *Configuring PHP*
- *Operating PHP*

###### 4.1.1.4.1.1 Installing PHP

**To install PHP:**

1. Download the JDBC driver installer from the SQream Drivers page.

2. Create a DSN.

3. Install the **uODBC** extension for your PHP installation.

   For more information, navigate to PHP Documentation and see the topic menu on the right side of the page.

#### 4.1.1.4.1.2 Configuring PHP

You can configure PHP in one of the following ways:

- When compiling, configure PHP to enable uODBC using `./configure --with-pdo-odbc=unixODBC,/usr/local`.
- Install `php-odbc` and `php-pdo` along with PHP using your distribution package manager. SQream recommends a minimum of version 7.1 for the best results.

---

**Note:** PHP's string size limitations truncates fetched text, which you can override by doing one of the following:

- Increasing the **php.ini** default setting, such as the *odbc.defaultlrl* to **10000**.
- Setting the size limitation in your code before making your connection using **ini_set("odbc.defaultlrl", "10000");**.
- Setting the size limitation in your code before fetchng your result using **odbc_longreadlen($result, "10000");**.

---

#### 4.1.1.4.1.3 Operating PHP

After configuring PHP, you can test your connection.

**To test your connection:**

1. Create a test connection file using the correct parameters for your SQream installation, as shown below:

```
<?php // Construct a DSN connection string
$dsn  = "SqreamODBC"; // Create a connection
$conn = odbc_connect($dsn, '', '');
if (!($conn)) {
    echo "Connection to SQream DB via ODBC failed: " . odbc_errormsg($conn);
}
$sql = "SELECT show_version()"; // Execute the query
$rs  = odbc_exec($conn, $sql);
while (odbc_fetch_row($rs)) {
    for ($i = 1; $i <= odbc_num_fields($rs); $i++) {
        echo "Result is " . odbc_result($rs, $i);
    }
}
echo "\n";
odbc_close($conn); // Finally, close the connection
?>
```

   For more information, download the sample `PHP example connection file` shown above.

   The following is an example of a valid DSN line:

```
$dsn = "odbc:Driver={SqreamODBCDriver};Server=192.168.0.5;Port=5000;
↪Database=master;User=rhendricks;Password=super_secret;Service=sqream";
```

2. Run the PHP file either directly with PHP (`php test.php`) or through a browser.

   For more information about supported DSN parameters, see *ODBC DSN Parameters*.

#### 4.1.1.5 Connect to SQream Using Power BI Desktop

##### 4.1.1.5.1 Overview

**Power BI Desktop** lets you connect to SQream and use underlying data as with other data sources in Power BI Desktop.

SQream integrates with Power BI Desktop to do the following:

- Extract and transform your datasets into usable visual models in approximately one minute.

- Use **DAX** functions **(Data Analysis Expressions)** to analyze your datasets.

- Refresh datasets as needed or by using scheduled jobs.

SQream uses Power BI for extracting data sets using the following methods:

- **Direct query** - Direct queries lets you connect easily with no errors, and refreshes Power BI artifacts, such as graphs and reports, in a considerable amount of time in relation to the time taken for queries to run using the SQream SQL CLI Reference guide.

- **Import** - Lets you extract datasets from remote databases.

The **Connect to SQream Using Power BI** page describes the following:



###### 4.1.1.5.1.1 Prerequisites

To connect to SQream, the following must be installed:

- **ODBC data source administrator** - 32 or 64, depending on your operating system. For Windows users, the ODBC data source administrator is embedded within the operating system.
- **SQream driver** - The SQream application required for interacting with the ODBC according to the configuration specified in the ODBC administrator tool.

#### 4.1.1.5.1.2 Installing Power BI Desktop

**To install Power BI Desktop:**

1. Download Power BI Desktop 64x.

2. Download and configure your ODBC driver.

   For more information about configuring your ODBC driver, see ODBC.

3. Navigate to **Windows > Documents** and create a folder called **Power BI Desktop Custom Connectors**.

4. In the **Power BI Desktop** folder, create a folder called **Custom Connectors**.

5. From the Client Drivers page, download the **PowerQuery.mez** file.

5. Save the PowerQuery.mez file in the **Custom Connectors** folder you created in Step 3.

6. Open the Power BI application.

7. Navigate to **File > Options and Settings > Option > Security > Data Extensions**, and select **(Not Recommended) Allow any extension to load without validation or warning**.

8. Restart the Power BI Desktop application.

9. From the **Get Data** menu, select **SQream**.

10. Click **Connect** and provide the information shown in the following table:

| Element Name | Description |
|---|---|
| Server | Provide the network address to your database server. You can use a hostname or an IP address. |
| Port | Provide the port that the database is responding to at the network address. |
| Database | Provide the name of your database or the schema on your database server. |
| User | Provide a SQreamdb username. |
| Passwords | Provide a password for your user. |

11. Under **Data Connectivity mode**, select **DirectQuery mode**.

12. Click **Connect**.

13. Provide your user name and password and click **Connect**.

##### 4.1.1.5.1.3 Best Practices for Power BI

SQream recommends using Power BI in the following ways for acquiring the best performance metrics:

- Creating bar, pie, line, or plot charts when illustrating one or more columns.
- Displaying trends and statuses using visual models.
- Creating a unified view using **PowerQuery** to connect different data sources into a single dashboard.

##### 4.1.1.5.1.4 Supported SQream Driver Versions

SQream supports the following SQream driver versions:

- The **PowerQuery Connector** is an additional layer on top of the ODBC.
- SQream Driver Installation (ODBC v4.1.1) - Contact your administrator for the link to download ODBC v4.1.1.

##### 4.1.1.5.1.5 Related Information

For more information, see the Glossary.

### 4.1.1.6 Connect to SQream Using R

You can use R to interact with a SQream DB cluster.

This tutorial is a guide that will show you how to connect R to SQream DB.

**In this topic:**



#### 4.1.1.6.1 JDBC

1. Get the *SQream DB JDBC driver*.
2. In R, install RJDBC

```
> install.packages("RJDBC")
Installing package into 'C:/Users/r/...'
(as 'lib' is unspecified)

package 'RJDBC' successfully unpacked and MD5 sums checked
```

3. Import the RJDBC library

```
> library(RJDBC)
```

4. Set the classpath and initialize the JDBC driver which was previously installed. For example, on Windows:

```
> cp = c("C:\\Program Files\\SQream Technologies\\JDBC Driver\\2020.1-3.2.0\\
↪sqream-jdbc-3.2.jar")
> .jinit(classpath=cp)
> drv <- JDBC("com.sqream.jdbc.SQDriver","C:\\Program Files\\SQream Technologies\\
↪JDBC Driver\\2020.1-3.2.0\\sqream-jdbc-3.2.jar")
```

5. Open a connection with a JDBC connection string and run your first statement

```
> con <- dbConnect(drv,"jdbc:Sqream://127.0.0.1:3108/master;user=rhendricks;
↪password=Tr0ub4dor&3;cluster=true")

> dbGetQuery(con,"select top 5 * from t")
   xint  xtinyint xsmallint xbigint
1     1        82      5067       1
2     2        14      1756       2
3     3        91     22356       3
4     4        84     17232       4
5     5        13     14315       5
```

6. Close the connection

```
> close(con)
```

#### 4.1.1.6.1.1 A full example

```
> library(RJDBC)
> cp = c("C:\\Program Files\\SQream Technologies\\JDBC Driver\\2020.1-3.2.0\\sqream-
↪jdbc-3.2.jar")
> .jinit(classpath=cp)
> drv <- JDBC("com.sqream.jdbc.SQDriver","C:\\Program Files\\SQream Technologies\\
↪JDBC Driver\\2020.1-3.2.0\\sqream-jdbc-3.2.jar")
> con <- dbConnect(drv,"jdbc:Sqream://127.0.0.1:3108/master;user=rhendricks;
↪password=Tr0ub4dor&3;cluster=true")
> dbGetQuery(con,"select top 5 * from t")
   xint  xtinyint xsmallint xbigint
1     1        82      5067       1
2     2        14      1756       2
3     3        91     22356       3
4     4        84     17232       4
5     5        13     14315       5
> close(con)
```

#### 4.1.1.6.2 ODBC

1. Install the *SQream DB ODBC driver* for your operating system, and create a DSN.
2. In R, install RODBC

```
> install.packages("RODBC")
Installing package into 'C:/Users/r/...'
(as 'lib' is unspecified)

package 'RODBC' successfully unpacked and MD5 sums checked
```

3. Import the RODBC library

```
> library(RODBC)
```

4. Open a connection handle to an existing DSN (`my_cool_dsn` in this example)

```
> ch <- odbcConnect("my_cool_dsn",believeNRows=F)
```

5. Run your first statement

```
> sqlQuery(ch,"select top 5 * from t")
   xint  xtinyint xsmallint xbigint
1     1        82      5067       1
2     2        14      1756       2
3     3        91     22356       3
4     4        84     17232       4
5     5        13     14315       5
```

6. Close the connection

```
> close(ch)
```

##### 4.1.1.6.2.1 A full example

```
> library(RODBC)
> ch <- odbcConnect("my_cool_dsn",believeNRows=F)
> sqlQuery(ch,"select top 5 * from t")
   xint  xtinyint xsmallint xbigint
1     1        82      5067       1
2     2        14      1756       2
3     3        91     22356       3
4     4        84     17232       4
5     5        13     14315       5
> close(ch)
```

### 4.1.1.7 Connecting to SQream Using SAP BusinessObjects

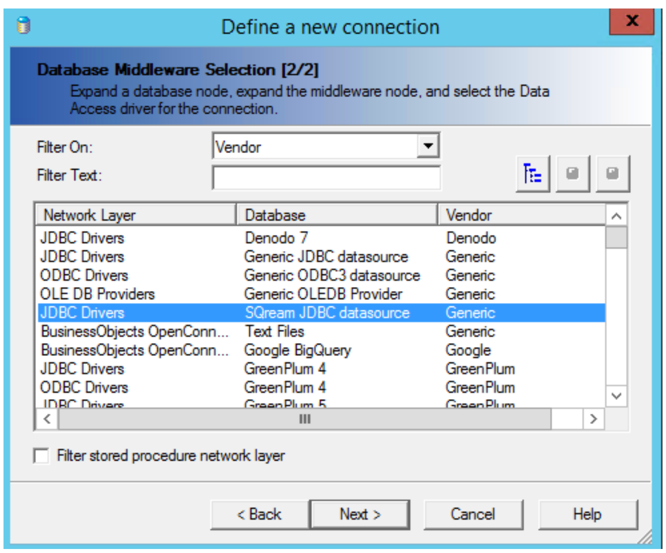
The **Connecting to SQream Using SAP BusinessObjects** guide includes the following sections:

- *Overview*
- *Establising a New Connection Using a Generic JDCB Connector*

#### 4.1.1.7.1 Overview

The **Connecting to SQream Using SAP BusinessObjects** guide describes the best practices for configuring a connection between SQream and the SAP BusinessObjects BI platform. SAP BO's multi-tier architecture includes both client and server components, and this guide describes integrating SQream with SAP BO's object client tools using a generic JDBC connector. The instructions in this guide are relevant to both the **Universe Design Tool (UDT)** and the **Information Design Tool (IDT)**. This document only covers how to establish a connection using the generic out-of-the-box JDBC connectors, and does not cover related business object products, such as the **Business Objects Data Integrator**.

The **Define a new connection** window below shows the generic JDBC driver, which you can use to establish a new connection to a database.




SAP BO also lets you customize the interface to include a SQream data source.

#### 4.1.1.7.2 Establising a New Connection Using a Generic JDCB Connector

This section shows an example of using a generic JDBC connector to establish a new connection.

**To establish a new connection using a generic JDBC connector:**

1. In the fields, provide a user name, password, database URL, and JDBC class.

   The following is the correct format for the database URL:

   ```
   <pre>jdbc:Sqream://<ipaddress>:3108/<nameofdatabase>
   ```

SQream recommends quickly testing your connection to SQream by selecting the Generic JDBC data source in the **Define a new connection** window. When you connect using a generic JDBC data source you do not need to modify your configuration files, but are limited to the out-of-the-box settings defined in the default **jdbc.prm** file.

**Note:** Modifying the jdbc.prm file for the generic driver impacts all other databases using the same driver.

For more information, see Connection String Examples.

2. (Optonal)If you are using the generic JDBC driver specific to SQream, modify the jdbc.sbo file to include the SQream JDBC driver location by adding the following lines under the Database section of the file:

```
Database Active="Yes" Name="SQream JDBC data source">
<JDBCDriver>
<ClassPath>
<Path>C:\Program Files\SQream Technologies\JDBC Driver\2021.2.0-4.5.3\sqream-jdbc-
↪4.5.3.jar</Path>
</ClassPath>
</Parameter>
<Parameter Name="JDBC Class">
com.sqream.jdbc.SQDriver

</JDBCDriver>
</DataBase>
```

3. Restart the BusinessObjects server.

   When the connection is established, **SQream** is listed as a driver selection.

#### 4.1.1.8 SAS Viya

##### 4.1.1.8.1 Overview

SAS Viya is a cloud-enabled analytics engine used for producing useful insights. The **Connect to SQream Using SAS Viya** page describes how to connect to SAS Viya, and describes the following:



###### 4.1.1.8.1.1 Installing SAS Viya

The **Installing SAS Viya** section describes the following:

#### 4.1.1.8.1.2 Downloading SAS Viya

Integrating with SQream has been tested with SAS Viya v.03.05 and newer.

To download SAS Viya, see SAS Viya.

#### 4.1.1.8.1.3 Installing the JDBC Driver

The SQream JDBC driver is required for establishing a connection between SAS Viya and SQream.

**To install the JDBC driver:**

1. Download the JDBC driver.

2. Unzip the JDBC driver into a location on the SAS Viya server.

   SQream recommends creating the directory `/opt/sqream` on the SAS Viya server.

#### 4.1.1.8.1.4 Configuring SAS Viya

After installing the JDBC driver, you must configure the JDBC driver from the SAS Studio so that it can be used with SQream Studio.

**To configure the JDBC driver from the SAS Studio:**

1. Sign in to the SAS Studio.

2. From the **New** menu, click **SAS Program**.

3. Configure the SQream JDBC connector by adding the following rows:

```
options sastrace='d,d,d,d'
sastraceloc=saslog
nostsuffix
msglevel=i
sql_ip_trace=(note,source)
DEBUG=DBMS_SELECT;

options validvarname=any;

libname sqlib jdbc driver="com.sqream.jdbc.SQDriver"
   classpath="/opt/sqream/sqream-jdbc-4.0.0.jar"
   URL="jdbc:Sqream://sqream-cluster.piedpiper.com:3108/raviga;cluster=true"
   user="rhendricks"
   password="Tr0ub4dor3"
   schema="public"
   PRESERVE_TAB_NAMES=YES
   PRESERVE_COL_NAMES=YES;
```

For more information about writing a connection string, see **Connect to SQream DB with a JDBC Application** and navigate to Connection String.

#### 4.1.1.8.1.5 Operating SAS Viya

The **Operating SAS Viya** section describes the following:



#### 4.1.1.8.1.6 Using SAS Viya Visual Analytics

This section describes how to use SAS Viya Visual Analytics.

**To use SAS Viya Visual Analytics:**

1. Log in to SAS Viya Visual Analytics using your credentials:
2. Click **New Report**.
3. Click **Data**.
4. Click **Data Sources**.
5. Click the **Connect** icon.
6. From the **Type** menu, select **Database**.
7. Provide the required information and select **Persist this connection beyond the current session**.
8. Click **Advanced** and provide the required information.
9. Add the following additional parameters by clicking **Add Parameters**:

| Name | Value |
|---|---|
| class | com.sqream.jdbc.SQDriver |
| class-Path | *<path_to_jar_file>* |
| url | jdbc:Sqream://*<IP>*:*<port>*/*<database>*;cluster=true |
| user-name | <username> |
| pass-word | <password> |

10. Click **Test Connection**.
11. If the connection is successful, click **Save**.

If your connection is not successful, see *Troubleshooting SAS Viya* below.

#### 4.1.1.8.1.7 Troubleshooting SAS Viya

The **Best Practices and Troubleshooting** section describes the following best practices and troubleshooting procedures when connecting to SQream using SAS Viya:



#### 4.1.1.8.1.8 Inserting Only Required Data

When using SAS Viya, SQream recommends using only data that you need, as described below:

- Insert only the data sources you need into SAS Viya, excluding tables that don't require analysis.

- To increase query performance, add filters before analyzing. Every modification you make while analyzing data queries the SQream database, sometimes several times. Adding filters to the datasource before exploring limits the amount of data analyzed and increases query performance.

#### 4.1.1.8.1.9 Creating a Separate Service for SAS Viya

SQream recommends creating a separate service for SAS Viya with the DWLM. This reduces the impact that Tableau has on other applications and processes, such as ETL. In addition, this works in conjunction with the load balancer to ensure good performance.

#### 4.1.1.8.1.10 Locating the SQream JDBC Driver

In some cases, SAS Viya cannot locate the SQream JDBC driver, generating the following error message:

```
java.lang.ClassNotFoundException: com.sqream.jdbc.SQDriver
```

**To locate the SQream JDBC driver:**

1. Verify that you have placed the JDBC driver in a directory that SAS Viya can access.

2. Verify that the classpath in your SAS program is correct, and that SAS Viya can access the file that it references.

3. Restart SAS Viya.

For more troubleshooting assistance, see the SQream Support Portal.

##### 4.1.1.8.1.11 Supporting TEXT

In SAS Viya versions lower than 4.0, casting `TEXT` to `CHAR` changes the size to 1,024, such as when creating a table including a `TEXT` column. This is resolved by casting `TEXT` into `CHAR` when using the JDBC driver.

#### 4.1.1.9 Connect to SQream Using SQL Workbench

You can use SQL Workbench to interact with a SQream DB cluster. SQL Workbench/J is a free SQL query tool, and is designed to run on any JRE-enabled environment.

This tutorial is a guide that will show you how to connect SQL Workbench to SQream DB.

**In this topic:**

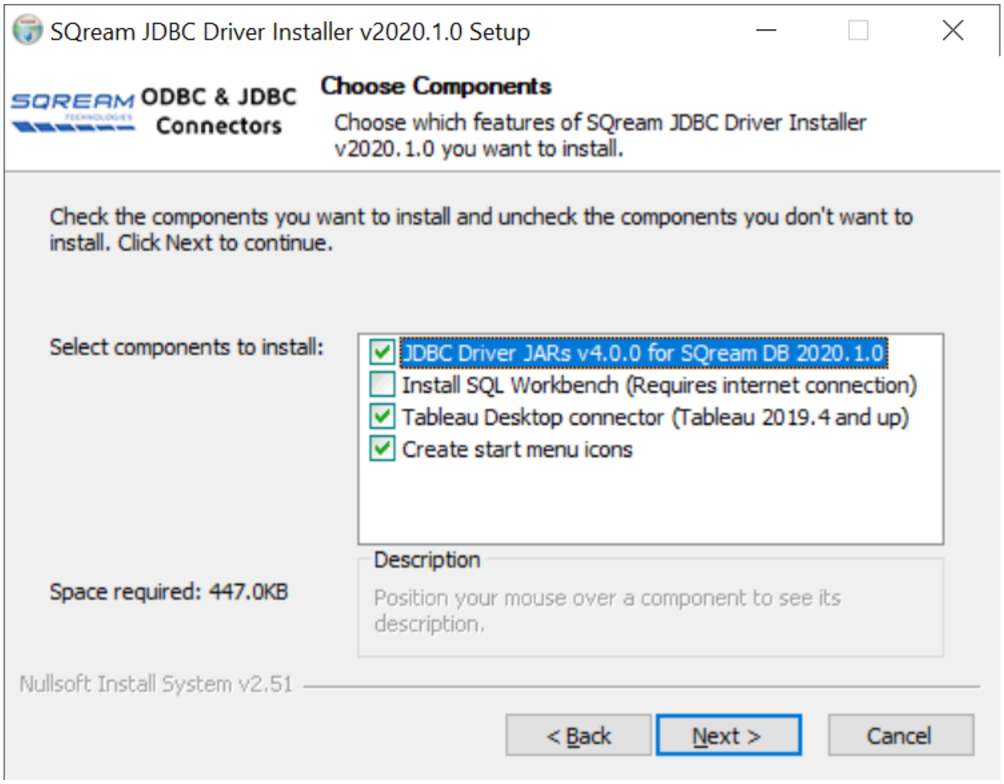


##### 4.1.1.9.1 Installing SQL Workbench with the SQream Installer

This section applies to Windows only.

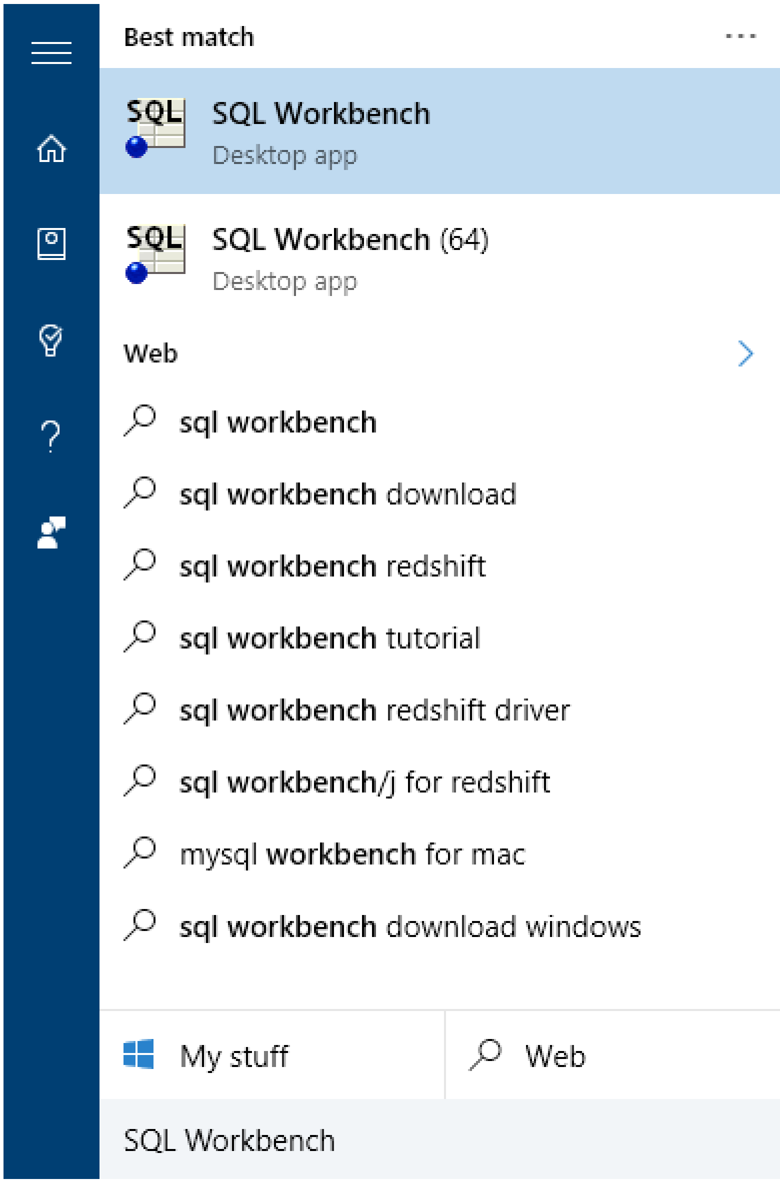
SQream DB's driver installer for Windows can install the Java prerequisites and SQL Workbench for you.

1. Get the JDBC driver installer available for download from the SQream Drivers page. The Windows installer takes care of the Java prerequisites and subsequent configuration.
2. Install the driver by following the on-screen instructions in the easy-to-follow installer. By default, the installer does not install SQL Workbench. Make sure to select the item!




**Note:** The installer will install SQL Workbench in `C:\Program Files\SQream Technologies\SQLWorkbench` by default. You can change this path during the installation.

1. Once finished, SQL Workbench is installed and contains the necessary configuration for connecting to SQream DB clusters.
2. Start SQL Workbench from the Windows start menu. Be sure to select **SQL Workbench (64)** if you're on 64-bit Windows.




You are now ready to create a profile for your cluster. Continue to *Creating a new connection profile*.

#### 4.1.1.9.2 Installing SQL Workbench Manually

This section applies to Linux and MacOS only.

#### 4.1.1.9.2.1 Install Java Runtime

Both SQL Workbench and the SQream DB JDBC driver require Java 1.8 or newer. You can install either Oracle Java or OpenJDK.

**Oracle Java**

Download and install Java 8 from Oracle for your platform - https://www.java.com/en/download/manual.jsp

**OpenJDK**

For Linux and BSD, see https://openjdk.java.net/install/

For Windows, SQream recommends Zulu 8 https://www.azul.com/downloads/zulu-community/?&version=java-8-lts&architecture=x86-64-bit&package=jdk

#### 4.1.1.9.2.2 Get the SQream DB JDBC Driver

SQream DB's JDBC driver is provided as a zipped JAR file, available for download from the SQream Drivers page.

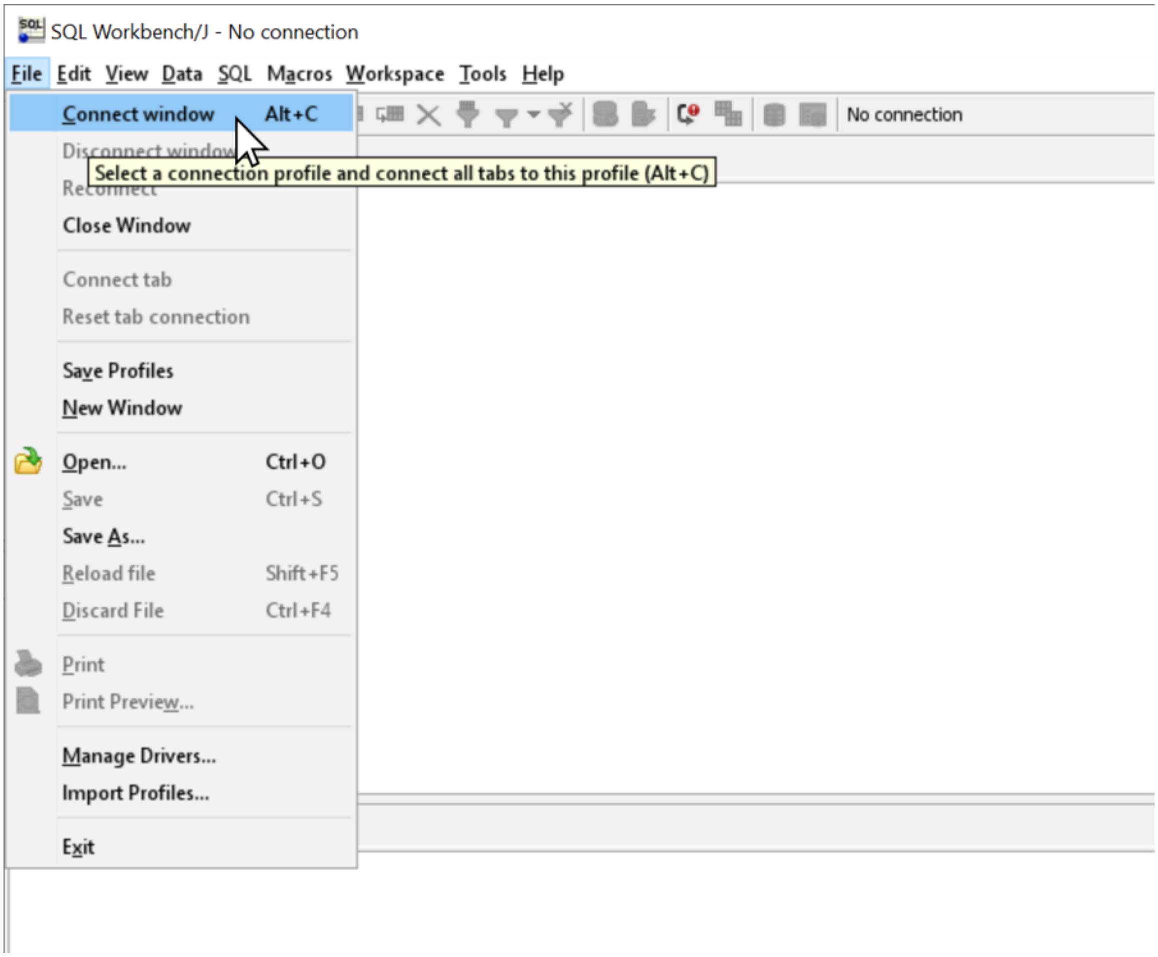
Download and extract the JAR file from the zip archive.

#### 4.1.1.9.2.3 Install SQL Workbench

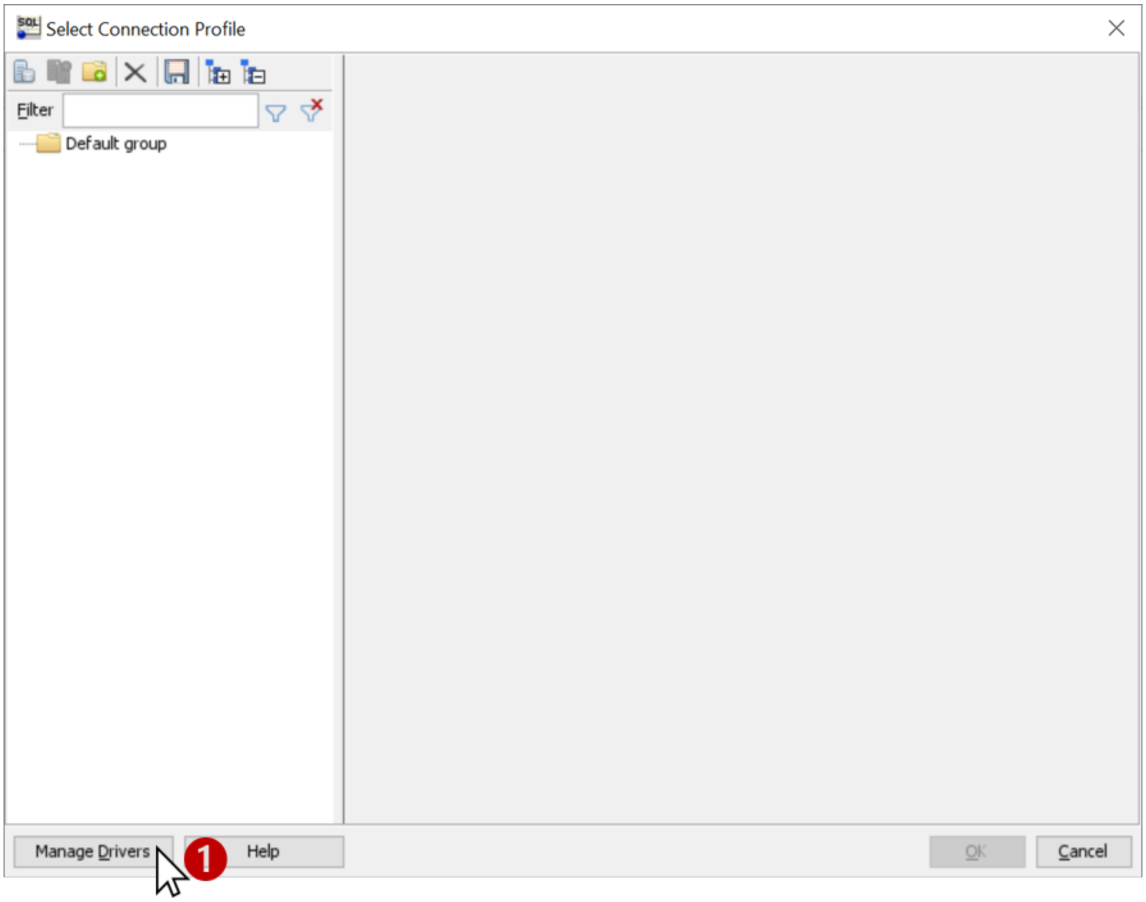
1. Download the latest stable release from https://www.sql-workbench.eu/downloads.html . The **Generic package for all systems** is recommended.
2. Extract the downloaded ZIP archive into a directory of your choice.
3. Start SQL workbench. If you are using 64 bit windows, run `SQLWorkbench64.exe` instead of `SQLWOrkbench.exe`.

#### 4.1.1.9.2.4 Setting up the SQream DB JDBC Driver Profile

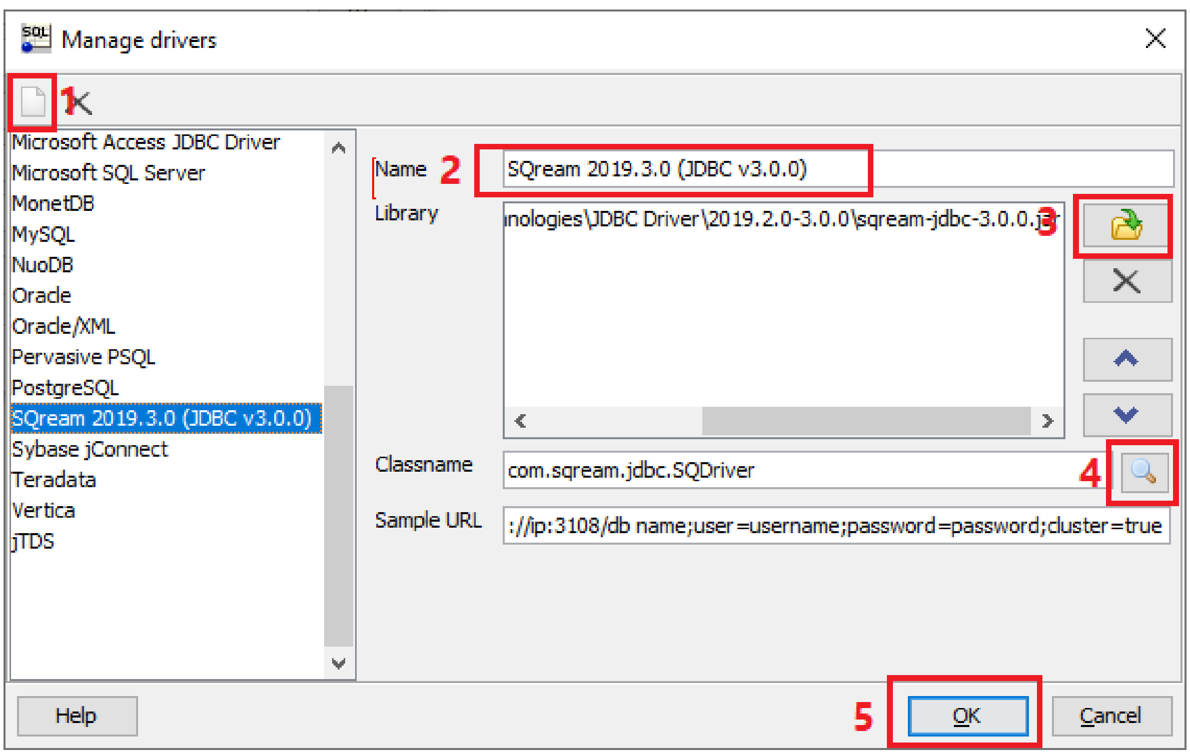
1. Define a connection profile - *File* ‣ *Connect window (Alt+C)*




2. Open the drivers management window - *Manage Drivers*



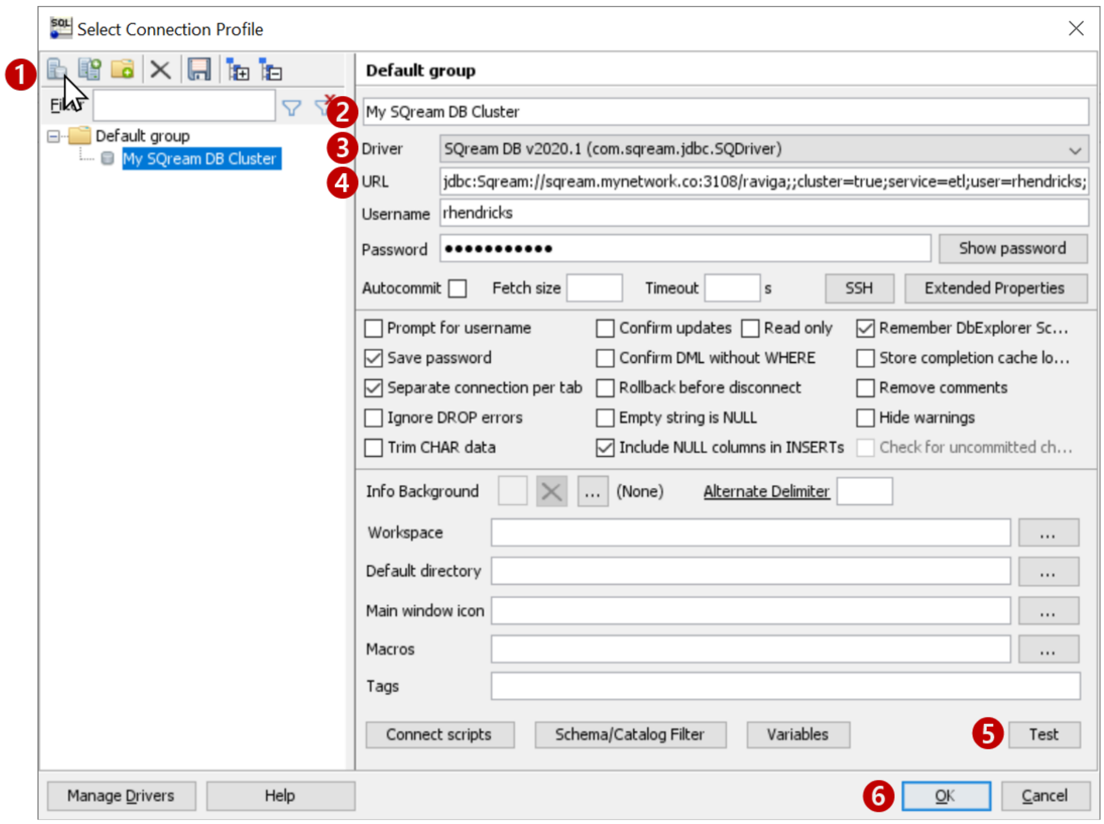

3. Create the SQream DB driver profile



1. Click on the Add new driver button ("New" icon)
2. Name the driver as you see fit. We recommend calling it SQream DB <version>, where <version> is the version you have installed.
3. Add the JDBC drivers from the location where you extracted the SQream DB JDBC JAR.

   If you used the SQream installer, the file will be in `C:\Program Files\SQream Technologies\JDBC Driver\`
4. Click the magnifying glass button to detect the classname automatically. Other details are purely optional
5. Click OK to save and return to "new connection screen"

#### 4.1.1.9.3 Create a New Connection Profile for Your Cluster




1. Create new connection by clicking the New icon (top left)
2. Give your connection a descriptive name
3. Select the SQream Driver that was created in the previous screen
4. Type in your connection string. To find out more about your connection string (URL), see the Connection string documentation.
5. Text the connection details
6. Click OK to save the connection profile and connect to SQream DB

#### 4.1.1.9.4 Suggested Optional Configuration

If you installed SQL Workbench manually, you can set a customization to help SQL Workbench show information correctly in the DB Explorer panel.

1. Locate your workbench.settings file On Windows, typically: `C:\Users\<user name>\.sqlworkbench\workbench.settings` On Linux, `$HOME/.sqlworkbench`
2. Add the following line at the end of the file:

```
workbench.db.sqreamdb.schema.retrieve.change.catalog=true
```

3. Save the file and restart SQL Workbench

### 4.1.1.10 Connecting to SQream Using Tableau

#### 4.1.1.10.1 Overview

SQream's Tableau connector plugin, based on standard JDBC, enables storing and fast querying large volumes of data.

The **Connecting to SQream Using Tableau** page is a Quick Start Guide that describes how install Tableau and the JDBC driver and connect to SQream for data analysis. It also describes using best practices and troubleshoot issues that may occur while installing Tableau. SQream supports both Tableau Desktop and Tableau Server on Windows, MacOS, and Linux distributions.

For more information on SQream's integration with Tableau, see Tableau's Extension Gallery.

The Connecting to SQream Using Tableau page describes the following:



##### 4.1.1.10.1.1 Installing the JDBC Driver and Tableau Connector Plugin

This section describes how to install the JDBC driver using the fully-integrated Tableau connector plugin (Tableau Connector, or **.taco** file). SQream has been tested with Tableau versions 9.2 and newer.

You can connect to SQream using Tableau by doing one of the following:

- **For MacOS or Linux** - See *Installing the JDBC Driver*.

#### 4.1.1.10.1.2 Installing the JDBC Driver

If you are using MacOS, Linux, or the Tableau server, after installing the Tableau Desktop application you can install the JDBC driver manually. When the driver is installed, you can connect to SQream.

**To install the JDBC driver:**

1. Download the JDBC installer and SQream Tableau connector (.taco) file from the *from the client drivers page*.

2. Based on your operating system, your Tableau driver directory is located in one of the following places:

   - **Tableau Desktop on MacOS:** *~/Library/Tableau/Drivers*

   - **Tableau Desktop on Windows:** *C:\Program Files\Tableau\Drivers*

   - **Tableau on Linux**: */opt/tableau/tableau_driver/jdbc*

   Note the following when installing the JDBC driver:

   - You must have read permissions on the .jar file.

   - Tableau requires a JDBC 4.0 or later driver.

   - Tableau requires a Type 4 JDBC driver.

   - The latest 64-bit version of Java 8 is installed.

3. Install the **SQreamDB.taco** file by moving the SQreamDB.taco file into the Tableau connectors directory.

   Based on the installation method that you used, your Tableau driver directory is located in one of the following places:

   - **Tableau Desktop on Windows:** *C:\Users\<your user>\My Tableau Repository\Connectors*

   - **Tableau Desktop on MacOS:** *~/My Tableau Repository/Connectors*

You can now restart Tableau Desktop or Server to begin using the SQream driver by connecting to SQream as described in the section below.

#### 4.1.1.10.1.3 Connecting to SQream

After installing the JDBC driver you can connect to SQream.

**To connect to SQream:**

1. Start Tableau Desktop.

2. In the **Connect** menu, in the **To a Server** sub-menu, click **More…**.

   More connection options are displayed.

3. Select **SQream DB by SQream Technologies**.

   The **New Connection** dialog box is displayed.

4. In the New Connection dialog box, fill in the fields and click **Sign In**.

   The following table describes the fields:

| Item | Description | Example |
|---|---|---|
| Server | Defines the server of the SQream worker. | `127.0.0.1` or `sqream.mynetwork.co` |
| Port | Defines the TCP port of the SQream worker. | `3108` when using a load balancer, or `5100` when connecting directly to a worker with SSL. |
| Database | Defines the database to establish a connection with. | `master` |
| Cluster | Enables (`true`) or disables (`false`) the load balancer. After enabling or disabling the load balance, verify the connection. | |
| Username | Specifies the username of a role to use when connecting. | `rhendricks` |
| Password | Specifies the password of the selected role. | `Tr0ub4dor&3` |
| Require SSL (recommended) | Sets SSL as a requirement for establishing this connection. | |

The connection is established and the data source page is displayed.

#### 4.1.1.10.1.4 Setting Up SQream Tables as Data Sources

After connecting to SQream you must set up the SQream tables as data sources.

**To set up SQream tables as data sources:**

1. From the **Table** menu, select the desired database and schema.

   SQream's default schema is **public**.

2. Drag the desired tables into the main area (labeled **Drag tables here**).

   This area is also used for specifying joins and data source filters.

3. Open a new sheet to analyze data.

#### 4.1.1.10.1.5 Tableau Best Practices and Troubleshooting

This section describes the following best practices and troubleshooting procedures when connecting to SQream using Tableau:



#### 4.1.1.10.1.6 Using Tableau's Table Query Syntax

Dragging your desired tables into the main area in Tableau builds queries based on its own syntax. This helps ensure increased performance, while using views or custom SQL may degrade performance. In addition, SQream recommends using the create_view to create pre-optimized views, which your datasources point to.

#### 4.1.1.10.1.7 Creating a Separate Service for Tableau

SQream recommends creating a separate service for Tableau with the DWLM. This reduces the impact that Tableau has on other applications and processes, such as ETL. In addition, this works in conjunction with the load balancer to ensure good performance.

#### 4.1.1.10.1.8 Troubleshooting Workbook Performance Before Deploying to the Tableau Server

Tableau has a built-in performance recorder that shows how time is being spent. If you're seeing slow performance, this could be the result of a misconfiguration such as setting concurrency too low.

Use the Tableau Performance Recorder for viewing the performance of queries run by Tableau. You can use this information to identify queries that can be optimized by using views.

#### 4.1.1.10.1.9 Troubleshooting Error Codes

Tableau may be unable to locate the SQream JDBC driver. The following message is displayed when Tableau cannot locate the driver:

```
Error Code: 37CE01A3, No suitable driver installed or the URL is incorrect
```

**To troubleshoot error codes:**

If Tableau cannot locate the SQream JDBC driver, do the following:

1. Verify that the JDBC driver is located in the correct directory:

- **Tableau Desktop on Windows:** *C:Program FilesTableauDrivers*

- **Tableau Desktop on MacOS:** *~/Library/Tableau/Drivers*

- **Tableau on Linux**: */opt/tableau/tableau_driver/jdbc*

2. Find the file path for the JDBC driver and add it to the Java classpath:

- **For Linux** - `export CLASSPATH=<absolute path of SQream DB JDBC driver>;$CLASSPATH`
- **For Windows** - add an environment variable for the classpath:

If you experience issues after restarting Tableau, see the SQream support portal.

#### 4.1.1.11 Connecting to SQream Using Talend

##### 4.1.1.11.1 Overview

This page describes how to use Talend to interact with a SQream cluster. The Talend connector is used for reading data from a SQream cluster and loading data into SQream. In addition, this page provides a viability report on Talend's comptability with SQream for stakeholders.

The **Connecting to SQream Using Talend** describes the following:



###### 4.1.1.11.1.1 Creating a New Metadata JDBC DB Connection

**To create a new metadata JDBC DB connection:**

1. In the **Repository** panel, nagivate to **Metadata** and right-click **Db connections**.

2. Select **Create connection**.

3. In the **Name** field, type a name.

   Note that the name cannot contain spaces.

4. In the **Purpose** field, type a purpose and click **Next**.

   Note that you cannot continue to the next step until you define both a Name and a Purpose.

5. In the **DB Type** field, select **JDBC**.

6. In the **JDBC URL** field, type the relevant connection string.

   For connection string examples, see Connection Strings.

7. In the **Drivers** field, click the **Add** button.

   The **"newLine"** entry is added.

8. One the **"newLine'** entry, click the ellipsis.

   The **Module** window is displayed.

9. From the Module window, select **Artifact repository(local m2/nexus)** and select **Install a new module**.

10. Click the ellipsis.

    Your hard drive is displayed.

11. Navigate to a **JDBC jar file** (such as **sqream-jdbc-4.5.3.jar**)and click **Open**.

12. Click **Detect the module install status**.

13. Click **OK**.

    The JDBC that you selected is displayed in the **Driver** field.

14. Click **Select class name**.

15. Click **Test connection**.

    If a driver class is not found (for example, you didn't select a JDBC jar file), the following error message is displayed:

    After creating a new metadata JDBC DB connection, you can do the following:

    - Use your new metadata connection.

    - Drag it to the **job** screen.

    - Build Talend components.

    For more information on loading data from JSON files to the Talend Open Studio, see How to Load Data from JSON Files in Talend.

#### 4.1.1.11.1.2 Supported SQream Drivers

The following list shows the supported SQream drivers and versions:

- **JDBC** - Version 4.3.3 and higher.

- **ODBC** - Version 4.0.0. This version requires a Bridge to connect. For more information on the required Bridge, see Connecting Talend on Windows to an ODBC Database.

#### 4.1.1.11.1.3 Supported Data Sources

Talend Cloud connectors let you create reusable connections with a wide variety of systems and environments, such as those shown below. This lets you access and read records of a range of diverse data.

- **Connections:** Connections are environments or systems for storing datasets, including databases, file systems, distributed systems and platforms. Because these systems are reusable, you only need to establish connectivity with them once.

- **Datasets:** Datasets include database tables, file names, topics (Kafka), queues (JMS) and file paths (HDFS). For more information on the complete list of connectors and datasets that Talend supports, see Introducing Talend Connectors.

#### 4.1.1.11.1.4 Known Issues

As of 6/1/2021 schemas were not displayed for tables with identical names.

If you experience issues using Talend, see the SQream support portal.

### 4.1.1.12 Connecting to SQream Using TIBCO Spotfire

#### 4.1.1.12.1 Overview

The **TIBCO Spotfire** software is an analytics solution that enables visualizing and exploring data through dashboards and advanced analytics.

This document is a Quick Start Guide that describes the following:

- *Establishing a Connection between TIBCO Spotfire and SQream*
- *Troubleshooting*

#### 4.1.1.12.1.1 Establishing a Connection between TIBCO Spotfire and SQream

TIBCO Spotfire supports the following versions:

- **JDBC driver** - Version 4.5.2
- **ODBC driver** - Version 4.1.1

SQream supports TIBCO Spotfire version 7.12.0.

The **Establishing a JDBC Connection between TIBCO Spotfire and SQream** section describes the following:

- *Creating a JDBC Connection*
- *Creating an ODBC Connection*
- *Creating the SQream Data Source Template*
- *Creating a Data Source*
- *Creating an Information Link*

#### 4.1.1.12.1.2 Creating a JDBC Connection

For TIBCO Spotfire to recognize SQream, you must add the correct JDBC jar file to Spotfire's loaded binary folder. The following is an example of a path to the Spotfire loaded binaries folder: `C:\tibco\tss\7.12.0\tomcat\bin`.

For the complete TIBCO Spotfire documentation, see TIBCO Spotfire® JDBC Data Access Connectivity Details.

#### 4.1.1.12.1.3 Creating an ODBC Connection

**To create an ODBC connection**

1. Install and configure ODBC on Windows.

   For more information, see *Install and Configure ODBC on Windows*.

2. Launch the TIBCO Spotfire application.

3. From the **File** menu click **Add Data Tables**.

   The **Add Database Tables** window is displayed.

4. Click **Add** and select **Database**.

   The **Open Database** window is displayed.

5. In the **Data source type** area, select **ODBC SQream** (Odbc Data Provider) and click **Configure**.

   The **Configure Data Source and Connection** window is displayed.

6. Select **System or user data source** and from the drop-down menu select the DSN of your data source (SQreamDB).

7. Provide your database username and password and click **OK**.

8. In the **Open Database** window, click **OK**.

   The **Specify Tables and Columns** window is displayed.

9. In the **Specify Tables and Columns** window, select the checkboxes corresponding to the tables and columns that you want to include in your SQL statement.

10. In the **Data source name** field, set your data source name and click **OK**.

    Your data source is displayed in the **Data tables** area.

11. In the **Add Data Tables** dialog, click **OK** to load the data from your ODBC data source into Spotfire.

**Note:** Verify that you have checked the SQL statement.

#### 4.1.1.12.1.4 Creating the SQream Data Source Template

After creating a connection, you can create your SQream data source template.

**To create your SQream data source template:**

1. Log in to the TIBCO Spotfire Server Configuration Tool.

2. From the **Configuration** tab, in the **Configuration Start** menu, click **Data Source Templates**.

   The **Data Source Templates** list is displayed.

3. From the Data Source Templates list do one of the following:

- Override an existing template:

  1. In the template text field, select an existing template.

  2. Copy and paste your data source template text.

- Create a new template:

  1. Click **New**.

     The **Add Data Source Template** window is displayed.

  2. In the **Name** field, define your template name.

  3. In the **Data Source Template** text field, copy and paste your data source template text.

     The following is an example of a data source template:

```
<jdbc-type-settings>
  <type-name>SQream   </type-name>
  <driver>com.sqream.jdbc.SQDriver   </driver>
  <connection-url-pattern>jdbc:Sqream://&lt;host&gt;:&lt;port&gt;/database;
↪user=sqream;password=sqream;cluster=true   </connection-url-pattern>
  <supports-catalogs>true   </supports-catalogs>
  <supports-schemas>true   </supports-schemas>
  <supports-procedures>false   </supports-procedures>
  <table-types>TABLE,EXTERNAL_TABLE   </table-types>
  <java-to-sql-type-conversions>
   <type-mapping>
      <from>Bool   </from>
      <to>Integer   </to>
    </type-mapping>
    <type-mapping>
      <from>VARCHAR(2048)   </from>
      <to>String   </to>
    </type-mapping>
    <type-mapping>
      <from>INT   </from>
      <to>Integer   </to>
    </type-mapping>
    <type-mapping>
      <from>BIGINT   </from>
```

(continues on next page)

(continued from previous page)

```
      <to>LongInteger   </to>
    </type-mapping>
    <type-mapping>
      <from>Real   </from>
      <to>Real   </to>
    </type-mapping>
         <type-mapping>
      <from>Decimal   </from>
      <to>Float   </to>
    </type-mapping>
     <type-mapping>
      <from>Numeric   </from>
      <to>Float   </to>
    </type-mapping>
    <type-mapping>
      <from>Date   </from>
      <to>DATE   </to>
    </type-mapping>
    <type-mapping>
      <from>DateTime   </from>
      <to>DateTime   </to>
    </type-mapping>
   </java-to-sql-type-conversions>
  <ping-command>   </ping-command>
</jdbc-type-settings>
```

4. Click **Save configuration**.

5. Close and restart your Spotfire server.

#### 4.1.1.12.1.5 Creating a Data Source

After creating the SQream data source template, you can create a data source.

**To create a data source:**

1. Launch the TIBCO Spotfire application.

2. From the **Tools** menu, select **Information Designer**.

   The **Information Designer** window is displayed.

3. From the **New** menu, click **Data Source**.

   The **Data Source** tab is displayed.

4. Provide the following information:

   - **Name** - define a unique name.

- **Type** - use the same type template name you used while configuring your template. See **Step 3** in *Creating the SQream Data Source Template*.
- **Connection URL** - use the standard JDBC connection string, `<ip>:<port>/database`.
- **No. of connections** - define a number between **1** and **100**. SQream recommends setting your number of connections to **100**.
- **Username and Password** - define your SQream username and password.

#### 4.1.1.12.1.6 Creating an Information Link

After creating a data source, you can create an information link.

**To create an information link**:

1. From the **Tools** menu, select **Information Designer**.

   The **Information Designer** window is displayed.

2. From the **New** menu, click **Information Link**.

   The **Information link** tab is displayed.

3. From the **Elements** tab, select a column type and click **Add**.

   The column type is added to the **Elements** region as a filter.

   Note the following:

   - You can select procedures from the Elements region.
   - You can remove an element by selecting an element and clicking **Remove**.

   **Tip:** If the Elements menu is not displayed, you can display it by clicking the **Elements** tab. You can simultaneously select multiple elements by pressing **Ctrl** and making additional selections, and select a range of elements by holding **Shift** and clicking two elements.

4. If the elements you select originate from more than one data source table, specify a **Join path**.
5. *Optional* - In the **Description** region, type the description of the information link.
6. *Optional* - To filter your data, expand the **Filters** section and do the following:
   1. From the **Information Link** region, select the element you added in Step 3 above.
   2. Click **Add**.

      The **Add Column** window is displayed.

3. From the drop-down list, select a column to add a hard filter to and click **OK**.

   The selected column is added to the Filters list.

4. Repeat steps 2 and 3 to add filters to additional columns.

5. For each column, from the **Filter Type** drop-down list, select **range** or **values**.

   **Note:** Filtering by range means entering the upper and lower limits of the desired range. Filtering by values means entering the exact values that you want to include in the returned data, separated by semicolon.

6. In the **Values** field type the desired values separated with semicolons, or set the upper and lower limits in the **Min Value** and **Max Value** fields. Alternatively, you can type `?param_name` in the Values field to use a parameter as the filter for the selected column, where `param_name` is the name used to identify the parameter.

   **Note:** Because limits are inclusive, setting the lower limit to **1000** includes the value **1000** in the data table.

   **Note:** When setting upper and lower limits on **String** type columns, `A` precedes `AA`, and a lone letter precedes words beginning with that latter. For example, `S** precedes **Smith**, indicating that the name ``Smith` will not be present when you select names from `D` to `S`. The order of characters is standard ASCII.

For more information on adding filters, see Adding Hard Filters.

7. *Optional* - To add runtime filtering prompts, expand the **Prompts** section and do the following:

   1. Click **Add**.

      The **Add Column** window is displayed.

   2. From the **Select column** list, select a column to add a prompt to and click **OK**.

      The selected column is added to the Prompts list.

   3. Repeat **Step 1** to add prompts to additional columns.

   4. Do the following for each column:

      - Make a selection from the **Prompt Type** drop-down list.
      - Select or clear **Mandatory**.
      - *Optional* - Set your **Max Selections**.

   For more information on adding prompts, see Adding Prompts.

8. *Optional* - Expand the **Conditioning** section and specify one of the following conditions:

   - None
   - Distinct
   - Pivot

   Note that you can edit the Pivot conditioning by selecting **Pivot** and clicking **Edit**.

9. *Optional* - Expand the **Parameters** section and define your parameters.

10. *Optional* - Expand the **Properties** section and define your properties.

11. *Optional* - Expand the **Caching** section and enable or disable whether your information link can be cached.

12. Click **Save**.

    The **Save As** window is displayed.

13. In the tree, select where you want to save the information link.

14. In the **Name** field, type a name and description for the information link.

15. Click **Save**.

    The new information link is added to the library and can be accessed by other users.

**Tip:** You can test the information link directly by clicking **Open Data**. You can also view and edit the SQL belonging to the information link by clicking **SQL**.

For more information on the Information Link attributes, see Information Link Tab.

#### 4.1.1.12.1.7 Troubleshooting

The **Troubleshooting** section describes the following scenarios:

- *The JDBC Driver does not Support Boolean, Decimal, or Numeric Types*
- *Information Services do not Support Live Queries*

#### 4.1.1.12.1.8 The JDBC Driver does not Support Boolean, Decimal, or Numeric Types

When attempting to load data, the the Boolean, Decimal, or Numeric column types are not supported and generate the following error:

```
Failed to execute query: Unsupported JDBC data type in query result: Bool (HRESULT:␣
↪80131500)
```

The error above is resolved by casting the columns as follows:

- `Bool` columns to `INT`.
- `Decimal` and `Numeric` columns to `REAL`.

For more information, see the following:

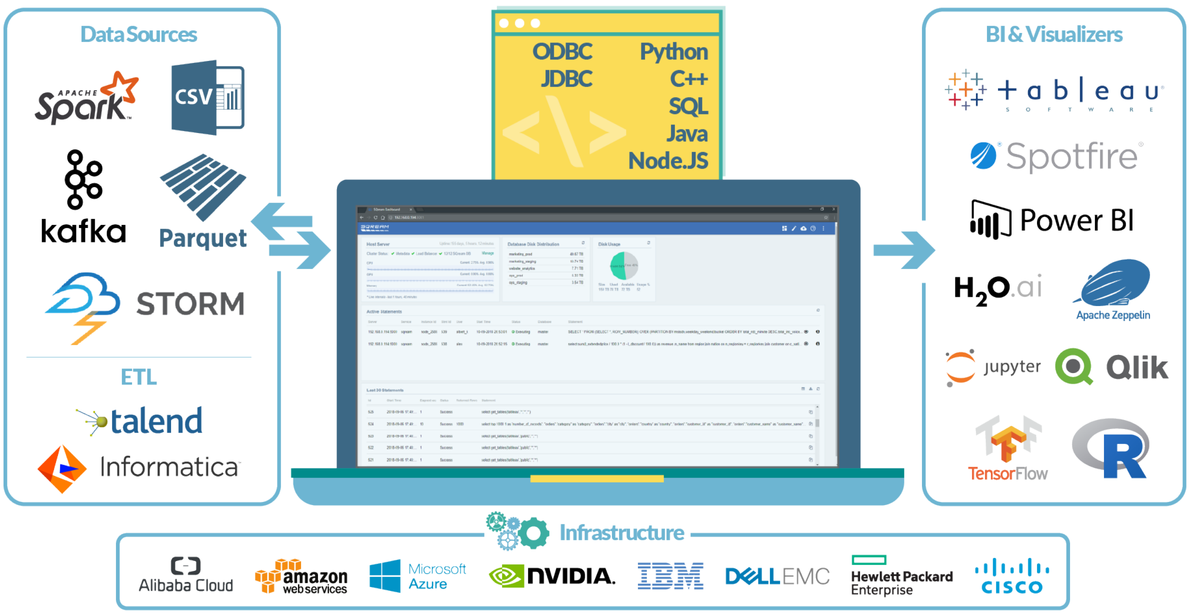
- **Resolving this error** - Details on Change Data Types.

- **Supported data types** - *Data Types*.

#### 4.1.1.12.1.9 Information Services do not Support Live Queries

TIBCO Spotfire data connectors support live queries, but no APIs currently exist for creating custom data connectors. This is resolved by creating a customized SQream adapter using TIBCO's **Data Virtualization (TDV)** or the **Spotfire Advanced Services (ADS)**. These can be used from the built-in TDV connector to enable live queries.

This resolution applies to JDBC and ODBC drivers.




## 4.2 Client Drivers for 2022.1

The guides on this page describe how to use the Sqream DB client drivers and client applications with SQream.

### 4.2.1 Client Driver Downloads

#### 4.2.1.1 All Operating Systems

The following are applicable to all operating systems:

- **JDBC** - recommended installation via `mvn`:
  - JDBC .jar file - sqream-jdbc-4.5.3 (.jar)
  - JDBC driver
- **Tableau**:
  - Tableau connector - SQream (.taco)
  - Tableau manual installation

- **Power BI**:
  - Power BI PowerQuery connector - SQream (.mez)
  - Power BI manual installation

#### 4.2.1.2 Windows

The following are applicable to Windows:

- **ODBC installer** - SQream Drivers v2020.2.0, with Tableau customizations. Please contact your SSream represe-native for this installer.

  For more information on installing and configuring ODBC on Windows, see *Install and configure ODBC on Windows*.

- **Net driver** - SQream .Net driver v3.0.2

#### 4.2.1.3 Linux

The following are applicable to Linux:

- SQream SQL (x86_64) - sqream-sql-v2020.1.1_stable.x86_64.tar.gz
- Sqream SQL CLI Reference - Interactive command-line SQL client for Intel-based machines

- SQream SQL*(IBM POWER9) - sqream-sql-v2020.1.1_stable.ppc64le.tar.gz
- Sqream SQL CLI Reference - Interactive command-line SQL client for IBM POWER9-based machines

- ODBC Installer - Please contact your SQream representative for this installer.

##### 4.2.1.3.1 JDBC

The SQream JDBC driver lets you connect to SQream using many Java applications and tools. This page describes how to write a Java application using the JDBC interface. The JDBC driver requires Java 1.8 or newer.

The JDBC page includes the following sections:

##### 4.2.1.3.1.1 Installing the JDBC Driver

The **Installing the JDBC Driver** section describes the following:



##### 4.2.1.3.1.2 Prerequisites

The SQream JDBC driver requires Java 1.8 or newer, and SQream recommends using Oracle Java or OpenJDK.:

- **Oracle Java** - Download and install Java 8 from Oracle for your platform.
- **OpenJDK** - Install OpenJDK
- **Windows** - SQream recommends installing Zulu 8

##### 4.2.1.3.1.3 Getting the JAR file

SQream provides the JDBC driver as a zipped JAR file, available for download from the *client drivers download page*. This JAR file can be integrated into your Java-based applications or projects.

##### 4.2.1.3.1.4 Extracting the ZIP Archive

Run the following command to extract the JAR file from the ZIP archive:

```
$ unzip sqream-jdbc-4.3.0.zip
```

##### 4.2.1.3.1.5 Setting Up the Class Path

To use the driver, you must include the JAR named `sqream-jdbc-<version>.jar` in the class path, either by inserting it in the `CLASSPATH` environment variable, or by using flags on the relevant Java command line.

For example, if the JDBC driver has been unzipped to `/home/sqream/sqream-jdbc-4.3.0.jar`, the following command is used to run application:

```
$ export CLASSPATH=/home/sqream/sqream-jdbc-4.3.0.jar:$CLASSPATH
$ java my_java_app
```

Alternatively, you can pass `-classpath` to the Java executable file:

```
$ java -classpath .:/home/sqream/sqream-jdbc-4.3.0.jar my_java_app
```

#### 4.2.1.3.1.6 Connecting to SQream Using a JDBC Application

You can connect to SQream using one of the following JDBC applications:

- *Driver Class*
- *Connection String*
- *Sample Java Program*

#### 4.2.1.3.1.7 Driver Class

Use `com.sqream.jdbc.SQDriver` as the driver class in the JDBC application.

#### 4.2.1.3.1.8 Connection String

JDBC drivers rely on a connection string.

The following is the syntax for SQream:

```
jdbc:Sqream://<host and port>/<database name>;user=<username>;password=<password>
↪sqream;[<optional parameters>; ...]
```

#### 4.2.1.3.1.9 Connection Parameters

The following table shows the connection string parameters:

| Item | State | Default | Description |
|---|---|---|---|
| `<host and port>` | Mandatory | None | Hostname and port of the SQream DB worker. For example, `127.0.0.1:5000`, `sqream.mynetwork.co:3108` |
| `<database name>` | Mandatory | None | Database name to connect to. For example, `master` |
| `username=<username>` | Mandatory | None | Username of a role to use for connection. For example, `username=rhendricks` |
| `password=<password>` | Mandatory | None | Specifies the password of the selected role. For example, `password=Tr0ub4dor&3` |
| `service=<service>` | Optional | `sqream` | Specifices service queue to use. For example, `service=etl` |
| `<ssl>` | Optional | `false` | Specifies SSL for this connection. For example, `ssl=true` |
| `<cluster>` | Optional | `true` | Connect via load balancer (use only if exists, and check port). |
| `<fetchSize>` | Optional | `true` | Enables on-demand loading, and defines double buffer size for result. The `fetchSize` parameter is rounded according to chunk size. For example, `fetchSize=1` loads one row and is rounded to one chunk. If the fetchSize is 100,600, a chunk size of 100,000 loads, and is rounded to, two chunks. |
| `<insertBuffer>` | Optional | `true` | Defines the bytes size for inserting a buffer before flushing data to the server. Clients running a parameterized insert (network insert) can define the amount of data to collect before flushing the buffer. |
| `<loggerLevel>` | Optional | `true` | Defines the logger level as either `debug` or `trace`. |
| `<logFile>` | Optional | `true` | Enables the file appender and defines the file name. The file name can be set as either the file name or the file path. |

#### 4.2.1.3.1.10 Connection String Examples

The following is an example of a SQream cluster with load balancer and no service queues (with SSL):

```
jdbc:Sqream://sqream.mynetwork.co:3108/master;user=rhendricks;password=Tr0ub4dor&3;
↪ssl=true;cluster=true
```

The following is a minimal example for a local standalone SQream database:

```
jdbc:Sqream://127.0.0.1:5000/master;user=rhendricks;password=Tr0ub4dor&3
```

The following is an example of a SQream cluster with load balancer and a specific service queue named `etl`, to the database named `raviga`

```
jdbc:Sqream://sqream.mynetwork.co:3108/raviga;user=rhendricks;password=Tr0ub4dor&3;
↪cluster=true;service=etl
```

##### 4.2.1.3.1.11 Sample Java Program

You can download the `JDBC Application Sample File` below by right-clicking and saving it to your computer.

Listing 1: JDBC Application Sample

```java
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;

import java.io.IOException;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.sql.SQLException;



public class SampleTest {

    // Replace with your connection string
    static final String url = "jdbc:Sqream://sqream.mynetwork.co:3108/master;user=rhendricks;password=Tr0ub4dor&3;ssl=true;cluster=true";

    // Allocate objects for result set and metadata
    Connection conn   = null;
    Statement  stmt  = null;
    ResultSet  rs  = null;
    DatabaseMetaData  dbmeta  = null;

    int  res  =  0;

    public  void  testJDBC()  throws  SQLException,  IOException  {

        // Create a connection
        conn  =  DriverManager.getConnection(url,"rhendricks","Tr0ub4dor&3");

        // Create a table with a single integer column
        String sql  =  "CREATE TABLE test (x INT)";
        stmt = conn.createStatement();  // Prepare the statement
        stmt.execute(sql); // Execute the statement
        stmt.close(); // Close the statement handle

        // Insert some values into the newly created table
        sql = "INSERT INTO test VALUES (5),(6)";
        stmt = conn.createStatement();
        stmt.execute(sql);
        stmt.close();

        // Get values from the table
        sql = "SELECT * FROM test";
        stmt = conn.createStatement();
        rs  = stmt.executeQuery(sql);
        // Fetch all results one-by-one
        while(rs.next()) {
            res = rs.getInt(1);
```

(continues on next page)

(continued from previous page)

```
            System.out.println(res); // Print results to screen
        }
        rs.close(); // Close the result set
        stmt.close(); // Close the statement handle
    }


    public static void main(String[] args) throws SQLException, ␣
↪KeyManagementException, NoSuchAlgorithmException, IOException, ␣
↪ClassNotFoundException{

        // Load SQream DB JDBC driver
        Class.forName("com.sqream.jdbc.SQDriver");

        // Create test object and run
        SampleTest test = new SampleTest();
        test.testJDBC();
    }
}
```

#### 4.2.1.3.2 ODBC

##### 4.2.1.3.2.1 Install and Configure ODBC on Windows

The ODBC driver for Windows is provided as a self-contained installer.

This tutorial shows you how to install and configure ODBC on Windows.

**In this topic:**

#### 4.2.1.3.2.2 Installing the ODBC Driver

#### 4.2.1.3.2.3 Prerequisites

#### 4.2.1.3.2.4 Visual Studio 2015 Redistributables

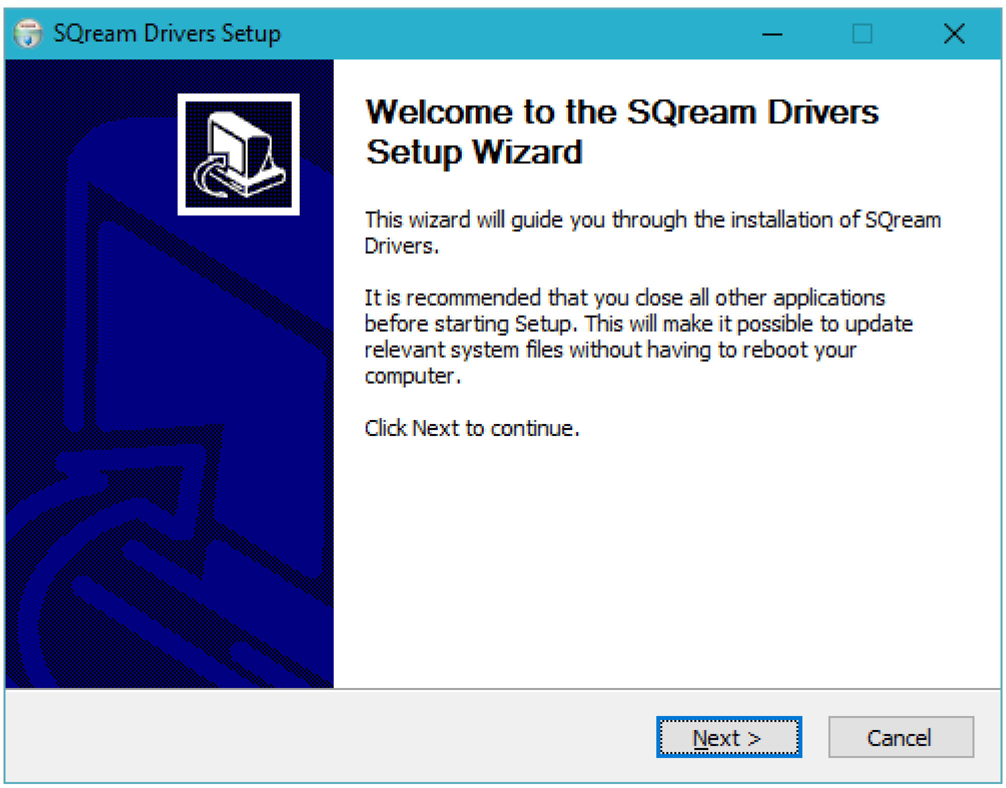
To install the ODBC driver you must first install Microsoft's **Visual C++ Redistributable for Visual Studio 2015**. To install Visual C++ Redistributable for Visual Studio 2015, see the Install Instructions.

#### 4.2.1.3.2.5 Administrator Privileges

The SQream DB ODBC driver requires administrator privileges on your computer to add the DSNs (data source names).

#### 4.2.1.3.2.6 1. Run the Windows installer

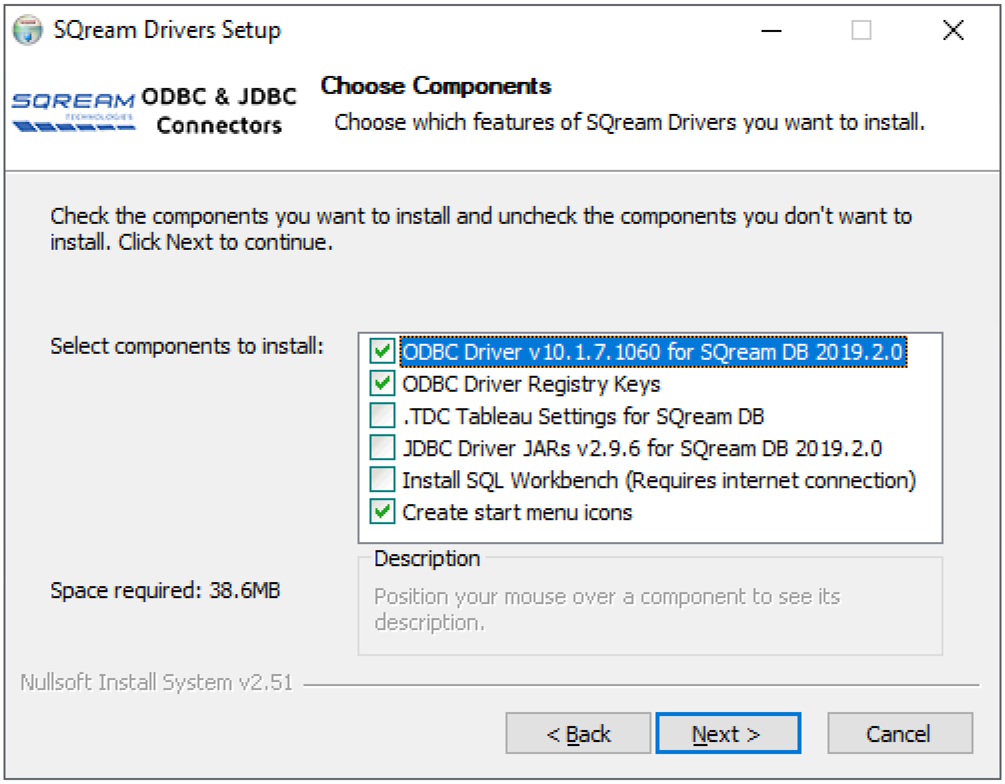
Install the driver by following the on-screen instructions in the easy-to-follow installer.




**Note:** The installer will install the driver in `C:\Program Files\SQream Technologies\ODBC Driver` by default. This path is changable during the installation.

#### 4.2.1.3.2.7 2. Selecting Components

The installer includes additional components, like JDBC and Tableau customizations.



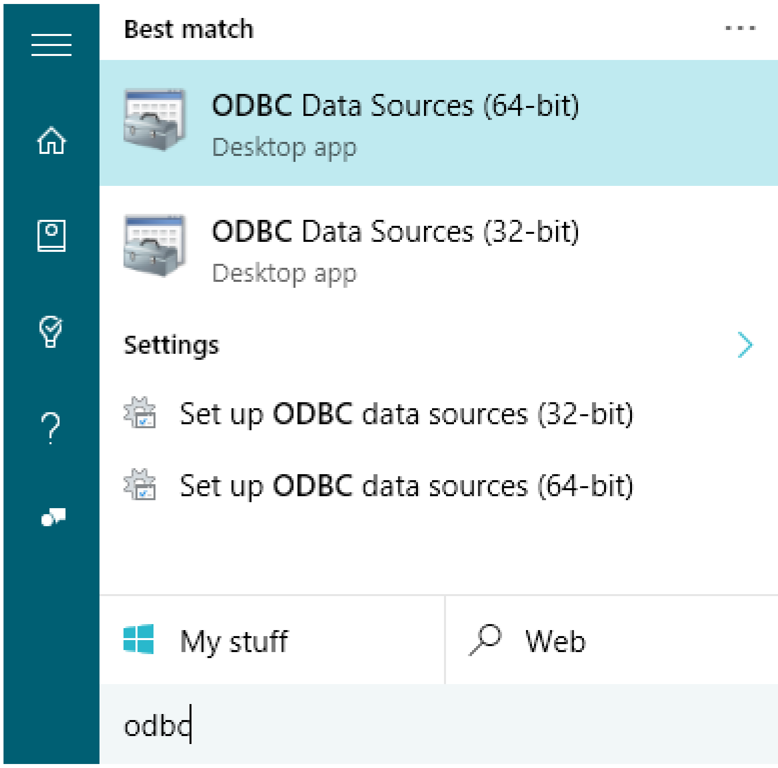

You can deselect items you don't want to install, but the items named **ODBC Driver DLL** and **ODBC Driver Registry Keys** must remain selected for a complete installation of the ODBC driver.

Once the installer finishes, you will be ready to configure the DSN for connection.

#### 4.2.1.3.2.8 3. Configuring the ODBC Driver DSN

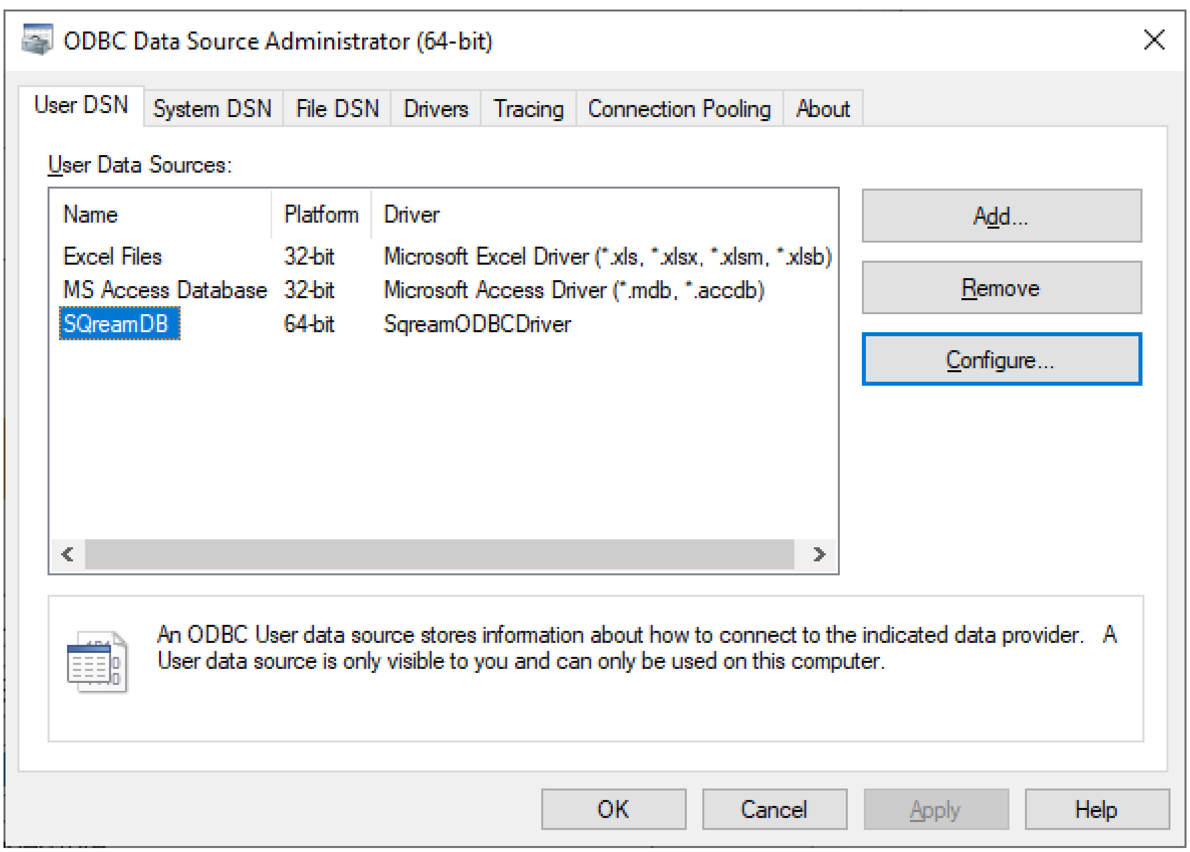
ODBC driver configurations are done via DSNs. Each DSN represents one SQream DB database.

1. Open up the Windows menu by clicking the Windows button on your keyboard (⊞ `Win`) or pressing the Windows button with your mouse.
2. Type **ODBC** and select **ODBC Data Sources (64-bit)**. Click the item to open up the setup window.

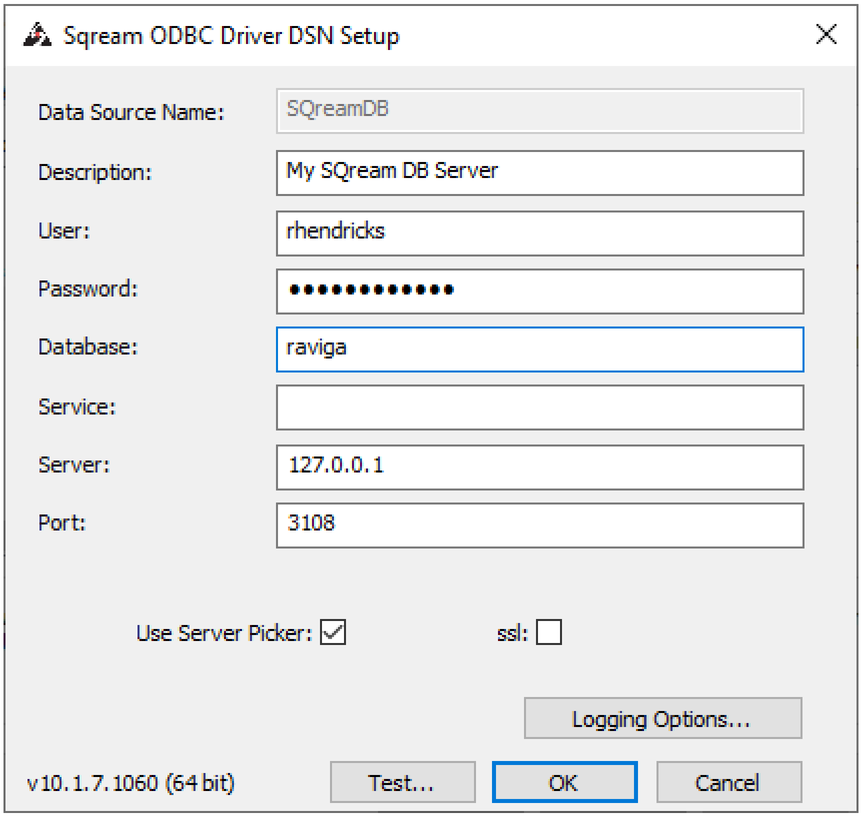



3. The installer has created a sample User DSN named **SQreamDB**

   You can modify this DSN, or create a new one (*Add* ‣ *SQream ODBC Driver* ‣ *Next*)

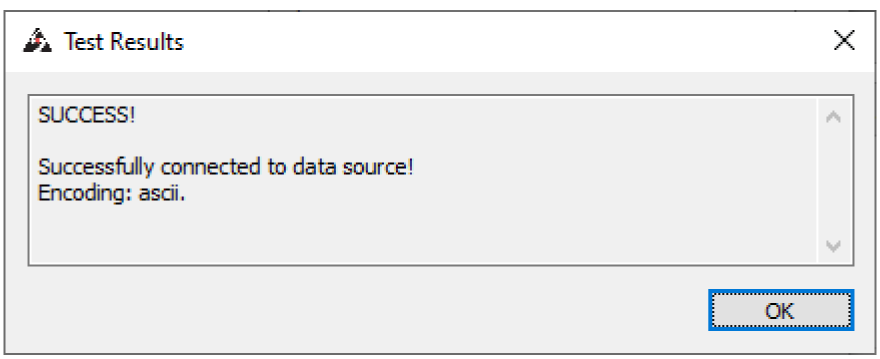


4. Enter your connection parameters. See the reference below for a description of the parameters.




5. When completed, save the DSN by selecting *OK*

**Tip:** Test the connection by clicking *Test* before saving. A successful test looks like this:




1. You can now use this DSN in ODBC applications like Tableau.

#### 4.2.1.3.2.9 Connection Parameters

| Item | Description |
|---|---|
| Data Source Name | An easily recognizable name that you'll use to reference this DSN. Once you set this, it can not be changed. |
| Description | A description of this DSN for your convenience. You can leave this blank. |
| User | Username of a role to use for connection. For example, `rhendricks` |
| Password | Specifies the password of the selected role. For example, `Tr0ub4dor&3` |
| Database | Specifies the database name to connect to. For example, `master` |
| Service | Specifices *service queue* to use. For example, `etl`. Leave blank for default service `sqream`. |
| Server | Hostname of the SQream DB worker. For example, `127.0.0.1` or `sqream.mynetwork.co` |
| Port | TCP port of the SQream DB worker. For example, `5000` or `3108` |
| User server picker | Connect via load balancer (use only if exists, and check port) |
| SSL | Specifies SSL for this connection |
| Logging options | Use this screen to alter logging options when tracing the ODBC connection for possible connection issues. |

#### 4.2.1.3.2.10 Troubleshooting

#### 4.2.1.3.2.11 Solving "Code 126" ODBC errors

After installing the ODBC driver, you may experience the following error:

```
The setup routines for the SQreamDriver64 ODBC driver could not be loaded due to
↪system error
code 126: The specified module could not be found.
(c:\Program Files\SQream Technologies\ODBC Driver\sqreamOdbc64.dll)
```

This is an issue with the Visual Studio Redistributable packages. Verify you've correctly installed them, as described in the *Visual Studio 2015 Redistributables* section above.

#### 4.2.1.3.2.12 Install and configure ODBC on Linux

The ODBC driver for Windows is provided as a shared library.

This tutorial shows how to install and configure ODBC on Linux.

**In this topic:**

- *ODBC DSN Parameters*

#### 4.2.1.3.2.13 Prerequisites

#### 4.2.1.3.2.14 unixODBC

The ODBC driver requires a driver manager to manage the DSNs. SQream DB's driver is built for unixODBC.

Verify unixODBC is installed by running:

```
$ odbcinst -j
unixODBC 2.3.4
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
USER DATA SOURCES..: /home/rhendricks/.odbc.ini
SQLULEN Size.......: 8
SQLLEN Size........: 8
SQLSETPOSIROW Size.: 8
```

Take note of the location of `.odbc.ini` and `.odbcinst.ini`. In this case, `/etc`. If `odbcinst` is not installed, follow the instructions for your platform below:

**Install unixODBC on:**

- *Install unixODBC on RHEL 7 / CentOS 7*
- *Install unixODBC on Ubuntu*

#### 4.2.1.3.2.15 Install unixODBC on RHEL 7 / CentOS 7

```
$ yum install -y unixODBC unixODBC-devel
```

#### 4.2.1.3.2.16 Install unixODBC on Ubuntu

```
$ sudo apt-get install unixodbc unixodbc-dev
```

#### 4.2.1.3.2.17 Install the ODBC driver with a script

Use this method if you have never used ODBC on your machine before. If you have existing DSNs, see the manual install process below.

1. Unpack the tarball Copy the downloaded file to any directory, and untar it to a new directory:

   ```
   $ mkdir -p sqream_odbc64
   $ tar xf sqream_2019.2.1_odbc_3.0.0_x86_64_linux.tar.gz -C sqream_odbc64
   ```

2. Run the first-time installer. The installer will create an editable DSN.

```
$ cd sqream_odbc64
./odbc_install.sh --install
```

3. Edit the DSN created by editing `/etc/.odbc.ini`. See the parameter explanation in the section *ODBC DSN Parameters*.

#### 4.2.1.3.2.18 Install the ODBC driver manually

Use this method when you have existing ODBC DSNs on your machine.

1. Unpack the tarball Copy the file you downloaded to the directory where you want to install it, and untar it:

   ```
   $ tar xf sqream_2019.2.1_odbc_3.0.0_x86_64_linux.tar.gz -C sqream_odbc64
   ```

   Take note of the directory where the driver was unpacked. For example, `/home/rhendricks/sqream_odbc64`

2. Locate the `.odbc.ini` and `.odbcinst.ini` files, using `odbcinst -j`.

   1. In `.odbcinst.ini`, add the following lines to register the driver (change the highlighted paths to match your specific driver):

      ```
      [ODBC Drivers]
      SqreamODBCDriver=Installed

      [SqreamODBCDriver]
      Description=Driver DSII SqreamODBC 64bit
      Driver=/home/rhendricks/sqream_odbc64/sqream_odbc64.so
      Setup=/home/rhendricks/sqream_odbc64/sqream_odbc64.so
      APILevel=1
      ConnectFunctions=YYY
      DriverODBCVer=03.80
      SQLLevel=1
      IconvEncoding=UCS-4LE
      ```

   2. In `.odbc.ini`, add the following lines to configure the DSN (change the highlighted parameters to match your installation):

      ```
      [ODBC Data Sources]
      MyTest=SqreamODBCDriver

      [MyTest]
      Description=64-bit Sqream ODBC
      Driver=/home/rhendricks/sqream_odbc64/sqream_odbc64.so
      Server="127.0.0.1"
      Port="5000"
      Database="raviga"
      Service=""
      User="rhendricks"
      Password="Tr0ub4dor&3"
      Cluster=false
      Ssl=false
      ```

      Parameters are in the form of `parameter = value`. For details about the parameters that can be set for each DSN, see the section *ODBC DSN Parameters*.

3. Create a file called `.sqream_odbc.ini` for managing the driver settings and logging. This file should be created alongside the other files, and add the following lines (change the highlighted parameters to match your installation):

```
# Note that this default DriverManagerEncoding of UTF-32 is for iODBC.␣
↪unixODBC uses UTF-16 by default.
# If unixODBC was compiled with -DSQL_WCHART_CONVERT, then UTF-32 is␣
↪the correct value.
# Execute 'odbc_config --cflags' to determine if you need UTF-32 or␣
↪UTF-16 on unixODBC
[Driver]
DriverManagerEncoding=UTF-16
DriverLocale=en-US
ErrorMessagesPath=/home/rhendricks/sqream_odbc64/ErrorMessages
LogLevel=0
LogNamespace=
LogPath=/tmp/
ODBCInstLib=libodbcinst.so
```

#### 4.2.1.3.2.19 Install the driver dependencies

Add the ODBC driver path to `LD_LIBRARY_PATH`:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/rhendricks/sqream_odbc64/lib
```

You can also add this previous command line to your `~/.bashrc` file in order to keep this installation working between reboots without re-entering the command manually

#### 4.2.1.3.2.20 Testing the connection

Test the driver using `isql`.

If the DSN created is called `MyTest` as the example, run isql in this format:

```
$ isql MyTest
```

#### 4.2.1.3.2.21 ODBC DSN Parameters

| Item | Default | Description |
|---|---|---|
| Data Source Name | None | An easily recognizable name that you'll use to reference this DSN. |
| Description | None | A description of this DSN for your convenience. This field can be left blank |
| User | None | Username of a role to use for connection. For example, `User="rhendricks"` |
| Password | None | Specifies the password of the selected role. For example, `User="Tr0ub4dor&3"` |
| Database | None | Specifies the database name to connect to. For example, `Database="master"` |
| Service | `sqream` | Specifices *service queue* to use. For example, `Service="etl"`. Leave blank (`Service=""`) for default service `sqream`. |
| Server | None | Hostname of the SQream DB worker. For example, `Server="127.0.0.1"` or `Server="sqream.mynetwork.co"` |
| Port | None | TCP port of the SQream DB worker. For example, `Port="5000"` or `Port="3108"` for the load balancer |
| Cluster | `false` | Connect via load balancer (use only if exists, and check port). For example, `Cluster=true` |
| Ssl | `false` | Specifies SSL for this connection. For example, `Ssl=true` |
| DriverManagerEncoding | `UTF-16` | Depending on how unixODBC is installed, you may need to change this to `UTF-32`. |
| ErrorMessagesPath | None | Location where the driver was installed. For example, `ErrorMessagePath=/home/rhendricks/sqream_odbc64/ErrorMessages`. |
| LogLevel | 0 | Set to 0-6 for logging. Use this setting when instructed to by SQream Support. For example, `LogLevel=1`<br><br>• 0 = Disable tracing<br>• 1 =<br>• 2 = Error tracing<br>• 3 =<br>Info tracing<br>• 5 = Debug tracing |

SQream has an ODBC driver to connect to SQream DB. This tutorial shows how to install the ODBC driver for Linux or Windows for use with applications like Tableau, PHP, and others that use ODBC.

| Platform | Versions supported |
|---|---|
| Windows | • Windows 7 (64 bit)<br>• Windows 8 (64 bit)<br>• Windows 10 (64 bit)<br>• Windows Server 2008 R2 (64 bit)<br>• Windows Server 2012<br>• Windows Server 2016<br>• Windows Server 2019 |
| Linux | • Red Hat Enterprise Linux (RHEL) 7<br>• CentOS 7<br>• Ubuntu 16.04<br>• Ubuntu 18.04 |

Other distributions may also work, but are not officially supported by SQream.

**In this topic:**



#### 4.2.1.3.2.22 Downloading the ODBC driver

The SQream DB ODBC driver is distributed by your SQream account manager. Before contacting your account manager, verify which platform the ODBC driver will be used on. Go to SQream Support or contact your SQream account manager to get the driver.

The driver is provided as an executable installer for Windows, or a compressed tarball for Linux platforms. After downloading the driver, follow the relevant instructions to install and configure the driver for your platform:

#### 4.2.1.3.2.23 Install and configure the ODBC driver

Continue based on your platform:

- *Install and Configure ODBC on Windows*
- *Install and configure ODBC on Linux*

### Need help?

If you couldn't find what you're looking for, we're always happy to help. Visit SQream's support portal for additional support.

### Looking for older drivers?

If you're looking for an older version of SQream DB drivers, versions 1.10 through 2019.2.1 are available at https://sqream.com/product/client-drivers/.

If you need a tool that SQream does not support, contact SQream Support or your SQream account manager for more information.

CHAPTER

FIVE

# EXTERNAL STORAGE PLATFORMS

SQream supports the following external storage platforms:

## 5.1 Inserting Data Using Amazon S3

SQream uses a native S3 connector for inserting data. The `s3://` URI specifies an external file path on an S3 bucket. File names may contain wildcard characters, and the files can be in CSV or columnar format, such as Parquet and ORC.

The **Amazon S3** describes the following topics:



### 5.1.1 S3 Configuration

Any database host with access to S3 endpoints can access S3 without any configuration. To read files from an S3 bucket, the database must have listable files.

### 5.1.2 S3 URI Format

With S3, specify a location for a file (or files) when using copy_from or external_tables.

The following is an example of the general S3 syntax:

```
s3://bucket_name/path
```

### 5.1.3 Authentication

SQream supports `AWS ID` and `AWS SECRET` authentication. These should be specified when executing a statement.

### 5.1.4 Examples

Use a foreign table to stage data from S3 before loading from CSV, Parquet, or ORC files.

The **Examples** section includes the following examples:



#### 5.1.4.1 Planning for Data Staging

The examples in this section are based on a CSV file, as shown in the following table:

The file is stored on Amazon S3, and this bucket is public and listable. To create a matching `CREATE FOREIGN TABLE` statement you can make note of the file structure.

#### 5.1.4.2 Creating a Foreign Table

Based on the source file's structure, you can create a foreign table with the appropriate structure, and point it to your file as shown in the following example:

```
CREATE FOREIGN TABLE nba
(
   Name varchar(40),
   Team varchar(40),
   Number tinyint,
   Position varchar(2),
   Age tinyint,
   Height varchar(4),
   Weight real,
   College varchar(40),
   Salary float
 )
 WRAPPER csv_fdw
```

(continues on next page)

(continued from previous page)

```
OPTIONS
  (
     LOCATION = 's3://sqream-demo-data/nba_players.csv',
     RECORD_DELIMITER = '\r\n' -- DOS delimited file
  )
;
```

In the example above the file format is CSV, and it is stored as an S3 object. If the path is on HDFS, you must change the URI accordingly. Note that the record delimiter is a DOS newline (`\r\n`).

For more information, see the following:

- **Creating a foreign table** - see create a foreign table.
- **Using SQream in an HDFS environment** - see *Using SQream in an HDFS Environment*.

### 5.1.4.3 Querying Foreign Tables

The following shows the data in the foreign table:

```
t=> SELECT * FROM nba LIMIT 10;
name          | team           | number | position | age | height | weight | college ␣
↪          | salary
--------------+----------------+--------+----------+-----+--------+--------+----------
↪----------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas    ␣
↪          |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |␣
↪Marquette         |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston␣
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia␣
↪State      |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |          ␣
↪          |  5000000
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |          ␣
↪          | 12000000
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU      ␣
↪          |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga  ␣
↪          |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |␣
↪Louisville        |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma␣
↪State     |  3431040
```

#### 5.1.4.4 Bulk Loading a File from a Public S3 Bucket

The `COPY FROM` command can also be used to load data without staging it first.

**Note:** The bucket must be publicly available and objects can be listed.

The following is an example of bulk loading a file from a public S3 bucket:

```
COPY nba FROM 's3://sqream-demo-data/nba.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n';
```

For more information on the `COPY FROM` command, see copy_from.

#### 5.1.4.5 Loading Files from an Authenticated S3 Bucket

The following is an example of loading fles from an authenticated S3 bucket:

```
COPY nba FROM 's3://secret-bucket/*.csv' WITH OFFSET 2 RECORD DELIMITER '\r\n'
AWS_ID '12345678'
AWS_SECRET 'super_secretive_secret';
```

## 5.2 Using SQream in an HDFS Environment

### 5.2.1 Configuring an HDFS Environment for the User sqream

This section describes how to configure an HDFS environment for the user **sqream** and is only relevant for users with an HDFS environment.

**To configure an HDFS environment for the user sqream:**

1. Open your **bash_profile** configuration file for editing:

```
$ vim /home/sqream/.bash_profile
```

2. Verify that the edits have been made:

```
source /home/sqream/.bash_profile
```

3. Check if you can access Hadoop from your machine:

```
$ hadoop fs -ls hdfs://<hadoop server name or ip>:8020/
```

4. Verify that an HDFS environment exists for SQream services:

```
$ ls -l /etc/sqream/sqream_env.sh
```

5. If an HDFS environment does not exist for SQream services, create one (sqream_env.sh):

```
$ #!/bin/bash

$ SQREAM_HOME=/usr/local/sqream
$ export SQREAM_HOME

```

(continues on next page)

(continued from previous page)

```
$ export JAVA_HOME=${SQREAM_HOME}/hdfs/jdk
$ export HADOOP_INSTALL=${SQREAM_HOME}/hdfs/hadoop
$ export CLASSPATH=`${HADOOP_INSTALL}/bin/hadoop classpath --glob`
$ export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_INSTALL}/lib/native
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${SQREAM_HOME}/lib:$HADOOP_COMMON_LIB_
↪NATIVE_DIR


$ PATH=$PATH:$HOME/.local/bin:$HOME/bin:${SQREAM_HOME}/bin/:${JAVA_HOME}/bin:
↪$HADOOP_INSTALL/bin
$ export PATH
```

*Back to top*

### 5.2.2 Authenticating Hadoop Servers that Require Kerberos

If your Hadoop server requires Kerberos authentication, do the following:

1. Create a principal for the user **sqream**.

```
$ kadmin -p root/admin@SQ.COM
$ addprinc sqream@SQ.COM
```

2. If you do not know yor Kerberos root credentials, connect to the Kerberos server as a root user with ssh and run **kadmin.local**:

```
$ kadmin.local
```

Running **kadmin.local** does not require a password.

3. If a password is not required, change your password to **sqream@SQ.COM**.

```
$ change_password sqream@SQ.COM
```

4. Connect to the hadoop name node using ssh:

```
$ cd /var/run/cloudera-scm-agent/process
```

5. Check the most recently modified content of the directory above:

```
$ ls -lrt
```

6. Look for a recently updated folder containing the text **hdfs**.

The following is an example of the correct folder name:

```
cd <number>-hdfs-<something>
```

This folder should contain a file named **hdfs.keytab** or another similar .keytab file.

7. Copy the .keytab file to user **sqream's** Home directory on the remote machines that you are planning to use Hadoop on.

8. Copy the following files to the **sqream sqream@server:<sqream folder>/hdfs/hadoop/etc/hadoop:** directory:

   - core-site.xml
   - hdfs-site.xml

9. Connect to the sqream server and verify that the .keytab file's owner is a user sqream and is granted the correct permissions:

```
$ sudo chown sqream:sqream /home/sqream/hdfs.keytab
$ sudo chmod 600 /home/sqream/hdfs.keytab
```

10. Log into the sqream server.
11. Log in as the user **sqream**.
12. Navigate to the Home directory and check the name of a Kerberos principal represented by the following .keytab file:

```
$ klist -kt hdfs.keytab
```

The following is an example of the correct output:

```
$ sqream@Host-121 ~ $ klist -kt hdfs.keytab
$ Keytab name: FILE:hdfs.keytab
$ KVNO Timestamp           Principal
$ ---- ------------------- ------------------------------------------------------
↪----
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 HTTP/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
$    5 09/15/2020 18:03:05 hdfs/nn1@SQ.COM
```

13. Verify that the hdfs service named **hdfs/nn1@SQ.COM** is shown in the generated output above.
14. Run the following:

```
$ kinit -kt hdfs.keytab hdfs/nn1@SQ.COM
```

15. Check the output:

```
$ klist
```

The following is an example of the correct output:

```
$ Ticket cache: FILE:/tmp/krb5cc_1000
$ Default principal: sqream@SQ.COM
$
$ Valid starting       Expires              Service principal
$ 09/16/2020 13:44:18  09/17/2020 13:44:18  krbtgt/SQ.COM@SQ.COM
```

16. List the files located at the defined server name or IP address:

```
$ hadoop fs -ls hdfs://<hadoop server name or ip>:8020/
```

17. Do one of the following:

    - If the list below is output, continue with Step 16.
    - If the list is not output, verify that your environment has been set up correctly.

If any of the following are empty, verify that you followed *Step 6* in the **Configuring an HDFS Environment for the User sqream** section above correctly:

```
$ echo $JAVA_HOME
$ echo $SQREAM_HOME
$ echo $CLASSPATH
$ echo $HADOOP_COMMON_LIB_NATIVE_DIR
$ echo $LD_LIBRARY_PATH
$ echo $PATH
```

18. Verify that you copied the correct keytab file.

19. Review this procedure to verify that you have followed each step.

*Back to top*

For more information, see the following:

- *Foreign Tables*
- copy_from
- copy_to

CHAPTER

SIX

# LOADING AND UNLOADING DATA

The **Loading Data** section describes concepts and operations related to importing data into your SQream database:

- Overview of loading data - Describes best practices and considerations for loading data into SQream from a variety of sources and locations.
- Alternatives to loading data (foreign tables) - Useful for running queries directly on external data without importing into your SQream database.
- Supported data types - Overview of supported data types, including descriptions, examples, and relevant aliases.
- Ingesting data from external sources - List of data ingestion sources that SQream supports.
- Inserting data from external tables - Inserts one or more rows into a table.
- Ingesting data from third party client platforms - Gives you direct access to a variety of drivers, connectors, tools, vizualisers, and utilities..
- Using the COPY FROM statement - Used for loading data from files located on a filesystem into SQream tables.
- Importing data using Studio - SQream's web-based client providing users with all functionality available from the command line in an intuitive and easy-to-use format.
- Loading data using Amazon S3 - Used for loading data from Amazon S3.
- Troubleshooting - Describes troubleshooting solutions related to importing data from the following:
  - SAS Viya
  - Tableau

The **Unloading Data** section describes concepts and operations related to exporting data from your SQream database:

- Overview of unloading data - Describes best practices and considerations for unloading data from SQream to a variety of sources and locations.
- The COPY TO statement - Used for unloading data from a SQream database table or query to a file on a filesystem.

CHAPTER

SEVEN

# FEATURE GUIDES

The **Feature Guides** section describes background processes that SQream uses to manage several areas of operation, such as data ingestion, load balancing, and access control.

This section describes the following features:

## 7.1 Query Healer

The **Query Healer** page describes the following:



### 7.1.1 Overview

The **Query Healer** periodically examines the progress of running statements, creating a log entry for all statements exceeding the `healerMaxInactivityHours` flag setting. The default setting of the `healerMaxInactivityHours` is five hours. The `healerMaxInactivityHours` log frequency is calculated as 5% of the flag setting. When set to to five hours (the default setting), the Query Healer triggers an examination every 15 minutes.

The following is an example of a log record for a query stuck in the query detection phase for more than five hours:

```
|INFO|0x00007f9a497fe700:Healer|192.168.4.65|5001|-1|master|sqream|-1|sqream|0|
↪"[ERROR]|cpp/SqrmRT/healer.cpp:140 |"Stuck query found. Statement ID: 72, Last␣
↪chunk producer updated: 1.
```

Once you identify the stuck worker, you can execute the `shutdown_server` utility function from this specific worker, as described in the next section.

### 7.1.2 Activating a Graceful Shutdown

You can activate a graceful shutdown if your log entry says `Stuck query found`, as shown in the example above. You can do this by setting the **shutdown_server** utility function to `select shutdown_server();`.

**To activte a graceful shutdown:**

1. Locate the IP and the Port of the stuck worker from the logs.

   **Note:** The log in the previous section identifies the IP **(192.168.4.65)** and port **(5001)** referring to the stuck query.

2. From the machine of the stuck query (IP: **192.168.4.65**, port: **5001**), connect to SQream SQL client:

```
./sqream sql --port=$STUCK_WORKER_IP --username=$SQREAM_USER --password=$SQREAM_
↪PASSWORD databasename=$SQREAM_DATABASE
```

3. Execute `shutdown_server`.

For more information, see the following:

- Activating the shutdown_server_command utility function. This page describes all of `shutdown_server` options.

- Configuring the shutdown_server flag.

### 7.1.3 Configuring the Healer

The following **Administration Worker** flags are required to configure the Query Healer:

- is_healer_on - Enables the Query Healer.

- healer_max_inactivity_hours - Defines the threshold for creating a log recording a slow statement. The log includes information about the log memory, CPU and GPU.

## 7.2 Data Encryption

The **Data Encryption** page describes the following:

### 7.2.1 Overview

**Data Encryption** helps protect sensitive data at rest by concealing it from unauthorized users in the event of a breach. This is achieved by scrambling the content into an unreadable format based on encryption and decryption keys. Typically speaking, this data pertains to **PII (Personally Identifiable Information)**, which is sensitive information such as credit card numbers and other information related to an identifiable person.

Users encrypt their data on a column basis by specifying `column_name` in the encryption syntax.

The demand for confidentiality has steadily increased to protect the growing volumes of private data stored on computer systems and transmitted over the internet. To this end, regulatory bodies such as the **General Data Protection Regulation (GDPR)** have produced requirements to standardize and enforce compliance aimed at protecting customer data.

Encryption can be used for the following:

- Creating tables up to three encrypted columns.
- Joining encrypted columns with other tables.
- Selecting data from an encrypted column.

For more information on the encryption syntax, see *Syntax*.

For more information on GDPR compliance requirements, see the GDPR checklist.

## 7.2.2 Encryption Methods

Data exists in one of following states and determines the encryption method:

- *Encrypting Data in Transit*
- *Encrypting Data at Rest*

### 7.2.2.1 Encrypting Data in Transit

**Data in transit** refers to data you use on a regular basis, usually stored on a database and accessed through applications or programs. This data is typically transferred between several physical or remote locations through email or uploading documents to the cloud. This type of data must therefore be protected while **in transit**. SQream encrypts data in transit using SSL when, for example, users insert data files from external repositories over a JDBC or ODBC connection.

For more information, see Use TLS/SSL When Possible.

### 7.2.2.2 Encrypting Data at Rest

**Data at rest** refers to data stored on your hard drive or on the cloud. Because this data can be potentially intercepted **physically**, it requires a form of encryption that protects your data wherever you store it. SQream faciliates encryption by letting you encrypt any columns located in your database that you want to keep private.

## 7.2.3 Data Types

Typically speaking, sensitive pertains to **PII (Personally Identifiable Information)**, which is sensitive information such as credit card numbers and other information related to an identifiable person.

SQream's data encryption feature supports encrypting column-based data belonging to the following data types:

- INT
- BIGINT
- TEXT

For more information on the above data types, see *Supported Data Types*.

### 7.2.4 Syntax

The following is the syntax for encrypting a new table:

```
CREATE TABLE <table name>  (
     <column_name> NOT NULL ENCRYPT,
     <column_name> <type_name> ENCRYPT,
     <column_name> <type_name>,
     <column_name> <type_name> ENCRYPT);
```

The following is an example of encrypting a new table:

```
CREATE TABLE client_name  (
     id BIGINT NOT NULL ENCRYPT,
     first_name TEXT ENCRYPT,
     last_name TEXT,
     salary INT ENCRYPT);
```

**Note:** Because encryption is not associated with any role, users with **Read** or **Insert** permissions can read tables containing encrypted data.

**Warning:** Your performance degradation increases in correlation with the amount of added columns.

You cannot encrypt more than three columns. Attempting to encrypt more than three columns displays the following error message:

```
Error preparing statement: Cannot create a table with more than three encrypted␣
↪columns.
```

## 7.3 Compression

SQream DB uses compression and encoding techniques to optimize query performance and save on disk space.

### 7.3.1 Encoding

Encoding converts data into a common format.

When data is stored in a columnar format, it is often in a common format. This is in contrast with data stored in CSVs for example, where everything is stored in a text format.

Because encoding uses specific data formats and encodings, it increases performance and reduces data size.

SQream DB encodes data in several ways depending on the data type. For example, a date is stored as an integer, with March 1st 1CE as the start. This is a lot more efficient than encoding the date as a string, and offers a wider range than storing it relative to the Unix Epoch.

### 7.3.2 Compression

Compression transforms data into a smaller format without losing accuracy (lossless).

After encoding a set of column values, SQream DB packs the data and compresses it.

Before data can be accessed, SQream DB needs to decompress it.

Depending on the compression scheme, the operations can be performed on the CPU or the GPU. Some users find that GPU compressions perform better for their data.

#### 7.3.2.1 Automatic compression

By default, SQream DB automatically compresses every column (see *Specifying compressions* below for overriding default compressions). This feature is called **automatic adaptive compression** strategy.

When loading data, SQream DB automatically decides on the compression schemes for specific chunks of data by trying several compression schemes and selecting the one that performs best. SQream DB tries to balance more agressive compressions with the time and CPU/GPU time required to compress and decompress the data.

#### 7.3.2.2 Compression strategies

| Compression name | Supported data types | Description | Location |
|---|---|---|---|
| `FLAT` | All types | No compression (forced) | • |
| `DEFAULT` | All types | Automatic scheme selection | • |
| `DICT` | Integer types, dates and timestamps, short texts | Dictionary compression with RLE. For each chunk, SQream DB creates a dictionary of distinct values and stores only their indexes.<br>Works best for integers and texts shorter than 120 characters, with <10% unique values.<br>Useful for storing ENUMs or keys, stock tickers, and dimensions.<br>If the data is optionally sorted, this compression will perform even better. | GPU |
| `P4D` | Integer types, dates and timestamps | Patched frame-of-reference + Delta<br>Based on the delta between consecutive values. Works best for monotonously increasing or decreasing numbers and timestamps | GPU |
| `LZ4` | Text types | Lempel-Ziv general purpose compression, used for texts | CPU |
| `SNAPPY` | Text types | General purpose compression, used for texts | CPU |
| `RLE` | Integer types, dates and timestamps | Run-length encoding. This replaces sequences of values with a single pair. It is best for low cardinality columns that are used to sort data (`ORDER BY`). | GPU |
| `SEQUENCE` | Integer types | Optimized RLE + Delta type for built-in identity columns. | GPU |

#### 7.3.2.3 Specifying compression strategies

When creating a table without any compression specifications, SQream DB defaults to automatic adaptive compression (`"default"`).

However, this can be overriden by specifying a compression strategy when creating a table.

##### 7.3.2.3.1 Explicitly specifying automatic compression

The following two are equivalent:

```
CREATE TABLE t (
   x INT,
   y TEXT(50)
);
```

In this version, the default compression is specified explicitly:

```
CREATE TABLE t (
   x INT CHECK('CS "default"'),
   y TEXT(50) CHECK('CS "default"')
);
```

##### 7.3.2.3.2 Forcing no compression (flat)

In some cases, you may wish to remove compression entirely on some columns, in order to reduce CPU or GPU resource utilization at the expense of increased I/O.

```
CREATE TABLE t (
   x INT NOT NULL CHECK('CS "flat"'), -- This column won't be compressed
   y TEXT(50) -- This column will still be compressed automatically
);
```

##### 7.3.2.3.3 Forcing compressions

In some cases, you may wish to force SQream DB to use a specific compression scheme based on your knowledge of the data.

For example:

```
CREATE TABLE t (
   id BIGINT NOT NULL CHECK('CS "sequence"'),
   y TEXT(110) CHECK('CS "lz4"'), -- General purpose text compression
   z TEXT(80) CHECK('CS "dict"'), -- Low cardinality column

);
```

#### 7.3.2.4 Examining compression effectiveness

Queries to the internal metadata catalog can expose how effective the compression is, as well as what compression schemes were selected.

Here is a sample query we can use to query the catalog:

```
SELECT c.column_name AS "Column",
       cc.compression_type AS "Actual compression",
       AVG(cc.compressed_size) "Compressed",
       AVG(cc.uncompressed_size) "Uncompressed",
       AVG(cc.uncompressed_size::FLOAT/ cc.compressed_size) -1 AS "Compression␣
↪effectiveness",
       MIN(c.compression_strategy) AS "Compression strategy"
 FROM sqream_catalog.chunk_columns cc
   INNER JOIN sqream_catalog.columns c
           ON cc.table_id = c.table_id
          AND cc.database_name = c.database_name
          AND cc.column_id = c.column_id

   WHERE c.table_name = 'some_table'  -- This is the table name which we want to␣
↪inspect

   GROUP BY 1,
            2;
```

Example (subset) from the `ontime` table:

```
stats=> SELECT c.column_name AS "Column",
.           cc.compression_type AS "Actual compression",
.           AVG(cc.compressed_size) "Compressed",
.           AVG(cc.uncompressed_size) "Uncompressed",
.           AVG(cc.uncompressed_size::FLOAT/ cc.compressed_size) -1 AS "Compression␣
↪effectiveness",
.           MIN(c.compression_strategy) AS "Compression strategy"
.    FROM sqream_catalog.chunk_columns cc
.      INNER JOIN sqream_catalog.columns c
.              ON cc.table_id = c.table_id
.             AND cc.database_name = c.database_name
.             AND cc.column_id = c.column_id
.
.    WHERE c.table_name = 'ontime'
.
.    GROUP BY 1,
.             2;

Column                     | Actual compression | Compressed | Uncompressed |␣
↪Compression effectiveness | Compression strategy
--------------------------+--------------------+------------+--------------+----------
↪------------------+---------------------
actualelapsedtime@null     | dict               |     129177 |      1032957 |        ␣
↪                 7 | default
actualelapsedtime@val      | dict               |    1379797 |      4131831 |        ␣
↪                 2 | default
airlineid                  | dict               |     578150 |      2065915 |        ␣
↪               2.7 | default
airtime@null               | dict               |     130011 |      1039625 |        ␣
↪                 7 | default
```

(continues on next page)

(continued from previous page)

```
airtime@null               | rle                  |     93404 |     1019833 |         ␣
↪        116575.61 | default
airtime@val                | dict                 |   1142045 |     4131831 |         ␣
↪             7.57 | default
arrdel15@null              | dict                 |    129177 |     1032957 |         ␣
↪                7 | default
arrdel15@val               | dict                 |    129183 |     4131831 |         ␣
↪            30.98 | default
arrdelay@null              | dict                 |    129177 |     1032957 |         ␣
↪                7 | default
arrdelay@val               | dict                 |   1389660 |     4131831 |         ␣
↪                2 | default
arrdelayminutes@null       | dict                 |    129177 |     1032957 |         ␣
↪                7 | default
arrdelayminutes@val        | dict                 |   1356034 |     4131831 |         ␣
↪             2.08 | default
arrivaldelaygroups@null    | dict                 |    129177 |     1032957 |         ␣
↪                7 | default
arrivaldelaygroups@val     | p4d                  |    516539 |     2065915 |         ␣
↪                3 | default
arrtime@null               | dict                 |    129177 |     1032957 |         ␣
↪                7 | default
arrtime@val                | p4d                  |   1652799 |     2065915 |         ␣
↪             0.25 | default
arrtimeblk                 | dict                 |    688870 |     9296621 |         ␣
↪            12.49 | default
cancellationcode@null      | dict                 |    129516 |     1035666 |         ␣
↪                7 | default
cancellationcode@null      | rle                  |     54392 |     1031646 |         ␣
↪        131944.62 | default
cancellationcode@val       | dict                 |    263149 |     1032957 |         ␣
↪             4.12 | default
cancelled                  | dict                 |    129183 |     4131831 |         ␣
↪            30.98 | default
carrier                    | dict                 |    578150 |     2065915 |         ␣
↪              2.7 | default
carrierdelay@null          | dict                 |    129516 |     1035666 |         ␣
↪                7 | default
carrierdelay@null          | flat                 |   1041250 |     1041250 |         ␣
↪                0 | default
carrierdelay@null          | rle                  |      4869 |     1026493 |         ␣
↪         202740.2 | default
carrierdelay@val           | dict                 |    834559 |     4131831 |         ␣
↪            14.57 | default
crsarrtime                 | p4d                  |   1652799 |     2065915 |         ␣
↪             0.25 | default
crsdeptime                 | p4d                  |   1652799 |     2065915 |         ␣
↪             0.25 | default
crselapsedtime@null        | dict                 |    130449 |     1043140 |         ␣
↪                7 | default
crselapsedtime@null        | rle                  |      3200 |     1013388 |         ␣
↪        118975.75 | default
crselapsedtime@val         | dict                 |   1182286 |     4131831 |         ␣
↪              2.5 | default
dayofmonth                 | dict                 |    688730 |     1032957 |         ␣
↪              0.5 | default
dayofweek                  | dict                 |    393577 |     1032957 |         ␣
↪             1.62 | default
```

(continues on next page)

(continued from previous page)

```
departuredelaygroups@null | dict          |    129177 |    1032957 |  ␣
↪               7 | default
departuredelaygroups@val  | p4d           |    516539 |    2065915 |  ␣
↪               3 | default
depdel15@null             | dict          |    129177 |    1032957 |  ␣
↪               7 | default
depdel15@val              | dict          |    129183 |    4131831 |  ␣
↪           30.98 | default
depdelay@null             | dict          |    129177 |    1032957 |  ␣
↪               7 | default
depdelay@val              | dict          |   1384453 |    4131831 |  ␣
↪            2.01 | default
depdelayminutes@null      | dict          |    129177 |    1032957 |  ␣
↪               7 | default
depdelayminutes@val       | dict          |   1362893 |    4131831 |  ␣
↪            2.06 | default
deptime@null              | dict          |    129177 |    1032957 |  ␣
↪               7 | default
deptime@val               | p4d           |   1652799 |    2065915 |  ␣
↪            0.25 | default
deptimeblk                | dict          |    688870 |    9296621 |  ␣
↪           12.49 | default
month                     | dict          |    247852 |    1035246 |  ␣
↪            3.38 | default
month                     | rle           |         5 |     607346 |  ␣
↪        121468.2 | default
origin                    | dict          |   1119457 |    3098873 |  ␣
↪            1.78 | default
quarter                   | rle           |         8 |    1032957 |  ␣
↪       136498.61 | default
securitydelay@null        | dict          |    129516 |    1035666 |  ␣
↪               7 | default
securitydelay@null        | flat          |   1041250 |    1041250 |  ␣
↪               0 | default
securitydelay@null        | rle           |      4869 |    1026493 |  ␣
↪        202740.2 | default
securitydelay@val         | dict          |    581893 |    4131831 |  ␣
↪           15.39 | default
tailnum@null              | dict          |    129516 |    1035666 |  ␣
↪               7 | default
tailnum@null              | rle           |     38643 |    1031646 |  ␣
↪       121128.68 | default
tailnum@val               | dict          |   1659918 |   12395495 |  ␣
↪           22.46 | default
taxiin@null               | dict          |    130011 |    1039625 |  ␣
↪               7 | default
taxiin@null               | rle           |     93404 |    1019833 |  ␣
↪       116575.61 | default
taxiin@val                | dict          |    839917 |    4131831 |  ␣
↪            8.49 | default
taxiout@null              | dict          |    130011 |    1039625 |  ␣
↪               7 | default
taxiout@null              | rle           |     84327 |    1019833 |  ␣
↪       116575.86 | default
taxiout@val               | dict          |    891539 |    4131831 |  ␣
↪            8.28 | default
totaladdgtime@null        | dict          |    129516 |    1035666 |  ␣
↪               7 | default
```

(continues on next page)

(continued from previous page)

```
totaladdgtime@null        | rle                |       3308 |      1031646 |         ␣
↪       191894.18 | default
totaladdgtime@val         | dict               |     465839 |      4131831 |         ␣
↪           20.51 | default
uniquecarrier             | dict               |     578221 |      7230705 |         ␣
↪           11.96 | default
year                      | rle                |          6 |      2065915 |         ␣
↪       317216.08 | default
```

##### 7.3.2.4.1 Notes on reading this table:

1. Higher numbers in the *effectiveness* column represent better compressions. 0 represents a column that wasn't compressed at all.
2. Column names are the internal representation. Names with `@null` and `@val` suffixes represent a nullable column's null (boolean) and values respectively, but are treated as one logical column.
3. The query lists all actual compressions for a column, so it may appear several times if the compression has changed mid-way through the loading (as with the `carrierdelay` column).
4. When `default` is the compression strategy, the system automatically selects the best compression. This can also mean no compression at all (`flat`).

## 7.3.3 Compression best practices

### 7.3.3.1 Let SQream DB decide on the compression strategy

In general, SQream DB will decide on the best compression strategy in most cases.

When overriding compression strategies, we recommend benchmarking not just storage size but also query and load performance.

### 7.3.3.2 Maximize the advantage of each compression schemes

Some compression schemes perform better when data is organized in a specific way.

For example, to take advantage of RLE, sorting a column may result in better performance and reduced disk-space and I/O usage. Sorting a column partially may also be beneficial. As a rule of thumb, aim for run-lengths of more than 10 consecutive values.

### 7.3.3.3 Choose data types that fit the data

Adapting to the narrowest data type will improve query performance and also reduce disk space usage. However, smaller data types may compress better than larger types.

For example, use the smallest numeric data type that will accommodate your data. Using `BIGINT` for data that fits in `INT` or `SMALLINT` can use more disk space and memory for query execution.

Using `FLOAT` to store integers will reduce compression's effectiveness significantly.

## 7.4 Python UDF (User-Defined Functions)

User-defined functions (UDFs) are a feature that extends SQream DB's built in SQL functionality. SQream DB's Python UDFs allow developers to create new functionality in SQL by writing the lower-level language implementation in Python.

**In this topic:**



### 7.4.1 A simple example

Most databases have an UPPER function, including SQream DB. However, assume that this function is missing for the sake of this example.

You can write a function in Python to uppercase a text value using the create_function syntax.

```
CREATE FUNCTION my_upper (x1 text)
  RETURNS text
  AS $$
return x1.upper()
$$ LANGUAGE PYTHON;
```

Let's break down this example:

- `CREATE FUNCTION my_upper` - Create a function called `my_upper`. This name must be unique in the current database
- `(x1 text)` - the function accepts one argument named `x1` which is of the SQL type `TEXT`. All data types are supported.
- `RETURNS text` - the function returns the same type - `TEXT`. All data types are supported.
- `AS $$` - what follows is some code that we don't want to quote, so we use dollar-quoting (`$$`) instead of single quotes (`'`).
- `return x1.upper()` - the Python function's body is the argument named `x1`, uppercased.
- `$$ LANGUAGE PYTHON` - this is the end of the function, and it's in the Python language.

#### Running this example

After creating the function, you can use it in any SQL query.

For example:

```
master=>CREATE TABLE jabberwocky(line text);
executed
master=> INSERT INTO jabberwocky VALUES
.   ('''Twas brillig, and the slithy toves '), ('      Did gyre and gimble in the␣
↪wabe: ')
.   ,('All mimsy were the borogoves, '), ('      And the mome raths outgrabe. ')
.   ,('"Beware the Jabberwock, my son! '), ('      The jaws that bite, the claws that␣
↪catch! ')
.   ,('Beware the Jubjub bird, and shun '), ('      The frumious Bandersnatch!" ');
executed
master=> SELECT line, my_upper(line) FROM jabberwocky;
line                                              | my_upper
--------------------------------------------------+-----------------------------------
↪-------------
'Twas brillig, and the slithy toves               | 'TWAS BRILLIG, AND THE SLITHY TOVES
      Did gyre and gimble in the wabe:            |       DID GYRE AND GIMBLE IN THE␣
↪WABE:
All mimsy were the borogoves,                     | ALL MIMSY WERE THE BOROGOVES,
      And the mome raths outgrabe.                |       AND THE MOME RATHS OUTGRABE.
"Beware the Jabberwock, my son!                   | "BEWARE THE JABBERWOCK, MY SON!
      The jaws that bite, the claws that catch!   |       THE JAWS THAT BITE, THE␣
↪CLAWS THAT CATCH!
Beware the Jubjub bird, and shun                  | BEWARE THE JUBJUB BIRD, AND SHUN
      The frumious Bandersnatch!"                 |       THE FRUMIOUS BANDERSNATCH!"
```

## 7.4.2 Why use UDFs?

- They allow simpler statements - You can create the function once, store it in the database, and call it any number of times in a statement.
- They can be shared - UDFs can be created by a database administrator, and then used by other roles.
- They can simplify downstream code - UDFs can be modified in SQream DB independently of program source code.

## 7.4.3 SQream DB's UDF support

### 7.4.3.1 Scalar functions

SQream DB's UDFs are scalar functions. This means that the UDF returns a single data value of the type defined in the `RETURNS` clause. For an inline scalar function, the returned scalar value is the result of a single statement.

#### 7.4.3.2 Python

At this time, SQream DB's UDFs are supported for Python.

Python 3.6.7 is installed alongside SQream DB, for use exclusively by SQream DB. You may have a different version of Python installed on your server.

To find which version of Python is installed for use by SQream DB, create and run this UDF:

```
master=> CREATE OR REPLACE FUNCTION py_version()
.   RETURNS text
.   AS $$
. import sys
. return ("Python version: " + sys.version + ". Path: " + sys.base_exec_prefix)
.   $$ LANGUAGE PYTHON;
executed
master=> SELECT py_version();
py_version
-------------------------------------------------------------------------------------
Python version: 3.6.7 (default, Jul 22 2019, 11:03:54) [GCC 5.4.0].
Path: /opt/sqream/python-3.6.7-5.4.0
```

#### 7.4.3.3 Using modules

To import a Python module, use the standard `import` syntax in the first lines of the user-defined function.

### 7.4.4 Finding existing UDFs in the catalog

The `user_defined_functions` catalog view contains function information.

Here's how you'd list all UDFs in the system:

```
master=> SELECT * FROM sqream_catalog.user_defined_functions;
database_name | function_id | function_name
--------------+-------------+--------------
master        |           1 | my_upper
```

### 7.4.5 Getting the DDL for a function

```
master=> SELECT GET_FUNCTION_DDL('my_upper');
ddl
----------------------------------------------------
create function "my_upper" (x1 text) returns text as
$$
   return x1.upper()
$$
language python volatile;
```

See get_function_ddl for more information.

### 7.4.6 Error handling

In UDFs, any error that occurs causes the execution of the function to stop. This in turn causes the statement that invoked the function to be canceled.

### 7.4.7 Permissions and sharing

To create a UDF, the creator needs the `CREATE FUNCTION` permission at the database level.

For example, to grant `CREATE FUNCTION` to a non-superuser role:

```
GRANT CREATE FUNCTION ON DATABASE master TO mjordan;
```

To execute a UDF, the role needs the `EXECUTE FUNCTION` permission for every function.

For example, to grant the permission to the `r_bi_users` role group, run:

```
GRANT EXECUTE ON FUNCTION my_upper TO r_bi_users;
```

**Note:** Functions are stored for each database, outside of any schema.

See more information about permissions in the *Access control guide*.

### 7.4.8 Best practices

Although user-defined functions add flexibility, they may have some performance drawbacks. They are not usually a replacement for subqueries or views.

In some cases, the user-defined function provides benefits like sharing extended functionality which makes it very appealing.

Use user-defined functions sparingly in the `WHERE` clause. SQream DB can't optimize the function's usage, and it will be called once for every value. If possible, you should narrow down the number of results before the UDF is called by using a subquery.

## 7.5 Workload Manager

The **Workload Manager** allows SQream workers to identify their availability to clients with specific service names. The load balancer uses that information to route statements to specific workers.

### 7.5.1 Overview

The Workload Manager allows a system engineer or database administrator to allocate specific workers and compute resources for various tasks.

For example:

1. Creating a service queue named `ETL` and allocating two workers exclusively to this service prevents non-`ETL` statements from utilizing these compute resources.
2. Creating a service for the company's leadership during working hours for dedicated access, and disabling this service at night to allow maintenance operations to use the available compute.

### 7.5.2 Setting Up Service Queues

By default, every worker subscribes to the `sqream` service queue.

Additional service names are configured in the configuration file for every worker, but can also be set on a per-session basis.

### 7.5.3 Example - Allocating ETL Resources

Allocating ETL resources ensures high quality service without requiring management users to wait.

The configuration in this example allocates resources as shown below:

- 1 worker for ETL work
- 3 workers for general queries
- All workers assigned to queries from management

| Service / Worker | Worker #1 | Worker #2 | Worker #3 | Worker #4 |
|---|---|---|---|---|
| ETL | ✓ | ✗ | ✗ | ✗ |
| Query service | ✗ | ✓ | ✓ | ✓ |
| Management | ✓ | ✓ | ✓ | ✓ |

This configuration gives the ETL queue dedicated access to one worker, which cannot be used..

Queries from management uses any available worker.

#### 7.5.3.1 Creating the Configuration

The persistent configuration for this set-up is listed in the four configuration files shown below.

Each worker gets a comma-separated list of service queues that it subscribes to. These services are specified in the `initialSubscribedServices` attribute.

Listing 1: Worker #1

```
{
    "compileFlags": {
    },
    "runtimeFlags": {
    },
    "runtimeGlobalFlags": {
       "initialSubscribedServices" : "etl,management"
    },
    "server": {
        "gpu": 0,
        "port": 5000,
        "cluster": "/home/rhendricks/raviga_database",
        "licensePath": "/home/sqream/.sqream/license.enc"
    }
}
```

Listing 2: Workers #2, #3, #4

```
{
    "compileFlags": {
    },
    "runtimeFlags": {
    },
    "runtimeGlobalFlags": {
       "initialSubscribedServices" : "query,management"
    },
    "server": {
        "gpu": 1,
        "port": 5001,
        "cluster": "/home/rhendricks/raviga_database",
        "licensePath": "/home/sqream/.sqream/license.enc"
    }
}
```

**Tip:** You can create this configuration temporarily (for the current session only) by using the subscribe_service and unsubscribe_service statements.

#### 7.5.3.2 Verifying the Configuration

Use show_subscribed_instances to view service subscriptions for each worker. Use SHOW_SERVER_STATUS to see the statement queues.

```
t=> SELECT SHOW_SUBSCRIBED_INSTANCES();
service    | servernode | serverip      | serverport
-----------+------------+---------------+-----------
management | node_9383  | 192.168.0.111 |       5000
etl        | node_9383  | 192.168.0.111 |       5000
query      | node_9384  | 192.168.0.111 |       5001
management | node_9384  | 192.168.0.111 |       5001
query      | node_9385  | 192.168.0.111 |       5002
management | node_9385  | 192.168.0.111 |       5002
query      | node_9551  | 192.168.1.91  |       5000
management | node_9551  | 192.168.1.91  |       5000
```

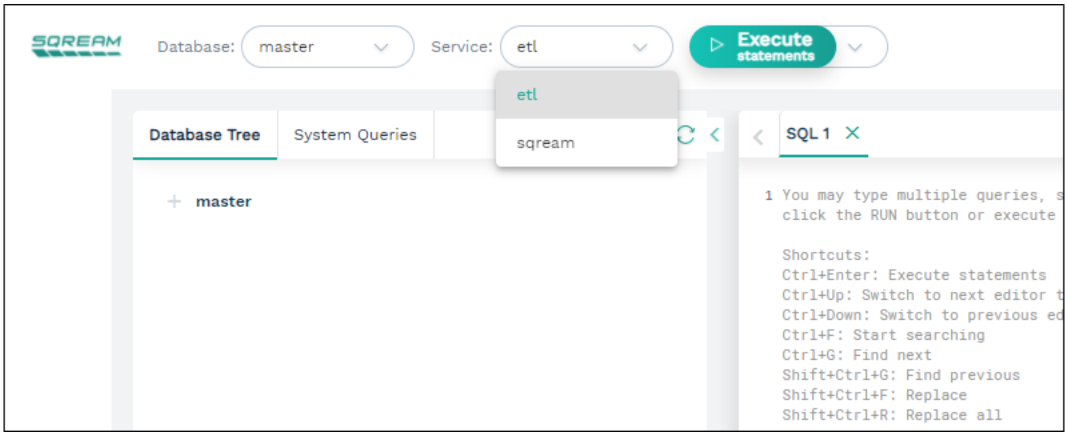
### 7.5.4 Configuring a Client Connection to a Specific Service

You can configure a client connection to a specific service in one of the following ways:

- *Using SQream Studio*
- *Using the SQream SQL CLI Reference*
- *Using a JDBC Client Driver*
- *Using an ODBC Client Driver*
- *Using a Python Client Driver*
- *Using a Node.js Client Driver*

#### 7.5.4.1 Using SQream Studio

When using **SQream Studio**, you can configure a client connection to a specific service from the SQream Studio, as shown below:




For more information, in Studio, see Executing Statements from the Toolbar.

#### 7.5.4.2 Using the SQream SQL CLI Reference

When using the **SQream SQL CLI Reference**, you can configure a client connection to a specific service by adding `--service=<service name>` to the command line, as shown below:

```
$ sqream sql --port=3108 --clustered --username=mjordan --databasename=master --
↪service=etl
Password:

Interactive client mode
To quit, use ^D or \q.

master=>_
```

For more information, see the *Sqream SQL CLI Reference*.

#### 7.5.4.3 Using a JDBC Client Driver

When using a **JDBC client driver**, you can configure a client connection to a specific service by adding `--service=<service name>` to the command line, as shown below:

Listing 3: JDBC Connection String

```
jdbc:Sqream://127.0.0.1:3108/raviga;user=rhendricks;password=Tr0ub4dor&3;service=etl;
↪cluster=true;ssl=false;
```

For more information, see the JDBC Client Driver.

#### 7.5.4.4 Using an ODBC Client Driver

When using an **ODBC client driver**, you can configure a client connection to a specific service on Linux by modifying the *DSN parameters* in `odbc.ini`.

For example, `Service="etl"`:

Listing 4: odbc.ini

```
[sqreamdb]
Description=64-bit Sqream ODBC
Driver=/home/rhendricks/sqream_odbc64/sqream_odbc64.so
Server="127.0.0.1"
Port="3108"
Database="raviga"
Service="etl"
User="rhendricks"
Password="Tr0ub4dor&3"
Cluster=true
Ssl=false
```

On Windows, change the parameter in the *DSN editing window*.

For more information, see the ODBC Client Driver.

#### 7.5.4.5 Using a Python Client Driver

When using a **Python client driver**, you can configure a client connection to a specific service by setting the `service` parameter in the connection command, as shown below:

Listing 5: Python

```
con = pysqream.connect(host='127.0.0.1', port=3108, database='raviga'
                       , username='rhendricks', password='Tr0ub4dor&3'
                       , clustered=True, use_ssl = False, service='etl')
```

For more information, see the Python (pysqream) connector.

#### 7.5.4.6 Using a Node.js Client Driver

When using a **Node.js client driver**, you can configure a client connection to a specific service by adding the service to the connection settings, as shown below:

Listing 6: Node.js

```
const Connection = require('sqreamdb');
const config = {
   host: '127.0.0.1',
   port: 3108,
   username: 'rhendricks',
   password: 'Tr0ub4dor&3',
   connectDatabase: 'raviga',
   cluster: 'true',
   service: 'etl'
};
```

For more information, see the Node.js Client Driver.

## 7.6 Transactions

SQream DB supports serializable transactions. This is also called 'ACID compliance'.

The implementation of transactions means that commit, rollback and recovery are all extremely fast.

SQream DB has extremely fast bulk insert speed, with minimal slowdown when running concurrent inserts. There is no performance reason to break large inserts up into multiple transactions.

The phrase "supporting transactions" for a database system sometimes means having good performance for OLTP workloads, SQream DB's transaction system does not have high performance for high concurrency OLTP workloads.

SQream DB also supports *transactional DDL*.

## 7.7 Concurrency and Locks

Locks are used in SQream DB to provide consistency when there are multiple concurrent transactions updating the database.

Read only transactions are never blocked, and never block anything. Even if you drop a database while concurrently running a query on it, both will succeed correctly (as long as the query starts running before the drop database commits).

### 7.7.1 Locking Modes

SQream DB has two kinds of locks:

- `exclusive` - this lock mode prevents the resource from being modified by other statements

  This lock tells other statements that they'll have to wait in order to change an object.

  DDL operations are always exclusive. They block other DDL operations, and update DML operations (insert and delete).

- `inclusive` - For insert operations, an inclusive lock is obtained on a specific object. This prevents other statements from obtaining an exclusive lock on the object.

  This lock allows other statements to insert or delete data from a table, but they'll have to wait in order to run DDL.

### 7.7.2 When are Locks Obtained?

| Operation | select | insert | delete, truncate | DDL |
|---|---|---|---|---|
| select | Concurrent | Concurrent | Concurrent | Concurrent |
| insert | Concurrent | Concurrent | Concurrent | Wait |
| delete, truncate | Concurrent | Concurrent | Wait | Wait |
| DDL | Concurrent | Wait | Wait | Wait |

Statements that wait will exit with an error if they hit the lock timeout. The default timeout is 3 seconds, see `statementLockTimeout`.

### 7.7.3 Monitoring Locks

Monitoring locks across the cluster can be useful when transaction contention takes place, and statements appear "stuck" while waiting for a previous statement to release locks.

The utility show_locks can be used to see the active locks.

In this example, we create a table based on results (create_table_as), but we are also effectively dropping the previous table (by using `OR REPLACE` which also drops the table). Thus, SQream DB applies locks during the table creation process to prevent the table from being altered during it's creation.

```
t=> SELECT SHOW_LOCKS();
statement_id | statement_string                                                      ␣
↪                        | username | server       | port | locked_object           ␣
↪        | lockmode  | statement_start_time | lock_start_time
-------------+-----------------------------------------------------------------------
↪------------------------+----------+--------------+------+-----------------------
↪--------+-----------+----------------------+--------------------
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | database$t            ␣
↪            | Inclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | globalpermission$     ␣
↪            | Exclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | schema$t$public       ␣
↪            | Inclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | table$t$public$nba2
↪$Insert      | Exclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
287          | CREATE OR REPLACE TABLE nba2 AS SELECT "Name" FROM nba WHERE REGEXP_
↪COUNT("Name", '( )+', 8)>1; | sqream   | 192.168.1.91 | 5000 | table$t$public$nba2
↪$Update      | Exclusive | 2019-12-26 00:03:30  | 2019-12-26 00:03:30
```

For more information on troubleshooting lock related issues, see *Lock Related Issues*.

## 7.8 Concurrency and Scaling in SQream DB

A SQream DB cluster can concurrently run one regular statement per worker process. A number of small statements will execute alongside these statements without waiting or blocking anything.

SQream DB supports `n` concurrent statements by having `n` workers in a cluster. Each worker uses a fixed slice of a GPU's memory, with usual values are around 8-16GB of GPU memory per worker. This size is ideal for queries running on large data with potentially large row sizes.

### 7.8.1 Scaling when data sizes grow

For many statements, SQream DB scales linearly when adding more storage and querying on large data sets. It uses very optimised 'brute force' algorithms and implementations, which don't suffer from sudden performance cliffs at larger data sizes.

### 7.8.2 Scaling when queries are queueing

SQream DB scales well by adding more workers, GPUs, and nodes to support more concurrent statements.

### 7.8.3 What to do when queries are slow

Adding more workers or GPUs does not boost the performance of a single statement or query.

To boost the performance of a single statement, start by examining the *best practices* and ensure the guidelines are followed.

Adding additional RAM to nodes, using more GPU memory, and faster CPUs or storage can also sometimes help.

#### Need help?

Analyzing complex workloads can be challenging. SQream's experienced customer support has the experience to advise on these matters to ensure the best experience.

Visit SQream's support portal for additional support.

CHAPTER

EIGHT

# OPERATIONAL GUIDES

The **Operational Guides** section describes processes that SQream users can manage to affect the way their system operates, such as creating storage clusters and monitoring query performance.

This section summarizes the following operational guides:

## 8.1 Access Control

### 8.1.1 Password Policy

The **Password Policy** describes the following:

- *Password Strength Requirements*
- *Brute Force Prevention*

#### 8.1.1.1 Password Strength Requirements

As part of our compliance with GDPR standards SQream relies on a strong password policy when accessing the CLI or Studio, with the following requirements:

- At least eight characters long.
- Mandatory upper and lowercase letters.
- At least one numeric character.
- May not include a username.
- Must include at least one special character, such as **?**, **!**, **$**, etc.

You can grant a password through the Studio graphic interface or through the CLI, as in the following example command:

```
CREATE ROLE user_a ;
GRANT LOGIN to user_a ;
GRANT PASSWORD 'BBAu47?fqPL' to user_a ;
```

Granting a password that does not comply with the above requirements generates an error message with a request to modify it;

```
The password you attempted to create does not comply with SQream's security
↪requirements.

Your password must:

* Be at least eight characters long.

* Contain upper and lowercase letters.

* Contain at least one numeric character.

* Not include a username.

* Include at least one special character, such as **?**, **!**, **$**, etc.
```

#### 8.1.1.2 Brute Force Prevention

Unsuccessfully attempting to log in three times displays the following message:

```
The user is locked. Please contact your system administrator to reset the password
↪and regain access functionality.
```

You must have superuser permissions to release a locked user to grant a new password:

```
GRANT PASSWORD '<password>' to <blocked_user>;
```

For more information, see login_max_retries.

> **Warning:** Because superusers can also be blocked, **you must have** at least two superusers per cluster.

### 8.1.2 Overview

Access control refers to SQream's authentication and authorization operations, managed using a **Role-Based Access Control (RBAC)** system, such as ANSI SQL or other SQL products. SQream's default permissions system is similar to Postgres, but is more powerful. SQream's method lets administrators prepare the system to automatically provide objects with their required permissions.

SQream users can log in from any worker, which verify their roles and permissions from the metadata server. Each statement issues commands as the role that you're currently logged into. Roles are defined at the cluster level, and are valid for all databases in the cluster. To bootstrap SQream, new installations require one `SUPERUSER` role, typically named `sqream`. You can only create new roles by connecting as this role.

Access control refers to the following basic concepts:

- **Role** - A role can be a user, a group, or both. Roles can own database objects (such as tables) and can assign permissions on those objects to other roles. Roles can be members of other roles, meaning a user role can inherit permissions from its parent role.

- **Authentication** - Verifies the identity of the role. User roles have usernames (or **role names**) and passwords.

- **Authorization** - Checks that a role has permissions to perform a particular operation, such as the grant command.

### 8.1.3 Managing Roles

Roles are used for both users and groups, and are global across all databases in the SQream cluster. For a ROLE to be used as a user, it requires a password and log-in and connect permissionss to the relevant databases.

The Managing Roles section describes the following role-related operations:



#### 8.1.3.1 Creating New Roles (Users)

A user role logging in to the database requires LOGIN permissions and as a password.

The following is the syntax for creating a new role:

```
CREATE ROLE <role_name> ;
GRANT LOGIN to <role_name> ;
GRANT PASSWORD <'new_password'> to <role_name> ;
GRANT CONNECT ON DATABASE <database_name> to <role_name> ;
```

The following is an example of creating a new role:

```
CREATE  ROLE  new_role_name  ;
GRANT  LOGIN  TO  new_role_name;
GRANT  PASSWORD  'my_password' to new_role_name;
GRANT  CONNECT  ON  DATABASE  master to new_role_name;
```

A database role may have a number of permissions that define what tasks it can perform, which are assigned using the grant command.

#### 8.1.3.2 Dropping a User

The following is the syntax for dropping a user:

```
DROP ROLE <role_name> ;
```

The following is an example of dropping a user:

```
DROP ROLE  admin_role ;
```

#### 8.1.3.3 Altering a User Name

The following is the syntax for altering a user name:

```
ALTER ROLE <role_name> RENAME TO <new_role_name> ;
```

The following is an example of altering a user name:

```
ALTER ROLE admin_role RENAME TO copy_role ;
```

#### 8.1.3.4 Changing a User Password

You can change a user role's password by granting the user a new password.

The following is an example of changing a user password:

```
GRANT  PASSWORD  <'new_password'>  TO  rhendricks;
```

**Note:** Granting a new password overrides any previous password. Changing the password while the role has an active running statement does not affect that statement, but will affect subsequent statements.

#### 8.1.3.5 Altering Public Role Permissions

There is a public role which always exists. Each role is granted to the `PUBLIC` role (i.e. is a member of the public group), and this cannot be revoked. You can alter the permissions granted to the public role.

The `PUBLIC` role has `USAGE` and `CREATE` permissions on `PUBLIC` schema by default, therefore, new users can create, insert, delete, and select from objects in the `PUBLIC` schema.

#### 8.1.3.6 Altering Role Membership (Groups)

Many database administrators find it useful to group user roles together. By grouping users, permissions can be granted to, or revoked from a group with one command. In SQream DB, this is done by creating a group role, granting permissions to it, and then assigning users to that group role.

To use a role purely as a group, omit granting it `LOGIN` and `PASSWORD` permissions.

The `CONNECT` permission can be given directly to user roles, and/or to the groups they are part of.

```
CREATE ROLE my_group;
```

Once the group role exists, you can add user roles (members) using the `GRANT` command. For example:

```
-- Add my_user to this group
GRANT my_group TO my_user;
```

To manage object permissions like databases and tables, you would then grant permissions to the group-level role (see the permissions table below.

All member roles then inherit the permissions from the group. For example:

```
-- Grant all group users connect permissions
GRANT  CONNECT  ON  DATABASE  a_database  TO  my_group;

-- Grant all permissions on tables in public schema
GRANT  ALL  ON  all  tables  IN  schema  public  TO  my_group;
```

Removing users and permissions can be done with the `REVOKE` command:

```
-- remove my_other_user from this group
REVOKE my_group FROM my_other_user;
```

### 8.1.4 Permissions

The following table displays the access control permissions:

| Permission | Description |
|---|---|
| **Object/Layer: All Databases** | |
| `LOGIN` | use role to log into the system (the role also needs connect permission on the database it is connecting to) |
| `PASSWORD` | the password used for logging into the system |
| `SUPERUSER` | no permission restrictions on any activity |
| **Object/Layer: Database** | |
| `SUPERUSER` | no permission restrictions on any activity within that database (this does not include modifying roles or permissions) |
| `CONNECT` | connect to the database |
| `CREATE` | create schemas in the database |
| `CREATE FUNCTION` | create and drop functions |
| **Object/Layer: Schema** | |
| `USAGE` | allows additional permissions within the schema |
| `CREATE` | create tables in the schema |
| **Object/Layer: Table** | |
| `SELECT` | select from the table |
| `INSERT` | insert into the table |
| `UPDATE` | UPDATE the value of certain columns in existing rows without creating a table |
| `DELETE` | delete and truncate on the table |
| `DDL` | drop and alter on the table |
| `ALL` | all the table permissions |
| **Object/Layer: Function** | |
| `EXECUTE` | use the function |
| `DDL` | drop and alter on the function |
| `ALL` | all function permissions |

#### 8.1.4.1 GRANT

grant gives permissions to a role.

```
-- Grant permissions at the instance/ storage cluster level:
GRANT

{ SUPERUSER
| LOGIN
| PASSWORD '<password>'
}
TO <role> [, ...]

-- Grant permissions at the database level:
     GRANT {{CREATE | CONNECT| DDL | SUPERUSER | CREATE FUNCTION} [, ...] | ALL
↪[PERMISSIONS]}

ON DATABASE <database> [, ...]
TO <role> [, ...]

-- Grant permissions at the schema level:
GRANT {{ CREATE | DDL | USAGE | SUPERUSER } [, ...] | ALL [
PERMISSIONS ]}
ON SCHEMA <schema> [, ...]
TO <role> [, ...]

-- Grant permissions at the object level:
GRANT {{SELECT | INSERT | DELETE | DDL } [, ...] | ALL [PERMISSIONS]}
ON { TABLE <table_name> [, ...] | ALL TABLES IN SCHEMA <schema_name> [, ...]}
TO <role> [, ...]

-- Grant execute function permission:
GRANT {ALL | EXECUTE | DDL} ON FUNCTION function_name
TO role;

-- Allows role2 to use permissions granted to role1
GRANT <role1> [, ...]
TO <role2>

 -- Also allows the role2 to grant role1 to other roles:
GRANT <role1> [, ...]
TO <role2>
WITH ADMIN OPTION
```

GRANT examples:

```
GRANT  LOGIN,superuser  TO  admin;

GRANT  CREATE  FUNCTION  ON  database  master  TO  admin;

GRANT  SELECT  ON  TABLE  admin.table1  TO  userA;

GRANT  EXECUTE  ON  FUNCTION  my_function  TO  userA;

GRANT  ALL  ON  FUNCTION  my_function  TO  userA;

GRANT  DDL  ON  admin.main_table  TO  userB;
```

(continues on next page)

(continued from previous page)

```
GRANT ALL ON all tables IN schema public TO userB;

GRANT admin TO userC;

GRANT superuser ON schema demo TO userA

GRANT admin_role TO userB;
```

#### 8.1.4.2 REVOKE

revoke removes permissions from a role.

```
-- Revoke permissions at the instance/ storage cluster level:
REVOKE
{ SUPERUSER
| LOGIN
| PASSWORD
}
FROM <role> [, ...]

-- Revoke permissions at the database level:
REVOKE {{CREATE | CONNECT | DDL | SUPERUSER | CREATE FUNCTION}[, ...] |ALL
↪[PERMISSIONS]}
ON DATABASE <database> [, ...]
FROM <role> [, ...]

-- Revoke permissions at the schema level:
REVOKE { { CREATE | DDL | USAGE | SUPERUSER } [, ...] | ALL [PERMISSIONS]}
ON SCHEMA <schema> [, ...]
FROM <role> [, ...]

-- Revoke permissions at the object level:
REVOKE { { SELECT | INSERT | DELETE | DDL } [, ...] | ALL }
ON { [ TABLE ] <table_name> [, ...] | ALL TABLES IN SCHEMA

      <schema_name> [, ...] }
FROM <role> [, ...]

-- Removes access to permissions in role1 by role 2
REVOKE <role1> [, ...] FROM <role2> [, ...] WITH ADMIN OPTION

-- Removes permissions to grant role1 to additional roles from role2
REVOKE <role1> [, ...] FROM <role2> [, ...] WITH ADMIN OPTION
```

Examples:

```
REVOKE superuser on schema demo from userA;

REVOKE delete on admin.table1 from userB;

REVOKE login from role_test;

REVOKE CREATE FUNCTION FROM admin;
```

#### 8.1.4.3 Default permissions

The default permissions system (See alter_default_permissions) can be used to automatically grant permissions to newly created objects (See the departmental example below for one way it can be used).

A default permissions rule looks for a schema being created, or a table (possibly by schema), and is table to grant any permission to that object to any role. This happens when the create table or create schema statement is run.

```
ALTER DEFAULT PERMISSIONS FOR target_role_name
     [IN schema_name, ...]
     FOR { TABLES | SCHEMAS }
     { grant_clause | DROP grant_clause}
     TO ROLE { role_name | public };

grant_clause ::=
  GRANT
     { CREATE FUNCTION
     | SUPERUSER
     | CONNECT
     | CREATE
     | USAGE
     | SELECT
     | INSERT
     | DELETE
     | DDL
     | EXECUTE
     | ALL
     }
```

### 8.1.5 Departmental Example

You work in a company with several departments.

The example below shows you how to manage permissions in a database shared by multiple departments, where each department has different roles for the tables by schema. It walks you through how to set the permissions up for existing objects and how to set up default permissions rules to cover newly created objects.

The concept is that you set up roles for each new schema with the correct permissions, then the existing users can use these roles.

A superuser must do new setup for each new schema which is a limitation, but superuser permissions are not needed at any other time, and neither are explicit grant statements or object ownership changes.

In the example, the database is called `my_database`, and the new or existing schema being set up to be managed in this way is called `my_schema`.

Our departmental example has four user group roles and seven users roles

There will be a group for this schema for each of the following:

| Group | Activities |
|---|---|
| database designers | create, alter and drop tables |
| updaters | insert and delete data |
| readers | read data |
| security officers | add and remove users from these groups |

#### 8.1.5.1 Setting up the department permissions

As a superuser, you connect to the system and run the following:

```
-- create the groups

CREATE ROLE my_schema_security_officers;
CREATE ROLE my_schema_database_designers;
CREATE ROLE my_schema_updaters;
CREATE ROLE my_schema_readers;

-- grant permissions for each role
-- we grant permissions for existing objects here too,
-- so you don't have to start with an empty schema

-- security officers

GRANT connect ON DATABASE my_database TO my_schema_security_officers;
GRANT usage ON SCHEMA my_schema TO my_schema_security_officers;

GRANT my_schema_database_designers TO my_schema_security_officers WITH ADMIN OPTION;
GRANT my_schema_updaters TO my_schema_security_officers WITH ADMIN OPTION;
GRANT my_schema_readers TO my_schema_security_officers WITH ADMIN OPTION;

-- database designers

GRANT connect ON DATABASE my_database TO my_schema_database_designers;
GRANT usage ON SCHEMA my_schema TO my_schema_database_designers;

GRANT create,ddl ON SCHEMA my_schema TO my_schema_database_designers;

-- updaters

GRANT connect ON DATABASE my_database TO my_schema_updaters;
GRANT usage ON SCHEMA my_schema TO my_schema_updaters;

GRANT SELECT,INSERT,DELETE ON ALL TABLES IN SCHEMA my_schema TO my_schema_updaters;

-- readers

GRANT connect ON DATABASE my_database TO my_schema_readers;
GRANT usage ON SCHEMA my_schema TO my_schema_readers;

GRANT SELECT ON ALL TABLES IN SCHEMA my_schema TO my_schema_readers;
GRANT EXECUTE ON ALL FUNCTIONS TO my_schema_readers;


-- create the default permissions for new objects

ALTER DEFAULT PERMISSIONS FOR my_schema_database_designers IN my_schema
 FOR TABLES GRANT SELECT,INSERT,DELETE TO my_schema_updaters;

-- For every table created by my_schema_database_designers, give access to my_schema_
↪readers:

ALTER DEFAULT PERMISSIONS FOR my_schema_database_designers IN my_schema
 FOR TABLES GRANT SELECT TO my_schema_readers;
```

**Note:**

- This process needs to be repeated by a user with `SUPERUSER` permissions each time a new schema is brought into this permissions management approach.
- By default, any new object created will not be accessible by our new `my_schema_readers` group. Running a `GRANT SELECT ...` only affects objects that already exist in the schema or database.

  If you're getting a `Missing the following permissions: SELECT on table 'database.public.tablename'` error, make sure that you've altered the default permissions with the `ALTER DEFAULT PERMISSIONS` statement.

#### 8.1.5.2 Creating new users in the departments

After the group roles have been created, you can now create user roles for each of your users.

```
-- create the new database designer users

CREATE  ROLE  ecodd;
GRANT  LOGIN  TO  ecodd;
GRANT  PASSWORD  'ecodds_secret_password'  TO ecodd;
GRANT  CONNECT  ON  DATABASE  my_database  TO  ecodd;
GRANT my_schema_database_designers TO ecodd;

CREATE  ROLE  ebachmann;
GRANT  LOGIN  TO  ebachmann;
GRANT  PASSWORD  'another_secret_password'  TO ebachmann;
GRANT  CONNECT  ON  DATABASE  my_database  TO  ebachmann;
GRANT my_database_designers TO ebachmann;

-- If a user already exists, we can assign that user directly to the group

GRANT my_schema_updaters TO rhendricks;

-- Create users in the readers group

CREATE  ROLE  jbarker;
GRANT  LOGIN  TO  jbarker;
GRANT  PASSWORD  'action_jack'  TO jbarker;
GRANT  CONNECT  ON  DATABASE  my_database  TO  jbarker;
GRANT my_schema_readers TO jbarker;

CREATE  ROLE  lbream;
GRANT  LOGIN  TO  lbream;
GRANT  PASSWORD  'artichoke123'  TO lbream;
GRANT  CONNECT  ON  DATABASE  my_database  TO  lbream;
GRANT my_schema_readers TO lbream;

CREATE  ROLE  pgregory;
GRANT  LOGIN  TO  pgregory;
GRANT  PASSWORD  'c1ca6a'  TO pgregory;
GRANT  CONNECT  ON  DATABASE  my_database  TO  pgregory;
GRANT my_schema_readers TO pgregory;

-- Create users in the security officers group
```

(continues on next page)

(continued from previous page)

```
CREATE  ROLE  hoover;
GRANT  LOGIN  TO  hoover;
GRANT  PASSWORD  'mintchip'  TO hoover;
GRANT  CONNECT  ON  DATABASE  my_database  TO  hoover;
GRANT my_schema_security_officers TO hoover;
```

After this setup:

- Database designers will be able to run any ddl on objects in the schema and create new objects, including ones created by other database designers
- Updaters will be able to insert and delete to existing and new tables
- Readers will be able to read from existing and new tables

All this will happen without having to run any more `GRANT` statements.

Any security officer will be able to add and remove users from these groups. Creating and dropping login users themselves must be done by a superuser.

## 8.2 Creating or Cloning Storage Clusters

When SQream DB is installed, it comes with a default storage cluster. This guide will help if you need a fresh storage cluster or a separate copy of an existing storage cluster.

### 8.2.1 Creating a new storage cluster

SQream DB comes with a CLI tool, *SqreamStorage*. This tool can be used to create a new empty storage cluster.

In this example, we will create a new cluster at `/home/rhendricks/raviga_database`:

```
$ SqreamStorage --create-cluster --cluster-root /home/rhendricks/raviga_database
Setting cluster version to: 26
```

This can also be written shorthand as `SqreamStorage -C -r /home/rhendricks/raviga_database`.

This `Setting cluster version...` message confirms the creation of the cluster successfully.

### 8.2.2 Tell SQream DB to use this storage cluster

#### 8.2.2.1 Permanently setting the storage cluster setting

To permanently set the new cluster location, change the `"cluster"` path listed in the configuration file.

For example:

```
{
    "compileFlags": {
    },
    "runtimeFlags": {
    },
    "runtimeGlobalFlags": {
    },
```

(continues on next page)

(continued from previous page)

```
    "server": {
        "gpu": 0,
        "port": 5000,
        "cluster": "/home/sqream/my_old_cluster",
        "licensePath": "/home/sqream/.sqream/license.enc"
    }
}
```

should be changed to

```
{
    "compileFlags": {
    },
    "runtimeFlags": {
    },
    "runtimeGlobalFlags": {
    },
    "server": {
        "gpu": 0,
        "port": 5000,
        "cluster": "/home/rhendricks/raviga_database",
        "licensePath": "/home/sqream/.sqream/license.enc"
    }
}
```

Now, the cluster should be restarted for the changes to take effect.

### 8.2.2.2 Start a temporary SQream DB worker with a storage cluster

Starting a SQream DB worker with a custom cluster path can be done in two ways:

#### 8.2.2.2.1 Using a configuration file (recommended)

Similar to the technique above, create a configuration file with the correct cluster path. Then, start `sqreamd` using the `-config` flag:

```
$ sqreamd -config config_file.json
```

#### 8.2.2.2.2 Using the command line parameters

Use sqreamd's command line parameters to override the default storage cluster path:

```
$ sqreamd /home/rhendricks/raviga_database 0 5000 /home/sqream/.sqream/license.enc
```

**Note:** sqreamd's command line parameters' order is `sqreamd <cluster path> <GPU ordinal> <TCP listen port (unsecured)> <License path>`

### 8.2.3 Copying an existing storage cluster

Copying an existing storage cluster to another path may be useful for testing or troubleshooting purposes.

1. Identify the location of the active storage cluster. This path can be found in the configuration file, under the `"cluster"` parameter.
2. Shut down the SQream DB cluster. This prevents very large storage directories from being modified during the copy process.
3. (optional) Create a tarball of the storage cluster, with `tar -zcvf sqream_cluster_`date +"%Y-%m-%d-%H-%M"`.tgz <cluster path>`. This will create a tarball with the current date and time as part of the filename.
4. Copy the storage cluster directory (or tarball) with `cp` to another location on the local filesystem, or use `rsync` to copy to a remote server.
5. After the copy is completed, start the SQream DB cluster to continue using SQream DB.

## 8.3 Foreign Tables

Foreign tables can be used to run queries directly on data without inserting it into SQream DB first. SQream DB supports read only foreign tables, so you can query from foreign tables, but you cannot insert to them, or run deletes or updates on them.

Running queries directly on external data is most effectively used for things like one off querying. If you will be repeatedly querying data, the performance will usually be better if you insert the data into SQream DB first.

Although foreign tables can be used without inserting data into SQream DB, one of their main use cases is to help with the insertion process. An insert select statement on a foreign table can be used to insert data into SQream using the full power of the query engine to perform ETL.

**In this topic:**

### 8.3.1 Supported Data Formats

SQream DB supports foreign tables over:

- Text files (e.g. CSV, PSV, TSV)
- ORC
- Parquet

### 8.3.2 Supported Data Staging

SQream can stage data from:

- a local filesystem (e.g. `/mnt/storage/....`)
- *Inserting Data Using Amazon S3* buckets (e.g. `s3://pp-secret-bucket/users/*.parquet`)
- *Using SQream in an HDFS Environment* (e.g. `hdfs://hadoop-nn.piedpiper.com/rhendricks/*.csv`)

### 8.3.3 Using Foreign Tables

Use a foreign table to stage data before loading from CSV, Parquet or ORC files.

#### 8.3.3.1 Planning for Data Staging

For the following examples, we will want to interact with a CSV file. Here's a peek at the table contents:

Table 1: nba.csv

| Name | Team | Number | Position | Age | Height | Weight | College | Salary |
|---|---|---|---|---|---|---|---|---|
| Avery Bradley | Boston Celtics | 0.0 | PG | 25.0 | 6-2 | 180.0 | Texas | 7730337.0 |
| Jae Crowder | Boston Celtics | 99.0 | SF | 25.0 | 6-6 | 235.0 | Marquette | 6796117.0 |
| John Holland | Boston Celtics | 30.0 | SG | 27.0 | 6-5 | 205.0 | Boston University | |
| R.J. Hunter | Boston Celtics | 28.0 | SG | 22.0 | 6-5 | 185.0 | Georgia State | 1148640.0 |
| Jonas Jerebko | Boston Celtics | 8.0 | PF | 29.0 | 6-10 | 231.0 | | 5000000.0 |
| Amir Johnson | Boston Celtics | 90.0 | PF | 29.0 | 6-9 | 240.0 | | 12000000.0 |
| Jordan Mickey | Boston Celtics | 55.0 | PF | 21.0 | 6-8 | 235.0 | LSU | 1170960.0 |
| Kelly Olynyk | Boston Celtics | 41.0 | C | 25.0 | 7-0 | 238.0 | Gonzaga | 2165160.0 |
| Terry Rozier | Boston Celtics | 12.0 | PG | 22.0 | 6-2 | 190.0 | Louisville | 1824360.0 |

The file is stored on *Inserting Data Using Amazon S3*, at `s3://sqream-demo-data/nba_players.csv`. We will make note of the file structure, to create a matching `CREATE_EXTERNAL_TABLE` statement.

#### 8.3.3.2 Creating a Foreign Table

Based on the source file structure, we we create a foreign table with the appropriate structure, and point it to the file.

```
CREATE foreign table nba
(
   Name varchar,
   Team varchar,
   Number tinyint,
   Position varchar,
   Age tinyint,
   Height varchar,
   Weight real,
   College varchar,
   Salary float
 )
   USING FORMAT CSV -- Text file
   WITH  PATH  's3://sqream-demo-data/nba_players.csv'
   RECORD DELIMITER '\r\n'; -- DOS delimited file
```

The file format in this case is CSV, and it is stored as an *Inserting Data Using Amazon S3* object (if the path is on *Using SQream in an HDFS Environment*, change the URI accordingly).

We also took note that the record delimiter was a DOS newline (`\r\n`).

#### 8.3.3.3 Querying Foreign Tables

Let's peek at the data from the foreign table:

```
t=> SELECT * FROM nba LIMIT 10;
name          | team           | number | position | age | height | weight | college ␣
↪         | salary
--------------+----------------+--------+----------+-----+--------+--------+----------
↪---------+---------
Avery Bradley | Boston Celtics |      0 | PG       |  25 | 6-2    |    180 | Texas    ␣
↪         |  7730337
Jae Crowder   | Boston Celtics |     99 | SF       |  25 | 6-6    |    235 |␣
↪Marquette        |  6796117
John Holland  | Boston Celtics |     30 | SG       |  27 | 6-5    |    205 | Boston␣
↪University |
R.J. Hunter   | Boston Celtics |     28 | SG       |  22 | 6-5    |    185 | Georgia␣
↪State     |  1148640
Jonas Jerebko | Boston Celtics |      8 | PF       |  29 | 6-10   |    231 |          ␣
↪         |  5000000
Amir Johnson  | Boston Celtics |     90 | PF       |  29 | 6-9    |    240 |          ␣
↪         | 12000000
Jordan Mickey | Boston Celtics |     55 | PF       |  21 | 6-8    |    235 | LSU      ␣
↪         |  1170960
Kelly Olynyk  | Boston Celtics |     41 | C        |  25 | 7-0    |    238 | Gonzaga  ␣
↪         |  2165160
Terry Rozier  | Boston Celtics |     12 | PG       |  22 | 6-2    |    190 |␣
↪Louisville       |  1824360
Marcus Smart  | Boston Celtics |     36 | PG       |  22 | 6-4    |    220 | Oklahoma␣
↪State     |  3431040
```

#### 8.3.3.4 Modifying Data from Staging

One of the main reasons for staging data is to examine the contents and modify them before loading them. Assume we are unhappy with weight being in pounds, because we want to use kilograms instead. We can apply the transformation as part of a query:

```
t=> SELECT name, team, number, position, age, height, (weight / 2.205) as weight,␣
↪college, salary
.          FROM nba
.          ORDER BY weight;

name                     | team                   | number | position | age | height␣
↪| weight   | college                | salary
-------------------------+------------------------+--------+----------+-----+--------
↪+----------+-----------------------+---------
Nikola Pekovic           | Minnesota Timberwolves |     14 | C        |  30 | 6-11   ␣
↪|  139.229 |                       | 12100000
Boban Marjanovic         | San Antonio Spurs      |     40 | C        |  27 | 7-3    ␣
↪| 131.5193 |                       |  1200000
Al Jefferson             | Charlotte Hornets      |     25 | C        |  31 | 6-10   ␣
↪| 131.0658 |                       | 13500000
Jusuf Nurkic             | Denver Nuggets         |     23 | C        |  21 | 7-0    ␣
↪| 126.9841 |                       |  1842000
Andre Drummond           | Detroit Pistons        |      0 | C        |  22 | 6-11   ␣
↪| 126.5306 | Connecticut           |  3272091
Kevin Seraphin           | New York Knicks        |      1 | C        |  26 | 6-10   ␣
↪| 126.0771 |                       |  2814000
Brook Lopez              | Brooklyn Nets          |     11 | C        |  28 | 7-0    ␣
↪| 124.7166 | Stanford              | 19689000
Jahlil Okafor            | Philadelphia 76ers     |      8 | C        |  20 | 6-11   ␣
↪| 124.7166 | Duke                  |  4582680
Cristiano Felicio        | Chicago Bulls          |      6 | PF       |  23 | 6-10   ␣
↪| 124.7166 |                       |   525093
[...]
```

Now, if we're happy with the results, we can convert the staged foreign table to a standard table

#### 8.3.3.5 Converting a Foreign Table to a Standard Database Table

create_table_as can be used to materialize a foreign table into a regular table.

**Tip:** If you intend to use the table multiple times, convert the foreign table to a standard table.

```
t=> CREATE TABLE real_nba AS
.     SELECT name, team, number, position, age, height, (weight / 2.205) as weight,␣
↪college, salary
.             FROM nba
.             ORDER BY weight;
executed
t=> SELECT * FROM real_nba LIMIT 5;

name               | team                   | number | position | age | height | weight␣
↪  | college      | salary
-----------------+------------------------+--------+----------+-----+--------+--------
↪--+-------------+---------
```

(continues on next page)

(continued from previous page)

```
Nikola Pekovic   | Minnesota Timberwolves |     14 | C         |  30 | 6-11   |  139.
↪229 |                 | 12100000
Boban Marjanovic | San Antonio Spurs      |     40 | C         |  27 | 7-3    | 131.
↪5193 |                 |  1200000
Al Jefferson     | Charlotte Hornets      |     25 | C         |  31 | 6-10   | 131.
↪0658 |                 | 13500000
Jusuf Nurkic     | Denver Nuggets         |     23 | C         |  21 | 7-0    | 126.
↪9841 |                 |  1842000
Andre Drummond   | Detroit Pistons        |      0 | C         |  22 | 6-11   | 126.
↪5306 | Connecticut |  3272091
```

### 8.3.4 Error Handling and Limitations

- Error handling in foreign tables is limited. Any error that occurs during source data parsing will result in the statement aborting.
- Foreign tables are logical and do not contain any data, their structure is not verified or enforced until a query uses the table. For example, a CSV with the wrong delimiter may cause a query to fail, even though the table has been created successfully:

```
t=> SELECT * FROM nba;
master=> select * from nba;
Record delimiter mismatch during CSV parsing. User defined line delimiter \n does␣
↪not match the first delimiter \r\n found in s3://sqream-demo-data/nba.csv
```

- Since the data for a foreign table is not stored in SQream DB, it can be changed or removed at any time by an external process. As a result, the same query can return different results each time it runs against a foreign table. Similarly, a query might fail if the external data is moved, removed, or has changed structure.

## 8.4 Deleting Data

The **Deleting Data** page describes how the **Delete** statement works and how to maintain data that you delete:

## 8.4.1 Overview

Deleting data typically refers to deleting rows, but can refer to deleting other table content as well. The general workflow for deleting data is to delete data followed by triggering a cleanup operation. The cleanup operation reclaims the space occupied by the deleted rows, discussed further below.

The **DELETE** statement deletes rows defined by a predicate that you have specified, preventing them from appearing in subsequent queries.

For example, the predicate below defines and deletes rows containing animals heavier than 1000 weight units:

```
farm=> DELETE FROM cool_animals WHERE weight > 1000;
```

The major benefit of the DELETE statement is that it deletes transactions simply and quickly.

## 8.4.2 The Deletion Process

Deleting rows occurs in the following two phases:

- **Phase 1 - Deletion** - All rows you mark for deletion are ignored when you run any query. These rows are not deleted until the clean-up phase.

- **Phase 2 - Clean-up** - The rows you marked for deletion in Phase 1 are physically deleted. The clean-up phase is not automated, letting users or DBAs control when to activate it. The files you marked for deletion during Phase 1 are removed from disk, which you do by by sequentially running the utility function commands `CLEANUP_CHUNKS` and `CLEANUP_EXTENTS`.

## 8.4.3 Usage Notes

The **Usage Notes** section includes important information about the DELETE statement:



### 8.4.3.1 General Notes

This section describes the general notes applicable when deleting rows:

- The alter_table command and other DDL operations are locked on tables that require clean-up. If the estimated clean-up time exceeds the permitted threshold, an error message is displayed describing how to override the threshold limitation. For more information, see *Concurrency and Locks*.

- If the number of deleted records exceeds the threshold defined by the `mixedColumnChunksThreshold` parameter, the delete operation is aborted. This alerts users that the large number of deleted records may result in a large number of mixed chunks. To circumvent this alert, use the following syntax (replacing `XXX` with the desired number of records) before running the delete operation:

```
set mixedColumnChunksThreshold=XXX;
```

#### 8.4.3.2 Deleting Data does not Free Space

With the exception of running a full table delete, deleting data does not free unused disk space. To free unused disk space you must trigger the clean-up process.

For more information on running a full table delete, see TRUNCATE.

For more information on freeing disk space, see *Triggering a Clean-Up*.

#### 8.4.3.3 Clean-Up Operations Are I/O Intensive

The clean-up process reduces table size by removing all unused space from column chunks. While this reduces query time, it is a time-costly operation occupying disk space for the new copy of the table until the operation is complete.

**Tip:** Because clean-up operations can create significant I/O load on your database, consider using them sparingly during ideal times.

If this is an issue with your environment, consider using `CREATE TABLE AS` to create a new table and then rename and drop the old table.

### 8.4.4 Examples

The **Examples** section includes the following examples:

- *Deleting Rows from a Table*
- *Deleting Values Based on Complex Predicates*
- *Identifying and Cleaning Up Tables*

#### 8.4.4.1 Deleting Rows from a Table

The following example shows how to delete rows from a table.

1. Display the table:

```
farm=> SELECT * FROM cool_animals;
```

The following table is displayed:

```
1,Dog                 ,7
2,Possum              ,3
3,Cat                 ,5
4,Elephant            ,6500
5,Rhinoceros          ,2100
6,\N,\N
```

2. Delete rows from the table:

```
farm=> DELETE FROM cool_animals WHERE weight > 1000;
```

3. Display the table:

```
farm=> SELECT * FROM cool_animals;
```

The following table is displayed:

```
1,Dog                 ,7
2,Possum              ,3
3,Cat                 ,5
6,\N,\N
```

### 8.4.4.2 Deleting Values Based on Complex Predicates

The following example shows how to delete values based on complex predicates.

1. Display the table:

```
farm=> SELECT * FROM cool_animals;
```

The following table is displayed:

```
1,Dog                 ,7
2,Possum              ,3
3,Cat                 ,5
4,Elephant            ,6500
5,Rhinoceros          ,2100
6,\N,\N
```

2. Delete rows from the table:

```
farm=> DELETE FROM cool_animals WHERE weight > 1000;
```

3. Display the table:

```
farm=> SELECT * FROM cool_animals;
```

The following table is displayed:

```
1,Dog                 ,7
2,Possum              ,3
3,Cat                 ,5
6,\N,\N
```

#### 8.4.4.3 Identifying and Cleaning Up Tables

The **Identifying and Cleaning Up Tables** section includes the following examples:

- *Listing Tables that Have Not Been Cleaned Up*
- *Identifying Predicates for Clean-Up*
- *Triggering a Clean-Up*

##### 8.4.4.3.1 Listing Tables that Have Not Been Cleaned Up

The following example shows how to list tables that have not been cleaned up:

```
farm=> SELECT t.table_name FROM sqream_catalog.delete_predicates dp
   JOIN sqream_catalog.tables t
   ON dp.table_id = t.table_id
   GROUP BY 1;
cool_animals

1 row
```

##### 8.4.4.3.2 Identifying Predicates for Clean-Up

The following example shows how to identify predicates for clean-up:

```
farm=> SELECT delete_predicate FROM sqream_catalog.delete_predicates dp
   JOIN sqream_catalog.tables t
   ON dp.table_id = t.table_id
   WHERE t.table_name = 'cool_animals';
weight > 1000

1 row
```

##### 8.4.4.3.3 Triggering a Clean-Up

The following example shows how to trigger a clean-up:

1. Run the chunk `CLEANUP_CHUNKS` command (also known as `SWEEP`) to reorganize the chunks:

   ```
   farm=> SELECT CLEANUP_CHUNKS('public','cool_animals');
   ```

2. Run the `CLEANUP_EXTENTS` command (also known as `VACUUM`) to delete the leftover files:

   ```
   farm=> SELECT CLEANUP_EXTENTS('public','cool_animals');
   ```

3. Display the table:

   ```
   farm=> SELECT delete_predicate FROM sqream_catalog.delete_predicates dp
      JOIN sqream_catalog.tables t
      ON dp.table_id = t.table_id
      WHERE t.table_name = 'cool_animals';
   ```

### 8.4.5 Best Practices

This section includes the best practices when deleting rows:

- Run `CLEANUP_CHUNKS` and `CLEANUP_EXTENTS` after running large `DELETE` operations.

- When you delete large segments of data from very large tables, consider running a `CREATE TABLE AS` operation instead, renaming, and dropping the original table.

- Avoid killing `CLEANUP_EXTENTS` operations in progress.

- SQream is optimized for time-based data, which is data naturally ordered according to date or timestamp. Deleting rows based on such columns leads to increased performance.

## 8.5 Exporting Data

You can export data from SQream, which you may want to do for the following reasons:

- To use data in external tables. See Working with External Data.
- To share data with other clients or consumers with different systems.
- To copy data into another SQream cluster.

SQream provides the following methods for exporting data:

- Copying data from a SQream database table or query to another file - See COPY TO.

## 8.6 Logging

### 8.6.1 Locating the Log Files

The *storage cluster* contains a `logs` directory. Each worker produces a log file in its own directory, which can be identified by the worker's hostname and port.

**Note:** Additional internal debug logs may reside in the main `logs` directory.

The worker logs contain information messages, warnings, and errors pertaining to SQream DB's operation, including:

- Server start-up and shutdown
- Configuration changes
- Exceptions and errors
- User login events
- Session events
- Statement execution success / failure
- Statement execution statistics

#### 8.6.1.1 Log Structure and Contents

The log is a CSV, with several fields.

Table 2: Log fields

| Field | Description |
|---|---|
| `#SQ#` | Start delimiter. When used with the end of line delimiter can be used to parse multi-line statements correctly |
| Row Id | Unique identifier for the row |
| Timestamp | Timestamp for the message (ISO 8601 date format) |
| Information Level | Information level of the message. See *information level table* below |
| Thread Id | System thread identifier (internal use) |
| Worker hostname | Hostname of the worker that generated the message |
| Worker port | Port of the worker that generated the message |
| Connection Id | Connection Id for the message. Defaults to `-1` if no connection |
| Database name | Database name that generated the message. Can be empty for no database |
| User Id | User role that was connected during the message. Can be empty if no user caused the message |
| Statement Id | Statement Id for the message. Defaults to `-1` if no statement |
| Service name | Service name for the connection. Can be empty. |
| Message type Id | Message type Id. See *message type table* below) |
| Message | Content for the message |
| `#EOM#` | End of line delimiter |

Table 3: Information Level

| Level | Description |
|---|---|
| `SYSTEM` | System information like start up, shutdown, configuration change |
| `FATAL` | Fatal errors that may cause outage |
| `ERROR` | Errors encountered during statement execution |
| `WARNING` | Warnings |
| `INFO` | Information and statistics |

Table 4: Message Type

| Type | Level | Description | Example message content |
|---|---|---|---|
| 1 | INFO | Statement start information | • "Query before parsing (statement handle opened)<br>• "SELECT * FROM nba WHERE ""Team"" NOT LIKE ""Portland%%""" (statement preparing) |
| 2 | INFO | Statement passed to another worker for execution | • ""Reconstruct query before parsing"<br>• "SELECT * FROM nba WHERE ""Team"" NOT LIKE ""Portland%%""" (statement preparing on node) |
| 4 | INFO | Statement has entered execution | "Statement execution" |
| 10 | INFO | Statement execution completed | "Success" / "Failed" |
| 20 | INFO | Compilation error, with accompanying error message | "Could not find function dateplart in catalog." |
| 21 | INFO | Execution error, with accompanying error message | Error text |
| 30 | INFO | Size of data read from disk in megabytes | 18 |
| 31 | INFO | Row count of result set | 45 |
| 32 | INFO | Processed Rows | 450134749978 |
| 100 | INFO | Session start - Client IP address | "192.168.5.5" |
| 101 | INFO | Login | "Login Success" / "Login Failed" |
| 110 | INFO | Session end | "Session ended" |
| 200 | INFO | show_node_info periodic output | |
| 500 | ERROR | Exception occured in a statement | "Cannot return the inverse cosine of a number not in [-1,1] range" |
| 1000 | SYSTEM | Worker startup message | "Server Start Time - 2019-12-30 21:18:31, SQream ver{v2020.2}" |
| 1002 | SYSTEM | Metadata | Metadata server |

#### 8.6.1.2 Log-Naming

Log file name syntax

`sqream_<date>_<sequence>.log`

- `date` is formatted `%y%m%d`, for example `20191231` for December 31st 2019.

  By default, each worker will create a new log file every time it is restarted.
- `sequence` is the log's sequence. When a log is rotated, the sequence number increases. This starts at `000`.

For example, `/home/rhendricks/sqream_storage/192.168.1.91_5000`.

See the *Changing Log Rotation* below for information about controlling this setting.

### 8.6.2 Log Control and Maintenance

#### 8.6.2.1 Changing Log Verbosity

A few configuration settings alter the verbosity of the logs:

Table 5: Log verbosity configuration

| Flag | Description | Default | Values |
|---|---|---|---|
| `logClientLevel` | Used to control which log level should appear in the logs | `4` (`INFO`) | `0` SYSTEM (lowest) - `4` INFO (highest). See *information level table* above. |
| `nodeInfoLoggingSec` | Sets an interval for automatically logging long-running statements' show_node_info output. Output is written as a message type `200`. | `60` (every minute) | Positive whole number >=1. |

#### 8.6.2.2 Changing Log Rotation

A few configuration settings alter the log rotation policy:

Table 6: Log rotation configuration

| Flag | Description | Default | Values |
|---|---|---|---|
| `useLogMaxFileSize` | Rotate log files once they reach a certain file size. When `true`, set the `logMaxFileSizeMB` accordingly. | `false` | `false` or `true`. |
| `logMaxFileSizeMB` | Sets the size threshold in megabytes after which a new log file will be opened. | `20` | `1` to `1024` (1MB to 1GB) |
| `logFileRotateTimeFrequency` | Frequency of log rotation | `never` | `daily`, `weekly`, `monthly`, `never` |

### 8.6.3 Collecting Logs from Your Cluster

Collecting logs from your cluster can be as simple as creating an archive from the `logs` subdirectory: `tar -czvf logs.tgz *.log`.

However, SQream DB comes bundled with a data collection utility and an SQL utility intended for collecting logs and additional information that can help SQream support drill down into possible issues.

#### 8.6.3.1 SQL Syntax

```
SELECT REPORT_COLLECTION(output_path, mode)
;

output_path ::=
   filepath

mode ::=
   log | db | db_and_log
```

#### 8.6.3.2 Command Line Utility

If you cannot access SQream DB for any reason, you can also use a command line toolto collect the same information:

```
$ ./bin/report_collection <path to storage> <path for output> <mode>
```

#### 8.6.3.3 Parameters

| Parameter | Description |
|---|---|
| `output_path` | Path for the output archive. The output file will be named `report_<date>_<time>.tar`. |
| `mode` | One of three modes: * `'log'` - Collects all log files * `'db'` - Collects the metadata database (includes DDL, but no data) * `'db_and_log'` - Collect both log files and metadata database |

#### 8.6.3.4 Example

Write an archive to `/home/rhendricks`, containing log files:

```
SELECT REPORT_COLLECTION('/home/rhendricks', 'log')
;
```

Write an archive to `/home/rhendricks`, containing log files and metadata database:

```
SELECT REPORT_COLLECTION('/home/rhendricks', 'db_and_log')
;
```

Using the command line utility:

```
$ ./bin/report_collection /home/rhendricks/sqream_storage /home/rhendricks db_and_log
```

### 8.6.4 Troubleshooting with Logs

#### 8.6.4.1 Loading Logs with Foreign Tables

Assuming logs are stored at `/home/rhendricks/sqream_storage/logs/`, a database administrator can access the logs using the external_tables concept through SQream DB.

```
CREATE FOREIGN TABLE logs
(
  start_marker      TEXT(4),
  row_id            BIGINT,
  timestamp         DATETIME,
  message_level     TEXT,
  thread_id         TEXT,
  worker_hostname   TEXT,
  worker_port       INT,
  connection_id     INT,
  database_name     TEXT,
  user_name         TEXT,
  statement_id      INT,
  service_name      TEXT,
  message_type_id   INT,
  message           TEXT,
  end_message       TEXT(5)
)
WRAPPER csv_fdw
OPTIONS
  (
     LOCATION = '/home/rhendricks/sqream_storage/logs/**/sqream*.log',
     DELIMITER = '|'
     CONTINUE_ON_ERROR = true
  )
;
```

For more information, see Loading Logs with Foreign Tables.

#### 8.6.4.2 Counting Message Types

```
t=> SELECT message_type_id, COUNT(*) FROM logs GROUP BY 1;
message_type_id | count
----------------+----------
              0 |         9
              1 |      5578
              4 |      2319
             10 |      2788
             20 |       549
             30 |       411
             31 |      1720
             32 |      1720
            100 |      2592
            101 |      2598
            110 |      2571
            200 |        11
            500 |       136
           1000 |        19
           1003 |        19
```

(continues on next page)

(continued from previous page)

```
           1004 |          19
           1010 |           5
```

### 8.6.4.3 Finding Fatal Errors

```
t=> SELECT message FROM logs WHERE message_type_id=1010;
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/leveldb/LOCK: Resource temporarily unavailable
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/leveldb/LOCK: Resource temporarily unavailable
Mismatch in storage version, upgrade is needed,Storage version: 25, Server version␣
↪is: 26
Mismatch in storage version, upgrade is needed,Storage version: 25, Server version␣
↪is: 26
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/LOCK: Resource temporarily unavailable
```

### 8.6.4.4 Countng Error Events Within a Certain Timeframe

```
t=> SELECT message_type_id,
.          COUNT(*)
.   FROM logs
.   WHERE message_type_id IN (1010,500)
.   AND timestamp BETWEEN '2019-12-20' AND '2020-01-01'
.   GROUP BY 1;
message_type_id | count
----------------+------
            500 |    18
           1010 |     3
```

### 8.6.4.5 Tracing Errors to Find Offending Statements

If we know an error occured, but don't know which statement caused it, we can find it using the connection ID and statement ID.

```
t=> SELECT connection_id, statement_id, message
.     FROM logs
.     WHERE message_level = 'ERROR'
.     AND timestamp BETWEEN '2020-01-01' AND '2020-01-06';
connection_id | statement_id | message
--------------+--------------+--------------------------------------------------------
↪--------------------------------------------------------------------------------------
↪----------------------
           79 |           67 | Column type mismatch, expected UByte, got INT64 on␣
↪column Number, file name: /home/sqream/nba.parquet
```

Use the `connection_id` and `statement_id` to narrow down the results.

```
t=>   SELECT database_name, message FROM logs
.       WHERE connection_id=79 AND statement_id=67 AND message_type_id=1;
database_name | message
```

(continues on next page)

(continued from previous page)

```
--------------+--------------------------
master        | Query before parsing
master        | SELECT * FROM nba_parquet
```

## 8.7 Monitoring Query Performance

When analyzing options for query tuning, the first step is to analyze the query plan and execution. The query plan and execution details explains how SQream DB processes a query and where time is spent. This document details how to analyze query performance with execution plans. This guide focuses specifically on identifying bottlenecks and possible optimization techniques to improve query performance. Performance tuning options for each query are different. You should adapt the recommendations and tips for your own workloads. See also our *Optimization and Best Practices* guide for more information about data loading considerations and other best practices.

**In this section:**

### 8.7.1 Setting Up the System for Monitoring

By default, SQream DB logs execution details for every statement that runs for more than 60 seconds. If you want to see the execution details for a currently running statement, see *Using the SHOW_NODE_INFO Command* below.

#### 8.7.1.1 Adjusting the Logging Frequency

To adjust the frequency of logging for statements, you may want to reduce the interval from 60 seconds down to, say, 5 or 10 seconds. Modify the configuration files and set the `nodeInfoLoggingSec` parameter as you see fit:

```
{
   "compileFlags":{
   },
   "runtimeFlags":{
   },
   "runtimeGlobalFlags":{
      "nodeInfoLoggingSec" : 5,
   },
   "server":{
   }
}
```

After restarting the SQream DB cluster, the execution plan details will be logged to the *standard SQream DB logs directory*, as a message of type `200`. You can see these messages with a text viewer or with queries on the log external_tables.

#### 8.7.1.2 Reading Execution Plans with a Foreign Table

First, create a foreign table for the logs

Once you've defined the foreign table, you can run queries to observe the previously logged execution plans. This is recommended over looking at the raw logs.

### 8.7.2 Using the `SHOW_NODE_INFO` Command

The show_node_info command returns a snapshot of the current query plan, similar to `EXPLAIN ANALYZE` from other databases. The show_node_info result, just like the periodically-logged execution plans described above, are an at-the-moment view of the compiler's execution plan and runtime statistics for the specified statement. To inspect a currently running statement, execute the `show_node_info` utility function in a SQL client like

*sqream sql*, the SQream Studio Editor, or any other third party SQL terminal.

In this example, we inspect a statement with statement ID of 176. The command looks like this:

```
t=> SELECT SHOW_NODE_INFO(176);
stmt_id | node_id | node_type          | rows | chunks | avg_rows_in_chunk | time     ␣
↪           | parent_node_id | read | write | comment     | timeSum
--------+---------+--------------------+------+--------+-------------------+----------
↪-----------+----------------+------+-------+------------+--------
    176 |       1 | PushToNetworkQueue |    1 |      1 |                 1 | 2019-12-
↪25 23:53:13 |             -1 |      |       |            |  0.0025
    176 |       2 | Rechunk            |    1 |      1 |                 1 | 2019-12-
↪25 23:53:13 |              1 |      |       |            |       0
    176 |       3 | GpuToCpu           |    1 |      1 |                 1 | 2019-12-
↪25 23:53:13 |              2 |      |       |            |       0
    176 |       4 | ReorderInput       |    1 |      1 |                 1 | 2019-12-
↪25 23:53:13 |              3 |      |       |            |       0
    176 |       5 | Filter             |    1 |      1 |                 1 | 2019-12-
↪25 23:53:13 |              4 |      |       |            |  0.0002
    176 |       6 | GpuTransform       |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              5 |      |       |            |  0.0002
    176 |       7 | GpuDecompress      |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              6 |      |       |            |       0
    176 |       8 | CpuToGpu           |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              7 |      |       |            |  0.0003
    176 |       9 | Rechunk            |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              8 |      |       |            |       0
    176 |      10 | CpuDecompress      |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |              9 |      |       |            |       0
    176 |      11 | ReadTable          |  457 |      1 |               457 | 2019-12-
↪25 23:53:13 |             10 | 4MB  |       | public.nba |  0.0004
```

### 8.7.3 Understanding the Query Execution Plan Output

Both show_node_info and the logged execution plans represents the query plan as a graph hierarchy, with data separated into different columns. Each row represents a single logical database operation, which is also called a **node** or **chunk producer**. A node reports several metrics during query execution, such as how much data it has read and written, how many chunks and rows, and how much time has elapsed. Consider the example show_node_info presented above. The source node with ID #11 (`ReadTable`), has a parent node ID #10 (`CpuDecompress`). If we were to draw this out in a graph, it'd look like this: .. figure:: /_static/images/show_node_info_graph.png

**height** 70em

**align** center

This graph explains how the query execution details are arranged in a logical order, from the bottom up.

The last node, also called the sink, has a parent node ID of -1, meaning it has no parent. This is typically a node that sends data over the network or into a table.

When using show_node_info, a tabular representation of the currently running statement execution is presented. See the examples below to understand how the query execution plan is instrumental in identifying bottlenecks and optimizing long-running statements.

#### 8.7.3.1 Information Presented in the Execution Plan

#### 8.7.3.2 Commonly Seen Nodes

Table 7: Node types

| Column name | Execution location | Description |
|---|---|---|
| `CpuDecompress` | CPU | Decompression operation, common for longer `TEXT` types |
| `CpuLoopJoin` | CPU | A non-indexed nested loop join, performed on the CPU |
| `CpuReduce` | CPU | A reduce process performed on the CPU, primarily with `DISTINCT` aggregates (e |
| `CpuToGpu`, `GpuToCpu` | | An operation that moves data to or from the GPU for processing |
| `CpuTransform` | CPU | A transform operation performed on the CPU, usually a *scalar function* |
| `DeferredGather` | CPU | Merges the results of GPU operations with a result set |
| `Distinct` | GPU | Removes duplicate rows (usually as part of the `DISTINCT` operation) |
| `Distinct_Merge` | CPU | The merge operation of the `Distinct` operation |
| `Filter` | GPU | A filtering operation, such as a `WHERE` or `JOIN` clause |
| `GpuDecompress` | GPU | Decompression operation |
| `GpuReduceMerge` | GPU | An operation to optimize part of the merger phases in the GPU |
| `GpuTransform` | GPU | A transformation operation such as a type cast or *scalar function* |
| `LocateFiles` | CPU | Validates external file paths for foreign data wrappers, expanding directories and G |
| `LoopJoin` | GPU | A non-indexed nested loop join, performed on the GPU |
| `ParseCsv` | CPU | A CSV parser, used after `ReadFiles` to convert the CSV into columnar data |
| `PushToNetworkQueue` | CPU | Sends result sets to a client connected over the network |
| `ReadFiles` | CPU | Reads external flat-files |
| `ReadTable` | CPU | Reads data from a standard table stored on disk |
| `Rechunk` | | Reorganize multiple small chunks into a full chunk. Commonly found after joins a |
| `Reduce` | GPU | A reduction operation, such as a `GROUP BY` |
| `ReduceMerge` | GPU | A merge operation of a reduction operation, helps operate on larger-than-RAM dat |
| `ReorderInput` | | Change the order of arguments in preparation for the next operation |
| `SeparatedGather` | GPU | Gathers additional columns for the result |
| `Sort` | GPU | Sort operation |
| `TakeRowsFromChunk` | | Take the first N rows from each chunk, to optimize `LIMIT` when used alongside `O` |
| `Top` | | Limits the input size, when used with `LIMIT` (or its alias `TOP`) |
| `UdfTransform` | CPU | Executes a *user defined function* |
| `UnionAll` | | Combines two sources of data when `UNION ALL` is used |
| `Window` | GPU | Executes a non-ranking window function |
| `WindowRanking` | GPU | Executes a ranking window function |
| `WriteTable` | CPU | Writes the result set to a standard table stored on disk |

**Tip:** The full list of nodes appears in the Node types table, as part of the show_node_info reference.

### 8.7.4 Examples

In general, looking at the top three longest running nodes (as is detailed in the `timeSum` column) can indicate the biggest bottlenecks. In the following examples you will learn how to identify and solve some common issues.

**In this section:**

#### 8.7.4.1 1. Spooling to Disk

When there is not enough RAM to process a statement, SQream DB will spill over data to the `temp` folder in the storage disk. While this ensures that a statement can always finish processing, it can slow down the processing significantly. It's worth identifying these statements, to figure out if the cluster is configured correctly, as well as potentially reduce the statement size. You can identify a statement that spools to disk by looking at the `write` column in the execution details. A node that spools will have a value, shown in megabytes in the `write` column. Common nodes that write spools include `Join` or `LoopJoin`.

##### 8.7.4.1.1 Identifying the Offending Nodes

1. Run a query.

   For example, a query from the TPC-H benchmark:

```
SELECT o_year,
       SUM(CASE WHEN nation = 'BRAZIL' THEN volume ELSE 0 END) / SUM(volume) AS␣
↪mkt_share
FROM (SELECT datepart(YEAR,o_orderdate) AS o_year,
             l_extendedprice*(1 - l_discount / 100.0) AS volume,
             n2.n_name AS nation
      FROM lineitem
        JOIN part ON p_partkey = CAST (l_partkey AS INT)
        JOIN orders ON l_orderkey = o_orderkey
        JOIN customer ON o_custkey = c_custkey
        JOIN nation n1 ON c_nationkey = n1.n_nationkey
        JOIN region ON n1.n_regionkey = r_regionkey
        JOIN supplier ON s_suppkey = l_suppkey
        JOIN nation n2 ON s_nationkey = n2.n_nationkey
      WHERE o_orderdate BETWEEN '1995-01-01' AND '1996-12-31') AS all_nations
GROUP BY o_year
ORDER BY o_year;
```

2. Observe the execution information by using the foreign table, or use `show_node_info`

   This statement is made up of 199 nodes, starting from a `ReadTable`, and finishes by returning only 2 results to the client.

   The execution below has been shortened, but note the highlighted rows for `LoopJoin`:

```
t=> SELECT message FROM logs WHERE message_type_id = 200 LIMIT 1;
message
-----------------------------------------------------------------------------------
↪-------
SELECT o_year,
       SUM(CASE WHEN nation = 'BRAZIL' THEN volume ELSE 0 END) / SUM(volume) AS␣
↪mkt_share
 : FROM (SELECT datepart(YEAR,o_orderdate) AS o_year,
 :              l_extendedprice*(1 - l_discount / 100.0) AS volume,
 :              n2.n_name AS nation
 :        FROM lineitem
 :          JOIN part ON p_partkey = CAST (l_partkey AS INT)
 :          JOIN orders ON l_orderkey = o_orderkey
 :          JOIN customer ON o_custkey = c_custkey
 :          JOIN nation n1 ON c_nationkey = n1.n_nationkey
 :          JOIN region ON n1.n_regionkey = r_regionkey
 :          JOIN supplier ON s_suppkey = l_suppkey
```

(continues on next page)

(continued from previous page)

```
:         JOIN nation n2 ON s_nationkey = n2.n_nationkey
:       WHERE o_orderdate BETWEEN '1995-01-01' AND '1996-12-31') AS all_nations
: GROUP BY o_year
: ORDER BY o_year
: 1,PushToNetworkQueue  ,2,1,2,2020-09-04 18:32:50,-1,,,,0.27
: 2,Rechunk             ,2,1,2,2020-09-04 18:32:50,1,,,,0.00
: 3,SortMerge           ,2,1,2,2020-09-04 18:32:49,2,,,,0.00
: 4,GpuToCpu            ,2,1,2,2020-09-04 18:32:49,3,,,,0.00
: 5,Sort                ,2,1,2,2020-09-04 18:32:49,4,,,,0.00
: 6,ReorderInput        ,2,1,2,2020-09-04 18:32:49,5,,,,0.00
: 7,GpuTransform        ,2,1,2,2020-09-04 18:32:49,6,,,,0.00
: 8,CpuToGpu            ,2,1,2,2020-09-04 18:32:49,7,,,,0.00
: 9,Rechunk             ,2,1,2,2020-09-04 18:32:49,8,,,,0.00
: 10,ReduceMerge         ,2,1,2,2020-09-04 18:32:49,9,,,,0.03
: 11,GpuToCpu            ,6,3,2,2020-09-04 18:32:49,10,,,,0.00
: 12,Reduce              ,6,3,2,2020-09-04 18:32:49,11,,,,0.64
[...]
: 49,LoopJoin            ,182369485,7,26052783,2020-09-04 18:32:36,48,1915MB,
↪1915MB,inner,4.94
[...]
: 98,LoopJoin            ,182369485,12,15197457,2020-09-04 18:32:16,97,2191MB,
↪2191MB,inner,5.01
[...]
: 124,LoopJoin            ,182369485,8,22796185,2020-09-04 18:32:03,123,3064MB,
↪3064MB,inner,6.73
[...]
: 150,LoopJoin            ,182369485,10,18236948,2020-09-04 18:31:47,149,12860MB,
↪12860MB,inner,23.62
[...]
: 199,ReadTable           ,20000000,1,20000000,2020-09-04 18:30:33,198,0MB,,
↪public.part,0.83
```

Because of the relatively low amount of RAM in the machine and because the data set is rather large at around 10TB, SQream DB needs to spool.

The total spool used by this query is around 20GB (1915MB + 2191MB + 3064MB + 12860MB).

#### 8.7.4.1.2 Common Solutions for Reducing Spool

- Increase the amount of spool memory available for the workers, as a proportion of the maximum statement memory. When the amount of spool memory is increased, SQream DB may not need to write to disk.

  This setting is called `spoolMemoryGB`. Refer to the configuration guide.

- Reduce the amount of **workers** per host, and increase the amount of spool available to the (now reduced amount of) active workers. This may reduce the amount of concurrent statements, but will improve performance for heavy statements.

#### 8.7.4.2 2. Queries with Large Result Sets

When queries have large result sets, you may see a node called `DeferredGather`. This gathering occurs when the result set is assembled, in preparation for sending it to the client.

##### 8.7.4.2.1 Identifying the Offending Nodes

1. Run a query.

   For example, a modified query from the TPC-H benchmark:

```
SELECT s.*,
       l.*,
       r.*,
       n1.*,
       n2.*,
       p.*,
       o.*,
       c.*
FROM lineitem l
  JOIN part p ON p_partkey = CAST (l_partkey AS INT)
  JOIN orders o ON l_orderkey = o_orderkey
  JOIN customer c ON o_custkey = c_custkey
  JOIN nation n1 ON c_nationkey = n1.n_nationkey
  JOIN region r ON n1.n_regionkey = r_regionkey
  JOIN supplier s ON s_suppkey = l_suppkey
  JOIN nation n2 ON s_nationkey = n2.n_nationkey
WHERE r_name = 'AMERICA'
AND   o_orderdate BETWEEN '1995-01-01' AND '1996-12-31'
AND   high_selectivity(p_type = 'ECONOMY BURNISHED NICKEL');
```

2. Observe the execution information by using the foreign table, or use `show_node_info`

   This statement is made up of 221 nodes, containing 8 `ReadTable` nodes, and finishes by returning billions of results to the client.

   The execution below has been shortened, but note the highlighted rows for `DeferredGather`:

```
t=> SELECT show_node_info(494);
stmt_id | node_id | node_type            | rows      | chunks | avg_rows_in_chunk␣
↪| time                | parent_node_id | read     | write | comment          |␣
↪timeSum
--------+---------+----------------------+-----------+--------+-------------------
↪+---------------------+----------------+----------+-------+------------------+----
↪----
    494 |       1 | PushToNetworkQueue   |    242615 |      1 |            242615␣
↪| 2020-09-04 19:07:55 |             -1 |          |       |                  |   ␣
↪0.36
    494 |       2 | Rechunk              |    242615 |      1 |            242615␣
↪| 2020-09-04 19:07:55 |              1 |          |       |                  |   ␣
↪   0
    494 |       3 | ReorderInput         |    242615 |      1 |            242615␣
↪| 2020-09-04 19:07:55 |              2 |          |       |                  |   ␣
↪   0
    494 |       4 | DeferredGather       |    242615 |      1 |            242615␣
↪| 2020-09-04 19:07:55 |              3 |          |       |                  |   ␣
↪0.16
```

(continues on next page)

(continued from previous page)

```
    [...]
    494 |      166 | DeferredGather         |   3998730 |      39 |              102531␣
↪| 2020-09-04 19:07:47 |             165 |           |         |                  |   ␣
↪21.75
    [...]
    494 |      194 | DeferredGather         |    133241 |      20 |                6662␣
↪| 2020-09-04 19:07:03 |             193 |           |         |                  |   ␣
↪0.41
    [...]
    494 |      221 | ReadTable              |  20000000 |      20 |             1000000␣
↪| 2020-09-04 19:07:01 |             220 | 20MB      |         | public.part      |   ␣
↪ 0.1
```

When you see `DeferredGather` operations taking more than a few seconds, that's a sign that you're selecting too much data. In this case, the DeferredGather with node ID 166 took over 21 seconds.

3. Modify the statement to see the difference Altering the select clause to be more restrictive will reduce the deferred gather time back to a few milliseconds.

```
SELECT DATEPART(year, o_orderdate) AS o_year,
       l_extendedprice * (1 - l_discount / 100.0) as volume,
       n2.n_name as nation
FROM ...
```

#### 8.7.4.2.2 Common Solutions for Reducing Gather Time

- Reduce the effect of the preparation time. Avoid selecting unnecessary columns (`SELECT * FROM...`), or reduce the result set size by using more filters.

### 8.7.4.3 3. Inefficient Filtering

When running statements, SQream DB tries to avoid reading data that is not needed for the statement by skipping chunks. If statements do not include efficient filtering, SQream DB will read a lot of data off disk. In some cases, you need the data and there's nothing to do about it. However, if most of it gets pruned further down the line, it may be efficient to skip reading the data altogether by using the metadata.

#### 8.7.4.3.1 Identifying the Situation

We consider the filtering to be inefficient when the `Filter` node shows that the number of rows processed is less than a third of the rows passed into it by the `ReadTable` node. For example: #.

Run a query.

In this example, we execute a modified query from the TPC-H benchmark. Our `lineitem` table contains 600,037,902 rows.

```
SELECT o_year,
       SUM(CASE WHEN nation = 'BRAZIL' THEN volume ELSE 0 END) / SUM(volume)␣
↪AS mkt_share
FROM (SELECT datepart(YEAR,o_orderdate) AS o_year,
             l_extendedprice*(1 - l_discount / 100.0) AS volume,
             n2.n_name AS nation
```

(continues on next page)

(continued from previous page)

```
      FROM lineitem
        JOIN part ON p_partkey = CAST (l_partkey AS INT)
        JOIN orders ON l_orderkey = o_orderkey
        JOIN customer ON o_custkey = c_custkey
        JOIN nation n1 ON c_nationkey = n1.n_nationkey
        JOIN region ON n1.n_regionkey = r_regionkey
        JOIN supplier ON s_suppkey = l_suppkey
        JOIN nation n2 ON s_nationkey = n2.n_nationkey
      WHERE r_name = 'AMERICA'
      AND   lineitem.l_quantity = 3
      AND   o_orderdate BETWEEN '1995-01-01' AND '1996-12-31'
      AND   high_selectivity(p_type = 'ECONOMY BURNISHED NICKEL')) AS all_
↪nations
GROUP BY o_year
ORDER BY o_year;
```

1. Observe the execution information by using the foreign table, or use `show_node_info`

   The execution below has been shortened, but note the highlighted rows for `ReadTable` and `Filter`:

```
t=> SELECT show_node_info(559);
stmt_id | node_id | node_type             | rows      | chunks | avg_rows_in_chunk␣
↪| time                | parent_node_id | read    | write | comment            |␣
↪timeSum
--------+---------+-----------------------+-----------+--------+------------------
↪+--------------------+----------------+---------+-------+-----------------+-----
↪---
    559 |       1 | PushToNetworkQueue    |         2 |      1 |                 2␣
↪| 2020-09-07 11:12:01 |             -1 |         |       |                 |   ␣
↪0.28
    559 |       2 | Rechunk               |         2 |      1 |                 2␣
↪| 2020-09-07 11:12:01 |              1 |         |       |                 |   ␣
↪   0
    559 |       3 | SortMerge             |         2 |      1 |                 2␣
↪| 2020-09-07 11:12:01 |              2 |         |       |                 |   ␣
↪   0
    559 |       4 | GpuToCpu              |         2 |      1 |                 2␣
↪| 2020-09-07 11:12:01 |              3 |         |       |                 |   ␣
↪   0
[...]
    559 |     189 | Filter                |  12007447 |     12 |           1000620␣
↪| 2020-09-07 11:12:00 |            188 |         |       |                 |   ␣
↪0.3
    559 |     190 | GpuTransform          | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            189 |         |       |                 |   ␣
↪0.02
    559 |     191 | GpuDecompress         | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            190 |         |       |                 |   ␣
↪0.16
    559 |     192 | GpuTransform          | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            191 |         |       |                 |   ␣
↪0.02
    559 |     193 | CpuToGpu              | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            192 |         |       |                 |   ␣
↪1.47
    559 |     194 | ReorderInput          | 600037902 |     12 |          50003158␣
↪| 2020-09-07 11:12:00 |            193 |         |       |                 |   ␣
↪   0
```

(continues on next page)

(continued from previous page)

```
15     559 |      195 | Rechunk                     | 600037902 |      12 |              50003158 | 2020-09-07 11:12:00 |             194 |           |         |                   |     0
16     559 |      196 | CpuDecompress               | 600037902 |      12 |              50003158 | 2020-09-07 11:12:00 |             195 |           |         |                   |     0
17     559 |      197 | ReadTable                   | 600037902 |      12 |              50003158 | 2020-09-07 11:12:00 |             196 | 7587MB    |         | public.lineitem   |   0.1
18 [...]
19     559 |      208 | Filter                      |    133241 |      20 |                  6662 | 2020-09-07 11:11:57 |             207 |           |         |                   |  0.01
20     559 |      209 | GpuTransform                |  20000000 |      20 |               1000000 | 2020-09-07 11:11:57 |             208 |           |         |                   |  0.02
21     559 |      210 | GpuDecompress               |  20000000 |      20 |               1000000 | 2020-09-07 11:11:57 |             209 |           |         |                   |  0.03
22     559 |      211 | GpuTransform                |  20000000 |      20 |               1000000 | 2020-09-07 11:11:57 |             210 |           |         |                   |     0
23     559 |      212 | CpuToGpu                    |  20000000 |      20 |               1000000 | 2020-09-07 11:11:57 |             211 |           |         |                   |  0.01
24     559 |      213 | ReorderInput                |  20000000 |      20 |               1000000 | 2020-09-07 11:11:57 |             212 |           |         |                   |     0
25     559 |      214 | Rechunk                     |  20000000 |      20 |               1000000 | 2020-09-07 11:11:57 |             213 |           |         |                   |     0
26     559 |      215 | CpuDecompress               |  20000000 |      20 |               1000000 | 2020-09-07 11:11:57 |             214 |           |         |                   |     0
27     559 |      216 | ReadTable                   |  20000000 |      20 |               1000000 | 2020-09-07 11:11:57 |             215 | 20MB      |         | public.part       |     0
```

- The `Filter` on line 9 has processed 12,007,447 rows, but the output of `ReadTable` on `public.lineitem` on line 17 was 600,037,902 rows. This means that it has filtered out 98% ($1 - \frac{600037902}{12007447} = 98\%$) of the data, but the entire table was read.
- The `Filter` on line 19 has processed 133,000 rows, but the output of `ReadTable` on `public.part` on line 27 was 20,000,000 rows. This means that it has filtered out >99% ($1 - \frac{133241}{20000000} = 99.4\%$) of the data, but the entire table was read. However, this table is small enough that we can ignore it.

2. Modify the statement to see the difference Altering the statement to have a `WHERE` condition on the clustered `l_orderkey` column of the `lineitem` table will help SQream DB skip reading the data.

```
SELECT o_year,
       SUM(CASE WHEN nation = 'BRAZIL' THEN volume ELSE 0 END) / SUM(volume) AS mkt_share
FROM (SELECT datepart(YEAR,o_orderdate) AS o_year,
             l_extendedprice*(1 - l_discount / 100.0) AS volume,
             n2.n_name AS nation
      FROM lineitem
```

(continues on next page)

(continued from previous page)

```
         JOIN part ON p_partkey = CAST (l_partkey AS INT)
         JOIN orders ON l_orderkey = o_orderkey
         JOIN customer ON o_custkey = c_custkey
         JOIN nation n1 ON c_nationkey = n1.n_nationkey
         JOIN region ON n1.n_regionkey = r_regionkey
         JOIN supplier ON s_suppkey = l_suppkey
         JOIN nation n2 ON s_nationkey = n2.n_nationkey
       WHERE r_name = 'AMERICA'
       AND   lineitem.l_orderkey > 4500000
       AND   o_orderdate BETWEEN '1995-01-01' AND '1996-12-31'
       AND   high_selectivity(p_type = 'ECONOMY BURNISHED NICKEL')) AS all_nations
GROUP BY o_year
ORDER BY o_year;
```

```
t=> SELECT show_node_info(586);
stmt_id | node_id | node_type            | rows      | chunks | avg_rows_in_chunk␣
↪| time                | parent_node_id | read    | write | comment          |␣
↪timeSum
--------+---------+----------------------+-----------+--------+------------------
↪+---------------------+----------------+--------+-------+-----------------+-----
↪---
[...]
    586 |      190 | Filter               | 494621593 |      8 |          61827699␣
↪| 2020-09-07 13:20:45 |            189 |         |       |                 |   ␣
↪0.39
    586 |      191 | GpuTransform         | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            190 |         |       |                 |   ␣
↪0.03
    586 |      192 | GpuDecompress        | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            191 |         |       |                 |   ␣
↪0.26
    586 |      193 | GpuTransform         | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            192 |         |       |                 |   ␣
↪0.01
    586 |      194 | CpuToGpu             | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            193 |         |       |                 |   ␣
↪1.86
    586 |      195 | ReorderInput         | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            194 |         |       |                 |    ␣
↪  0
    586 |      196 | Rechunk              | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            195 |         |       |                 |    ␣
↪  0
    586 |      197 | CpuDecompress        | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            196 |         |       |                 |    ␣
↪  0
    586 |      198 | ReadTable            | 494927872 |      8 |          61865984␣
↪| 2020-09-07 13:20:44 |            197 | 6595MB  |       | public.lineitem |   ␣
↪0.09
[...]
```

In this example, the filter processed 494,621,593 rows, while the output of `ReadTable` on `public.lineitem` was 494,927,872 rows. This means that it has filtered out all but 0.01% ($1 - \dfrac{494621593}{494927872} = 0.01\%$) of the data that was read.

The metadata skipping has performed very well, and has pre-filtered the data for us by pruning unnecessary chunks.

#### 8.7.4.3.2 Common Solutions for Improving Filtering

- Use clustering keys and naturally ordered data in your filters.
- Avoid full table scans when possible

### 8.7.4.4 4. Joins with `text` Keys

Joins on long text keys do not perform as well as numeric data types or very short text keys.

#### 8.7.4.4.1 Identifying the Situation

When a join is inefficient, you may note that a query spends a lot of time on the `Join` node. For example, consider these two table structures:

1. Run a query.

   In this example, we will join `t_a.fk` with `t_b.id`, both of which are `TEXT(50)`.

   ```
   SELECT AVG(t_b.j :: BIGINT),
          t_a.country_code
   FROM t_a
     JOIN t_b ON (t_a.fk = t_b.id)
   GROUP BY t_a.country_code
   ```

2. Observe the execution information by using the foreign table, or use `show_node_info`

   The execution below has been shortened, but note the highlighted rows for `Join`. The `Join` node is by far the most time-consuming part of this statement - clocking in at 69.7 seconds joining 1.5 billion records.

   ```
   t=> SELECT show_node_info(5);
   stmt_id | node_id | node_type            | rows       | chunks | avg_rows_in_chunk | time                | parent_node_id | read   | write  | comment     | timeSum
   --------+---------+----------------------+------------+--------+-------------------+---------------------+----------------+--------+--------+-------------+--------
   [...]
         5 |      19 | GpuTransform         | 1497366528 |    204 |           7340032 | 2020-09-08 18:29:03 |             18 |        |        |             |    1.46
         5 |      20 | ReorderInput         | 1497366528 |    204 |           7340032 | 2020-09-08 18:29:03 |             19 |        |        |             |       0
         5 |      21 | ReorderInput         | 1497366528 |    204 |           7340032 | 2020-09-08 18:29:03 |             20 |        |        |             |       0
         5 |      22 | Join                 | 1497366528 |    204 |           7340032 | 2020-09-08 18:29:03 |             21 |        |        | inner       |    69.7
         5 |      24 | AddSortedMinMaxMet.. |    6291456 |      1 |           6291456 | 2020-09-08 18:26:05 |             22 |        |        |             |       0
         5 |      25 | Sort                 |    6291456 |      1 |           6291456 | 2020-09-08 18:26:05 |             24 |        |        |             |    2.06
   [...]
   ```

(continues on next page)

(continued from previous page)

```
      5 |       31 | ReadTable            |    6291456 |      1 |            ␣
↪6291456 | 2020-09-08 18:26:03 |             30 | 235MB |        | public.t_b |  ␣
↪ 0.02
[...]
      5 |       41 | CpuDecompress        |   10000000 |      2 |            ␣
↪5000000 | 2020-09-08 18:26:09 |             40 |       |        |            |  ␣
↪    0
      5 |       42 | ReadTable            |   10000000 |      2 |            ␣
↪5000000 | 2020-09-08 18:26:09 |             41 | 14MB  |        | public.t_a |  ␣
↪    0
```

#### 8.7.4.4.2 Improving Query Performance

- In general, try to avoid TEXT as a join key. As a rule of thumb, BIGINT works best as a join key.
- Convert text values on-the-fly before running the query. For example, the crc64 function takes a text input and returns a BIGINT hash.

  For example:

```
SELECT AVG(t_b.j :: BIGINT),
      t_a.country_code
FROM t_a
JOIN t_b ON (crc64_join(t_a.fk) = crc64_join(t_b.id))
GROUP BY t_a.country_code
```

The execution below has been shortened, but note the highlighted rows for Join. The Join node went from taking nearly 70 seconds, to just 6.67 seconds for joining 1.5 billion records.

```
t=> SELECT show_node_info(6);
   stmt_id | node_id | node_type              | rows       | chunks | avg_rows_in_
↪chunk | time                | parent_node_id | read   | write | comment     |␣
↪timeSum
   --------+---------+----------------------+------------+--------+--------------
↪----+---------------------+----------------+-------+-------+------------+-------
↪-
   [...]
         6 |       19 | GpuTransform         | 1497366528 |     85 |            ␣
↪17825792 | 2020-09-08 18:57:04 |             18 |       |        |            |  ␣
↪  1.48
         6 |       20 | ReorderInput         | 1497366528 |     85 |            ␣
↪17825792 | 2020-09-08 18:57:04 |             19 |       |        |            |  ␣
↪    0
         6 |       21 | ReorderInput         | 1497366528 |     85 |            ␣
↪17825792 | 2020-09-08 18:57:04 |             20 |       |        |            |  ␣
↪    0
         6 |       22 | Join                 | 1497366528 |     85 |            ␣
↪17825792 | 2020-09-08 18:57:04 |             21 |       |        | inner      |  ␣
↪  6.67
         6 |       24 | AddSortedMinMaxMet.. |    6291456 |      1 |            ␣
↪6291456 | 2020-09-08 18:55:12 |             22 |       |        |            |  ␣
↪    0
   [...]
         6 |       32 | ReadTable            |    6291456 |      1 |            ␣
↪6291456 | 2020-09-08 18:55:12 |             31 | 235MB |        | public.t_b |  ␣
↪ 0.02
```

(continues on next page)

(continued from previous page)

```
  [...]
        6 |       43 | CpuDecompress        |   10000000 |      2 |        5000000 | 2020-09-08 18:55:13 |             42 |      |       |            |      0
        6 |       44 | ReadTable            |   10000000 |      2 |        5000000 | 2020-09-08 18:55:13 |             43 | 14MB |       | public.t_a |      0
```

- You can map some text values to numeric types by using a dimension table. Then, reconcile the values when you need them by joining the dimension table.

#### 8.7.4.5 5. Sorting on big TEXT fields

In general, SQream DB automatically inserts a `Sort` node which arranges the data prior to reductions and aggregations. When running a `GROUP BY` on large `TEXT` fields, you may see nodes for `Sort` and `Reduce` taking a long time.

##### 8.7.4.5.1 Identifying the Situation

When running a statement, inspect it with show_node_info. If you see `Sort` and `Reduce` among your top five longest running nodes, there is a potential issue. For example: #.

Run a query to test it out.

Our `t_inefficient` table contains 60,000,000 rows, and the structure is simple, but with an oversized `country_code` column:

```
CREATE TABLE t_inefficient (
   i INT NOT NULL,
   amt DOUBLE NOT NULL,
   ts DATETIME NOT NULL,
   country_code TEXT(100) NOT NULL,
   flag TEXT(10) NOT NULL,
   string_fk TEXT(50) NOT NULL
);
```

We will run a query, and inspect it's execution details:

```
t=> SELECT country_code,
.          SUM(amt)
.   FROM t_inefficient
.   GROUP BY country_code;
executed
time: 47.55s

country_code | sum
-------------+-----------
VUT          | 1195416012
GIB          | 1195710372
TUR          | 1195946178
[...]
```

```
t=> select show_node_info(30);
stmt_id | node_id | node_type          | rows     | chunks | avg_rows_in_chunk | time                | parent_node_id | read  | write | comment         | timeSum
```

(continues on next page)

(continued from previous page)

```
--------+---------+-------------------+----------+--------+-----------------
↪--+--------------------+----------------+-------+-------+-----------------
↪-----+--------
     30 |         1 | PushToNetworkQueue |       249 |       1 |                ␣
↪249 | 2020-09-10 16:17:10 |               -1 |         |        |             ␣
↪       |     0.25
     30 |         2 | Rechunk            |       249 |       1 |                ␣
↪249 | 2020-09-10 16:17:10 |                1 |         |        |             ␣
↪       |         0
     30 |         3 | ReduceMerge        |       249 |       1 |                ␣
↪249 | 2020-09-10 16:17:10 |                2 |         |        |             ␣
↪       |     0.01
     30 |         4 | GpuToCpu           |      1508 |      15 |                ␣
↪100 | 2020-09-10 16:17:10 |                3 |         |        |             ␣
↪       |         0
     30 |         5 | Reduce             |      1508 |      15 |                ␣
↪100 | 2020-09-10 16:17:10 |                4 |         |        |             ␣
↪       |     7.23
     30 |         6 | Sort               | 60000000 |      15 |            ␣
↪4000000 | 2020-09-10 16:17:10 |                5 |         |        |         ␣
↪           |     36.8
     30 |         7 | GpuTransform       | 60000000 |      15 |            ␣
↪4000000 | 2020-09-10 16:17:10 |                6 |         |        |         ␣
↪           |     0.08
     30 |         8 | GpuDecompress      | 60000000 |      15 |            ␣
↪4000000 | 2020-09-10 16:17:10 |                7 |         |        |         ␣
↪           |     2.01
     30 |         9 | CpuToGpu           | 60000000 |      15 |            ␣
↪4000000 | 2020-09-10 16:17:10 |                8 |         |        |         ␣
↪           |     0.16
     30 |        10 | Rechunk            | 60000000 |      15 |            ␣
↪4000000 | 2020-09-10 16:17:10 |                9 |         |        |         ␣
↪           |         0
     30 |        11 | CpuDecompress      | 60000000 |      15 |            ␣
↪4000000 | 2020-09-10 16:17:10 |               10 |         |        |         ␣
↪           |         0
     30 |        12 | ReadTable          | 60000000 |      15 |            ␣
↪4000000 | 2020-09-10 16:17:10 |               11 | 520MB |        | public.t_
↪inefficient |     0.05
```

1. We can look to see if there's any shrinking we can do on the `GROUP BY` key

```
t=> SELECT MAX(LEN(country_code)) FROM t_inefficient;
max
---
3
```

With a maximum string length of just 3 characters, our `TEXT(100)` is way oversized.

2. We can recreate the table with a more restrictive `TEXT(3)`, and can examine the difference in performance:

This time, the entire query took just 4.75 seconds, or just about 91% faster.

#### 8.7.4.5.2 Improving Sort Performance on Text Keys

When using TEXT, ensure that the maximum length defined in the table structure is as small as necessary. For example, if you're storing phone numbers, don't define the field as `TEXT(255)`, as that affects sort performance.

You can run a query to get the maximum column length (e.g. `MAX(LEN(a_column))`), and potentially modify the table structure.

### 8.7.4.6 6. High Selectivity Data

Selectivity is the ratio of cardinality to the number of records of a chunk. We define selectivity as $\frac{\text{Distinct values}}{\text{Total number of records in a chunk}}$ SQream DB has a hint called `HIGH_SELECTIVITY`, which is a function you can wrap a condition in. The hint signals to SQream DB that the result of the condition will be very sparse, and that it should attempt to rechunk the results into fewer, fuller chunks. .. note:

```
SQream DB doesn't do this automatically because it adds a significant overhead on␣
↪naturally ordered and
well-clustered data, which is the more common scenario.
```

#### 8.7.4.6.1 Identifying the Situation

This is easily identifiable - when the amount of average of rows in a chunk is small, following a `Filter` operation. Consider this execution plan:

```
t=> select show_node_info(30);
stmt_id | node_id | node_type          | rows       | chunks | avg_rows_in_chunk | time␣
↪                | parent_node_id | read   | write  | comment      | timeSum
--------+---------+--------------------+------------+--------+-------------------+------
↪---------------+----------------+--------+--------+-------------+--------
[...]
     30 |      38 | Filter             |      18160 |     74 |               245 | 2020-
↪09-10 12:17:09 |             37 |        |        |             |   0.012
[...]
     30 |      44 | ReadTable          |   77000000 |     74 |           1040540 | 2020-
↪09-10 12:17:09 |             43 | 277MB  |        | public.dim  |   0.058
```

The table was read entirely - 77 million rows into 74 chunks. The filter node reduced the output to just 18,160 relevant rows, but they're distributed across the original 74 chunks. All of these rows could fit in one single chunk, instead of spanning 74 rather sparse chunks.

#### 8.7.4.6.2 Improving Performance with High Selectivity Hints

- Use when there's a `WHERE` condition on an unclustered column, and when you expect the filter to cut out more than 60% of the result set.
- Use when the data is uniformly distributed or random

#### 8.7.4.7 7. Performance of unsorted data in joins

When data is not well-clustered or naturally ordered, a join operation can take a long time.

##### 8.7.4.7.1 Identifying the Situation

When running a statement, inspect it with show_node_info. If you see `Join` and `DeferredGather` among your top five longest running nodes, there is a potential issue. In this case, we're also interested in the number of chunks produced by these nodes.

Consider this execution plan:

```
t=> select show_node_info(30);
stmt_id | node_id | node_type          | rows      | chunks | avg_rows_in_chunk | time␣
↪              | parent_node_id | read   | write | comment    | timeSum
--------+---------+-------------------+-----------+--------+-------------------+------
↪---------------+----------------+-------+-------+------------+--------
[...]
     30 |      13 | ReorderInput       | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             12 |       |       |            |   4.681
     30 |      14 | DeferredGather     | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             13 |       |       |            |  29.901
     30 |      15 | ReorderInput       | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             14 |       |       |            |   3.053
     30 |      16 | GpuToCpu           | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             15 |       |       |            |   5.798
     30 |      17 | ReorderInput       | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             16 |       |       |            |   2.899
     30 |      18 | ReorderInput       | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             17 |       |       |            |   3.695
     30 |      19 | Join               | 181582598 |  70596 |              2572 | 2020-
↪09-10 12:17:10 |             18 |       |       | inner      |  22.745
[...]
     30 |      38 | Filter             |     18160 |     74 |               245 | 2020-
↪09-10 12:17:09 |             37 |       |       |            |   0.012
[...]
     30 |      44 | ReadTable          |  77000000 |     74 |           1040540 | 2020-
↪09-10 12:17:09 |             43 | 277MB |       | public.dim |   0.058
```

- `Join` is the node that matches rows from both table relations.
- `DeferredGather` gathers the required column chunks to decompress

Pay special attention to the volume of data removed by the `Filter` node. The table was read entirely - 77 million rows into 74 chunks. The filter node reduced the output to just 18,160 relevant rows, but they're distributed across the original 74 chunks. All of these rows could fit in one single chunk, instead of spanning 74 rather sparse chunks.

#### 8.7.4.7.2 Improving Join Performance when Data is Sparse

You can tell SQream DB to reduce the amount of chunks involved, if you know that the filter is going to be quite agressive by using the *HIGH_SELECTIVITY* hint described *above*. This forces the compiler to rechunk the data into fewer chunks. To tell SQream DB to rechunk the data, wrap a condition (or several) in the `HIGH_SELECTIVITY` hint:

```
-- Without the hint
SELECT *
FROM cdrs
WHERE
      RequestReceiveTime BETWEEN '2018-01-01 00:00:00.000' AND '2018-08-31 23:59:59.
↪999'
      AND EnterpriseID=1150
      AND MSISDN='9724871140341';

-- With the hint
SELECT *
FROM cdrs
WHERE
      HIGH_SELECTIVITY(RequestReceiveTime BETWEEN '2018-01-01 00:00:00.000' AND '2018-
↪08-31 23:59:59.999')
      AND EnterpriseID=1150
      AND MSISDN='9724871140341';
```

### 8.7.4.8 8. Manual Join Reordering

When joining multiple tables, you may wish to change the join order to join the smallest tables first.

#### 8.7.4.8.1 Identifying the situation

When joining more than two tables, the `Join` nodes will be the most time-consuming nodes.

#### 8.7.4.8.2 Changing the Join Order

Always prefer to join the smallest tables first. .. note:

```
We consider small tables to be tables that only retain a small amount of rows after␣
↪conditions
are applied. This bears no direct relation to the amount of total rows in the table.
```

Changing the join order can reduce the query runtime significantly. In the examples below, we reduce the time from 27.3 seconds to just 6.4 seconds.

Listing 1: Original query

```
-- This variant runs in 27.3 seconds
SELECT SUM(l_extendedprice / 100.0*(1 - l_discount / 100.0)) AS revenue,
       c_nationkey
FROM lineitem --6B Rows, ~183GB
  JOIN orders --1.5B Rows, ~55GB
  ON   l_orderkey = o_orderkey
  JOIN customer --150M Rows, ~12GB
  ON   c_custkey = o_custkey
```

(continues on next page)

(continued from previous page)

```
WHERE c_nationkey = 1
      AND   o_orderdate >= DATE '1993-01-01'
      AND   o_orderdate < '1994-01-01'
      AND   l_shipdate >= '1993-01-01'
      AND   l_shipdate <= dateadd(DAY,122,'1994-01-01')
GROUP BY c_nationkey
```

Listing 2: Modified query with improved join order

```
-- This variant runs in 6.4 seconds
SELECT SUM(l_extendedprice / 100.0*(1 - l_discount / 100.0)) AS revenue,
       c_nationkey
FROM orders --1.5B Rows, ~55GB
  JOIN customer --150M Rows, ~12GB
  ON   c_custkey = o_custkey
  JOIN lineitem --6B Rows, ~183GB
  ON   l_orderkey = o_orderkey

WHERE c_nationkey = 1
      AND   o_orderdate >= DATE '1993-01-01'
      AND   o_orderdate < '1994-01-01'
      AND   l_shipdate >= '1993-01-01'
      AND   l_shipdate <= dateadd(DAY,122,'1994-01-01')
GROUP BY c_nationkey
```

### 8.7.5 Further Reading

See our *Optimization and Best Practices* guide for more information about query optimization and data loading considerations.

## 8.8 Security

SQream DB has some security features that you should be aware of to increase the security of your data.

**In this topic:**

### 8.8.1 Overview

An **initial, unsecured** installation of SQream DB can carry some risks:

- Your data open to any client that can access an open node through an IP and port combination.
- The initial administrator username and password, when unchanged, can let anyone log in.
- Network connections to SQream DB aren't encrypted.

To avoid these security risks, SQream DB provides authentication, authorizaiton, logging, and network encryption.

Read through the best practices guide to understand more.

### 8.8.2 Security best practices for SQream DB

#### 8.8.2.1 Secure OS access

SQream DB often runs as a dedicated user on the host OS. This user is the file system owner of SQream DB data files.

Any user who logs in to the OS with this user can read or delete data from outside of SQream DB.

This user can also read any logs which may contain user login attempts.

Therefore, it is very important to secure the host OS and prevent unauthorized access.

System administrators should only log in to the host OS to perform maintenance tasks like upgrades. A database user should not log in using the same username in production environments.

#### 8.8.2.2 Change the default `SUPERUSER`

To bootstrap SQream DB, a new install will always have one `SUPERUSER` role, typically named `sqream`. After creating a second `SUPERUSER` role, remove or change the default credentials to the default `sqream` user.

No database user should ever use the default `SUPERUSER` role in a production environment.

#### 8.8.2.3 Create distinct user roles

Each user that signs in to a SQream DB cluster should have a distinct user role for several reasons:

- For logging and auditing purposes. Each user that logs in to SQream DB can be identified.
- For limiting permissions. Use groups and permissions to manage access. See our *Access Control* guide for more information.

#### 8.8.2.4 Limit `SUPERUSER` access

Limit users who have the `SUPERUSER` role.

A superuser role bypasses all permissions checks. Only system administrators should have `SUPERUSER` roles. See our *Access Control* guide for more information.

#### 8.8.2.5 Password strength guidelines

System administrators should verify the passwords used are strong ones.

SQream DB stores passwords as salted SHA1 hashes in the system catalog so they are obscured and can't be recovered. However, passwords may appear in server logs. Prevent access to server logs by securing OS access as described above.

Follow these recommendations to strengthen passwords:

- Pick a password that's easy to remember
- At least 8 characters
- Mix upper and lower case letters
- Mix letters and numbers
- Include non-alphanumeric characters (except `"` and `'`)

#### 8.8.2.6 Use TLS/SSL when possible

SQream DB's protocol implements client/server TLS security (even though it is called SSL).

All SQream DB connectors and drivers support transport encryption. Ensure that each connection uses SSL and the correct access port for the SQream DB cluster:

- The load balancer (`server_picker`) is often started with the secure port at an offset of 1 from the original port (e.g. port 3108 for the unsecured connection and port 3109 for the secured connection).
- A SQream DB worker is often started with the secure port enabled at an offset of 100 from the original port (e.g. port 5000 for the unsecured connection and port 5100 for the secured connection).

Refer to each *client driver* for instructions on enabling TLS/SSL.

## 8.9 Seeing System Objects as DDL

### 8.9.1 Dump specific objects

#### 8.9.1.1 Tables

See get_ddl for more information.

#### Examples

##### 8.9.1.1.1 Getting the DDL for a table

```
farm=> SELECT GET_DDL('cool_animals');
create table "public"."cool_animals" (
  "id" int not null,
  "name" text(30) not null,
  "weight" double null,
  "is_agressive" bool default false not null )
  ;
```

##### 8.9.1.1.2 Exporting table DDL to a file

```
COPY (SELECT GET_DDL('cool_animals')) TO '/home/rhendricks/animals.ddl';
```

#### 8.9.1.2 Views

See get_view_ddl for more information.

### Examples

##### 8.9.1.2.1 Listing all views

```
farm=> SELECT view_name FROM sqream_catalog.views;
view_name
----------------------
angry_animals
only_agressive_animals
```

##### 8.9.1.2.2 Getting the DDL for a view

```
farm=> SELECT GET_VIEW_DDL('angry_animals');
create view "public".angry_animals as
   select
      "cool_animals"."id" as "id",
      "cool_animals"."name" as "name",
      "cool_animals"."weight" as "weight",
      "cool_animals"."is_agressive" as "is_agressive"
    from
      "public".cool_animals as cool_animals
    where
      "cool_animals"."is_agressive" = false;
```

##### 8.9.1.2.3 Exporting view DDL to a file

```
COPY (SELECT GET_VIEW_DDL('angry_animals')) TO '/home/rhendricks/angry_animals.sql';
```

#### 8.9.1.3 User defined functions

See get_function_ddl for more information.

### Examples

#### 8.9.1.3.1 Listing all UDFs

```
master=> SELECT * FROM sqream_catalog.user_defined_functions;
database_name | function_id | function_name
--------------+-------------+--------------
master        |           1 | my_distance
```

#### 8.9.1.3.2 Getting the DDL for a function

```
master=> SELECT GET_FUNCTION_DDL('my_distance');
create function "my_distance" (x1 float,
                            y1 float,
                            x2 float,
                            y2 float) returns float as
   $$
   import  math
   if  y1  <  x1:
       return  0.0
   else:
       return  math.sqrt((y2  -  y1)  **  2  +  (x2  -  x1)  **  2)
   $$
   language python volatile;
```

### 8.9.1.4 Exporting function DDL to a file

```
COPY (SELECT GET_FUNCTION_DDL('my_distance')) TO '/home/rhendricks/my_distance.sql';
```

### 8.9.1.5 Saved queries

See list_saved_queries, show_saved_query for more information.

## 8.9.2 Dump entire database DDLs

Dumping the database DDL includes tables and views, but not UDFs and saved queries.

See dump_database_ddl for more information.

### Examples

#### 8.9.2.1 Exporting database DDL to a client

```
farm=> SELECT DUMP_DATABASE_DDL();
create table "public"."cool_animals" (
  "id" int not null,
  "name" text(30) not null,
  "weight" double null,
```

(continues on next page)

(continued from previous page)

```
  "is_agressive" bool default false not null
)
;

create view "public".angry_animals as
  select
      "cool_animals"."id" as "id",
      "cool_animals"."name" as "name",
      "cool_animals"."weight" as "weight",
      "cool_animals"."is_agressive" as "is_agressive"
    from
      "public".cool_animals as cool_animals
    where
      "cool_animals"."is_agressive" = false;
```

#### 8.9.2.2 Exporting database DDL to a file

```
COPY (SELECT DUMP_DATABASE_DDL()) TO '/home/rhendricks/database.ddl';
```

**Note:** To export data in tables, see copy_to.

## 8.10 Optimization and Best Practices

This topic explains some best practices of working with SQream DB.

See also our *Monitoring Query Performance* guide for more information.

**In this topic:**

## 8.10.1 Table design

This section describes best practices and guidelines for designing tables.

### 8.10.1.1 Use date and datetime types for columns

When creating tables with dates or timestamps, using the purpose-built `DATE` and `DATETIME` types over integer types or `TEXT` will bring performance and storage footprint improvements, and in many cases huge performance improvements (as well as data integrity benefits). SQream DB stores dates and datetimes very efficiently and can strongly optimize queries using these specific types.

### 8.10.1.2 Don't flatten or denormalize data

SQream DB executes JOIN operations very effectively. It is almost always better to JOIN tables at query-time rather than flatten/denormalize your tables.

This will also reduce storage size and reduce row-lengths.

We highly suggest using `INT` or `BIGINT` as join keys, rather than a text/string type.

### 8.10.1.3 Convert foreign tables to native tables

SQream DB's native storage is heavily optimized for analytic workloads. It is always faster for querying than other formats, even columnar ones such as Parquet. It also enables the use of additional metadata to help speed up queries, in some cases by many orders of magnitude.

You can improve the performance of all operations by converting foreign tables into native tables by using the create_table_as syntax.

For example,

```
CREATE TABLE native_table AS SELECT * FROM external_table
```

The one situation when this wouldn't be as useful is when data will be only queried once.

#### 8.10.1.4 Use information about the column data to your advantage

Knowing the data types and their ranges can help design a better table.

##### 8.10.1.4.1 Set `NULL` or `NOT NULL` when relevant

For example, if a value can't be missing (or `NULL`), specify a `NOT NULL` constraint on the columns.

Not only does specifying `NOT NULL` save on data storage, it lets the query compiler know that a column cannot have a `NULL` value, which can improve query performance.

### 8.10.2 Sorting

Data sorting is an important factor in minimizing storage size and improving query performance.

- Minimizing storage saves on physical resources and increases performance by reducing overall disk I/O. Prioritize the sorting of low-cardinality columns. This reduces the number of chunks and extents that SQream DB reads during query execution.
- Where possible, sort columns with the lowest cardinality first. Avoid sorting `TEXT` columns with lengths exceeding 50 characters.
- For longer-running queries that run on a regular basis, performance can be improved by sorting data based on the `WHERE` and `GROUP BY` parameters. Data can be sorted during insert by using external_tables or by using create_table_as.

### 8.10.3 Query best practices

This section describes best practices for writing SQL queries.

#### 8.10.3.1 Reduce data sets before joining tables

Reducing the input to a `JOIN` clause can increase performance. Some queries benefit from retreiving a reduced dataset as a subquery prior to a join.

For example,

```
SELECT store_name, SUM(amount)
FROM store_dim AS dim INNER JOIN store_fact AS fact ON dim.store_id=fact.store_id
WHERE p_date BETWEEN '2018-07-01' AND '2018-07-31'
GROUP BY 1;
```

Can be rewritten as

```
SELECT store_name, sum_amount
FROM store_dim AS dim INNER JOIN
   (SELECT SUM(amount) AS sum_amount, store_id
   FROM store_fact
   WHERE p_date BETWEEN '2018-07-01' AND '2018-07-31'
   group by 2) AS fact
ON dim.store_id=fact.store_id;
```

#### 8.10.3.2 Prefer the ANSI JOIN

SQream DB prefers the ANSI JOIN syntax. In some cases, the ANSI JOIN performs better than the non-ANSI variety.

For example, this ANSI JOIN example will perform better:

Listing 3: ANSI JOIN will perform better

```
SELECT p.name, s.name, c.name
FROM  "Products" AS p
JOIN  "Sales" AS s
  ON  p.product_id = s.sale_id
JOIN  "Customers" as c
  ON  s.c_id = c.id AND c.id = 20301125;
```

This non-ANSI JOIN is supported, but not recommended:

Listing 4: Non-ANSI JOIN may not perform well

```
SELECT p.name, s.name, c.name
FROM "Products" AS p, "Sales" AS s, "Customers" as c
WHERE p.product_id = s.sale_id
  AND s.c_id = c.id
  AND c.id = 20301125;
```

#### 8.10.3.3 Use the high selectivity hint

Selectivity is the ratio of cardinality to the number of records of a chunk. We define selectivity as $\frac{\text{Distinct values}}{\text{Total number of records in a chunk}}$

SQream DB has a hint function called `HIGH_SELECTIVITY`, which is a function you can wrap a condition in.

The hint signals to SQream DB that the result of the condition will be very sparse, and that it should attempt to rechunk the results into fewer, fuller chunks.

Use the high selectivity hint when you expect a predicate to filter out most values. For example, when the data is dispersed over lots of chunks (meaning that the data is not well-clustered).

For example,

```
SELECT store_name, SUM(amount) FROM store_dim
WHERE HIGH_SELECTIVITY(p_date = '2018-07-01')
GROUP BY 1;
```

This hint tells the query compiler that the `WHERE` condition is expected to filter out more than 60% of values. It never affects the query results, but when used correctly can improve query performance.

**Tip:** The `HIGH_SELECTIVITY()` hint function can only be used as part of the `WHERE` clause. It can't be used in equijoin conditions, cases, or in the select list.

Read more about identifying the scenarios for the high selectivity hint in our *Monitoring query performance guide*.

#### 8.10.3.4 Cast smaller types to avoid overflow in aggregates

When using an INT or smaller type, the SUM and COUNT operations return a value of the same type. To avoid overflow on large results, cast the column up to a larger type.

For example

```
SELECT store_name, SUM(amount :: BIGINT) FROM store_dim
GROUP BY 1;
```

#### 8.10.3.5 Prefer COUNT(*) and COUNT on non-nullable columns

SQream DB optimizes COUNT(*) queries very strongly. This also applies to COUNT(column_name) on non-nullable columns. Using COUNT(column_name) on a nullable column will operate quickly, but much slower than the previous variations.

#### 8.10.3.6 Return only required columns

Returning only the columns you need to client programs can improve overall query performance. This also reduces the overall result set, which can improve performance in third-party tools.

SQream is able to optimize out unneeded columns very strongly due to its columnar storage.

#### 8.10.3.7 Use saved queries to reduce recurring compilation time

saved_queries are compiled when they are created. The query plan is saved in SQream DB's metadata for later re-use.

Because the query plan is saved, they can be used to reduce compilation overhead, especially with very complex queries, such as queries with lots of values in an IN predicate.

When executed, the saved query plan is recalled and executed on the up-to-date data stored on disk.

See how to use saved queries in the saved queries guide.

#### 8.10.3.8 Pre-filter to reduce JOIN complexity

Filter and reduce table sizes prior to joining on them

```
SELECT store_name,
       SUM(amount)
FROM dimention dim
  JOIN fact ON dim.store_id = fact.store_id
WHERE p_date BETWEEN '2019-07-01' AND '2019-07-31'
GROUP BY store_name;
```

Can be rewritten as:

```
SELECT store_name,
       sum_amount
FROM dimention AS dim
  INNER JOIN (SELECT SUM(amount) AS sum_amount,
                     store_id
              FROM fact
              WHERE p_date BETWEEN '2019-07-01' AND '2019-07-31'
              GROUP BY store_id) AS fact ON dim.store_id = fact.store_id;
```

## 8.10.4 Data loading considerations

### 8.10.4.1 Allow and use natural sorting on data

Very often, tabular data is already naturally ordered along a dimension such as a timestamp or area.

This natural order is a major factor for query performance later on, as data that is naturally sorted can be more easily compressed and analyzed with SQream DB's metadata collection.

For example, when data is sorted by timestamp, filtering on this timestamp is more effective than filtering on an unordered column.

Natural ordering can also be used for effective delete operations.

## 8.10.5 Further reading and monitoring query performance

Read our *Monitoring Query Performance* guide to learn how to use the built in monitoring utilities. The guide also gives concerete examples for improving query performance.

CHAPTER

NINE

# SQREAM ACCELERATION STUDIO 5.4.7

The SQream Acceleration Studio is a web-based client for use with SQream. Studio provides users with all functionality available from the command line in an intuitive and easy-to-use format. This includes running statements, managing roles and permissions, and managing SQream clusters.

This section describes how to use the SQream Accleration Studio version 5.4.7:

## 9.1 Getting Started with SQream Acceleration Studio 5.4.7

### 9.1.1 Setting Up and Starting Studio

Studio is included with all dockerized installations of SQream DB. When starting Studio, it listens on the local machine on port 8080.

### 9.1.2 Logging In to Studio

**To log in to SQream Studio:**

1. Open a browser to the host on **port 8080**.

   For example, if your machine IP address is `192.168.0.100`, insert the IP address into the browser as shown below:

   ```
   $ http://192.168.0.100:8080
   ```

2. Fill in your SQream DB login credentials. These are the same credentials used for *sqream sql* or JDBC.

   When you sign in, the License Warning is displayed.

### 9.1.3 Navigating Studio's Main Features

When you log in, you are automatically taken to the **Editor** screen. The Studio's main functions are displayed in the **Navigation** pane on the left side of the screen.

From here you can navigate between the main areas of the Studio:

| Element | Description |
|---|---|
| *Dashboard* | Lets you monitor system health and manage queues and workers. |
| *Editor* | Lets you select databases, perform statement operations, and write and execute queries. |
| *Logs* | Lets you view usage logs. |
| *Roles* | Lets you create users and manage user permissions. |

By clicking the user icon, you can also use it for logging out and viewing the following:

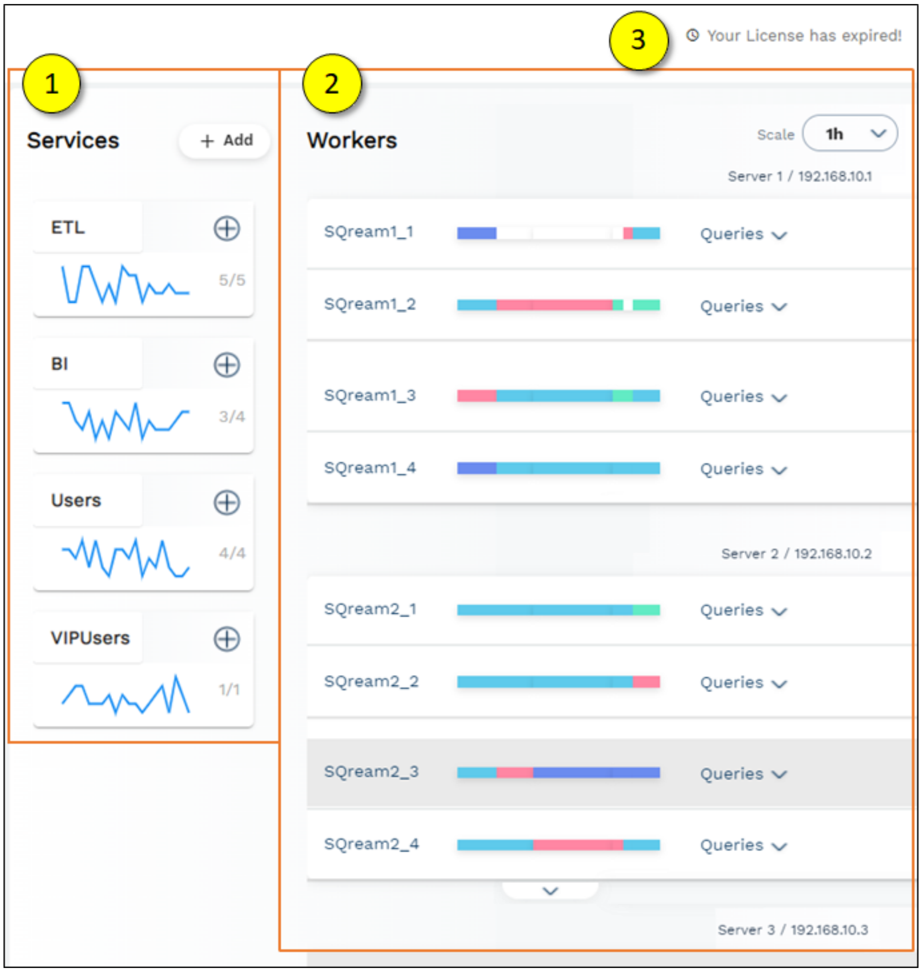
- User information
- Connection type
- SQream version
- SQream Studio version
- License expiration date
- License storage capacity
- Log out

## 9.2 Monitoring Workers and Services from the Dashboard

The **Dashboard** is used for the following:

- Monitoring system health.
- Viewing, monitoring, and adding defined service queues.
- Viewing and managing worker status and add workers.

The following is an image of the Dashboard:




You can only access the Dashboard if you signed in with a `SUPERUSER` role.

The following is a brief description of the Dashboard panels:

| No. | Element | Description |
|---|---|---|
| 1 | *Services panel* | Used for viewing and monitoring the defined service queues. |
| 2 | *Workers panel* | Monitors system health and shows each Sqreamd worker running in the cluster. |
| 3 | *License information* | Shows the remaining amount of days left on your license. |

*Back to Monitoring Workers and Services from the Dashboard*

### 9.2.1 Subscribing to Workers from the Services Panel

Services are used to categorize and associate (also known as **subscribing**) workers to particular services. The **Service** panel is used for viewing, monitoring, and adding defined service queues.

The following is a brief description of each pane:

| No. | Description |
|---|---|
| 1 | Adds a worker to the selected service. |
| 2 | Shows the service name. |
| 3 | Shows a trend graph of queued statements loaded over time. |
| 4 | Adds a service. |
| 5 | Shows the currently processed queries belonging to the service/total queries for that service in the system (including queued queries). |

#### 9.2.1.1 Adding A Service

You can add a service by clicking **+ Add** and defining the service name.

**Note:** If you do not associate a worker with the new service, it will not be created.

You can manage workers from the **Workers** panel. For more information about managing workers, see the following:

- *Managing Workers from the Workers Panel*
- Workers

*Back to Monitoring Workers and Services from the Dashboard*

### 9.2.2 Managing Workers from the Workers Panel

From the **Workers** panel you can do the following:

- *View workers*
- *Add a worker to a service*
- *View a worker's active query information*
- *View a worker's execution plan*

#### 9.2.2.1 Viewing Workers

The **Worker** panel shows each worker (`sqreamd`) running in the cluster. Each worker has a status bar that represents the status over time. The status bar is divided into 20 equal segments, showing the most dominant activity in that segment.

From the **Scale** dropdown menu you can set the time scale of the displayed information You can hover over segments in the status bar to see the date and time corresponding to each activity type:

- **Idle –** the worker is idle and available for statements.
- **Compiling –** the worker is compiling a statement and is preparing for execution.
- **Executing –** the worker is executing a statement after compilation.
- **Stopped –** the worker was stopped (either deliberately or due to an error).
- **Waiting –** the worker was waiting on an object locked by another worker.

#### 9.2.2.2 Adding A Worker to A Service

You can add a worker to a service by clicking the **add** button.

Clicking the **add** button shows the selected service's workers. You can add the selected worker to the service by clicking **Add Worker**. Adding a worker to a service does not break associations already made between that worker and other services.

#### 9.2.2.3 Viewing A Worker's Active Query Information

You can view a worker's active query information by clicking **Queries**, which displays them in the selected service.

Each statement shows the **query ID**, **status**, **service queue**, **elapsed time**, **execution time**, and **estimated completion status**. In addition, each statement can be stopped or expanded to show its execution plan and progress. For more information on viewing a statement's execution plan and progress, see *Viewing a Worker's Execution Plan* below.

#### 9.2.2.4 Viewing A Worker's Host Utilization

While viewing a worker's query information, clicking the **down arrow** expands to show the host resource utilization.

The graphs show the resource utilization trends over time, and the **CPU memory** and **utilization** and the **GPU utilization** values on the right. You can hover over the graph to see more information about the activity at any point on the graph.

Error notifications related to statements are displayed, and you can hover over them for more information about the error.

#### 9.2.2.5 Viewing a Worker's Execution Plan

Clicking the ellipsis in a service shows the following additional options:

- **Stop Query** - stops the query.
- **Show Execution Plan** - shows the execution plan as a table. The columns in the **Show Execution Plan** table can be sorted.

For more information on the current query plan, see SHOW_NODE_INFO. For more information on checking active sessions across the cluster, see SHOW_SERVER_STATUS.

#### 9.2.2.6 Managing Worker Status

In some cases you may want to stop or restart workers for maintenance purposes. Each Worker line has a ⋮ menu used for stopping, starting, or restarting workers.

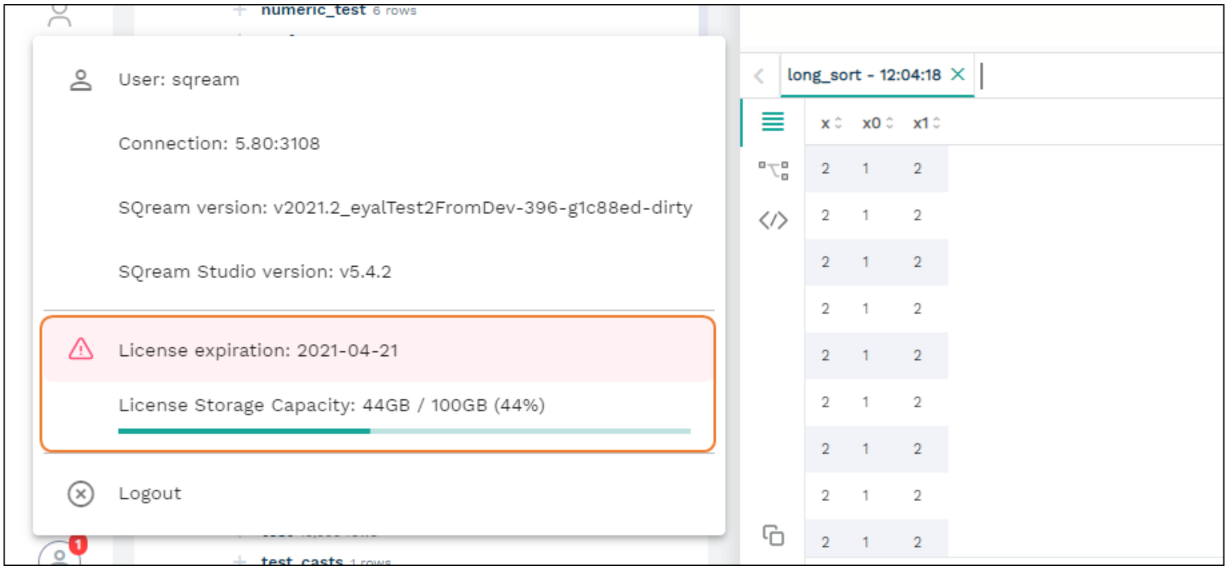
Starting or restarting workers terminates all queries related to that worker. When you stop a worker, its background turns gray.

*Back to Monitoring Workers and Services from the Dashboard*

### 9.2.3 License Information

The license information section shows the following:

- The amount of time in days remaining on the license.
- The license storage capacity.




*Back to Monitoring Workers and Services from the Dashboard*

## 9.3 Executing Statements and Running Queries from the Editor

The **Editor** is used for the following:

- Selecting an active database and executing queries.
- Performing statement-related operations and showing metadata.
- Executing pre-defined queries.
- Writing queries and statements and viewing query results.

The following is a brief description of the Editor panels:

| No. | Element | Description |
|---|---|---|
| 1 | *Toolbar* | Used to select the active database you want to work on, limit the number of rows, save query, etc. |
| 2 | *Database Tree and System Queries panel* | Shows a hierarchy tree of databases, views, tables, and columns |
| 3 | *Statement panel* | Used for writing queries and statements |
| 4 | *Results panel* | Shows query results and execution information. |

### 9.3.1 Executing Statements from the Toolbar

You can access the following from the Toolbar pane:

- **Database dropdown list** - select a database that you want to run statements on.

- **Service dropdown list** - select a service that you want to run statements on. The options in the service dropdown menu depend on the database you select from the **Database** dropdown list.

- **Execute** - lets you set which statements to execute. The **Execute** button toggles between **Execute** and **Stop**, and can be used to stop an active statement before it completes:
  - **Statements** - executes the statement at the location of the cursor.
  - **Selected** - executes only the highlighted text. This mode should be used when executing subqueries or sections of large queries (as long as they are valid SQLs).
  - **All** - executes all statements in a selected tab.
- **Format SQL** - Lets you reformat and reindent statements.

- **Download query** - Lets you download query text to your computer.

- **Open query** - Lets you upload query text from your computer.

- **Max Rows** - By default, the Editor fetches only the first 10,000 rows. You can modify this number by selecting an option from the **Max Rows** dropdown list. Note that setting a higher number may slow down your browser if the result is very large. This number is limited to 100,000 results. To see a higher number, you can save the results in a file or a table using the create_table_as command.

For more information on stopping active statements, see the STOP_STATEMENT command.

*Back to Executing Statements and Running Queries from the Editor*

### 9.3.2 Performing Statement-Related Operations from the Database Tree

From the Database Tree you can perform statement-related operations and show metadata (such as a number indicating the amount of rows in the table).

The database object functions are used to perform the following:

- The **SELECT** statement - copies the selected table's **columns** into the Statement panel as `SELECT` parameters.

- The **copy** feature - copies the selected table's **name** into the Statement panel.

- The **additional operations** - displays the following additional options:

| Function | Description |
|---|---|
| Insert statement | Generates an INSERT statement for the selected table in the editing area. |
| Delete statement | Generates a DELETE statement for the selected table in the editing area. |
| Create Table As statement | Generates a CREATE TABLE AS statement for the selected table in the editing area. |
| Rename statement | Generates an RENAME TABLE AS statement for renaming the selected table in the editing area. |
| Adding column statement | Generates an ADD COLUMN statement for adding columns to the selected table in the editing area. |
| Truncate table statement | Generates a TRUNCATE_IF_EXISTS statement for the selected table in the editing area. |
| Drop table statement | Generates a `DROP` statement for the selected object in the editing area. |
| Table DDL | Generates a DDL statement for the selected object in the editing area. To get the entire database DDL, click the  icon next to the database name in the tree root. See Seeing System Objects as DDL. |
| DDL Optimizer | The DDL Optimizer lets you analyze database tables and recommends possible optimizations. |

#### 9.3.2.1 Optimizing Database Tables Using the DDL Optimizer

The **DDL Optimizer** tab analyzes database tables and recommends possible optimizations according to SQream's best practices.

As described in the previous table, you can access the DDL Optimizer by clicking the **additional options icon** and selecting **DDL Optimizer**.

The following table describes the DDL Optimizer screen:

| Element | Description |
|---|---|
| Column area | Shows the column **names** and **column types** from the selected table. You can scroll down or to the right/left for long column lists. |
| Optimization area | Shows the number of rows to sample as the basis for running an optimization, the default setting (1,000,000) when running an optimization (this is also the overhead threshold used when analyzing `TEXT` fields), and the default percent buffer to add to `TEXT` lengths (10%). Attempts to determine field nullability. |
| Run Optimizer | Starts the optimization process. |

Clicking **Run Optimizer** adds a tab to the Statement panel showing the optimized results of the selected object.

For more information, see Optimization and Best Practices.

#### 9.3.2.2 Executing Pre-Defined Queries from the System Queries Panel

The **System Queries** panel lets you execute predefined queries and includes the following system query types:

- **Catalog queries** - Used for analyzing table compression rates, users and permissions, etc.

- **Admin queries** - Queries useful for SQream database management.

Clicking an item pastes the query into the Statement pane, and you can undo a previous operation by pressing **Ctrl + Z**.

### 9.3.3 Writing Statements and Queries from the Statement Panel

The multi-tabbed statement area is used for writing queries and statements, and is used in tandem with the toolbar. When writing and executing statements, you must first select a database from the **Database** dropdown menu in the toolbar. When you execute a statement, it passes through a series of statuses until completed. Knowing the status helps you with statement maintenance, and the statuses are shown in the **Results panel**.

The auto-complete feature assists you when writing statements by suggesting statement options.

The following table shows the statement statuses:

| Status | Description |
|---|---|
| Pending | The statement is pending. |
| In queue | The statement is waiting for execution. |
| Initializing | The statement has entered execution checks. |
| Executing | The statement is executing. |
| Statement stopped | The statement has been stopped. |

You can add and name new tabs for each statement that you need to execute, and Studio preserves your created tabs when you switch between databases. You can add new tabs by clicking , which creates a new tab to the right with a default name of SQL and an increasing number. This helps you keep track of your statements.

You can also rename the default tab name by double-clicking it and typing a new name and write multiple statements in tandem in the same tab by separating them with semicolons (`;`).If too many tabs to fit into the Statement Pane are open at the same time, the tab arrows are displayed. You can scroll through the tabs by clicking or , and close tabs by clicking . You can also close all tabs at once by clicking **Close all** located to the right of the tabs.

**Tip:** If this is your first time using SQream, see Getting Started.

*Back to Executing Statements and Running Queries from the Editor*

### 9.3.4 Viewing Statement and Query Results from the Results Panel

The results panel shows statement and query results. By default, only the first 10,000 results are returned, although you can modify this from the studio_editor_toolbar, as described above. By default, executing several statements together opens a separate results tab for each statement. Executing statements together executes them serially, and any failed statement cancels all subsequent executions.


The following is a brief description of the Results panel views highlighted in the figure above:

| Element | Description |
|---|---|
| *Results view* | Lets you view search query results. |
| *Execution Details view* | Lets you analyze your query for troubleshooting and optimization purposes. |
| *SQL view* | Lets you see the SQL view. |

*Back to Executing Statements and Running Queries from the Editor*

#### 9.3.4.1 Searching Query Results in the Results View

The **Results view** lets you view search query results.

From this view you can also do the following:

- View the amount of time (in seconds) taken for a query to finish executing.
- Switch and scroll between tabs.
- Close all tabs at once.
- Enable keeping tabs by selecting **Keep tabs**.
- Sort column results.

##### 9.3.4.1.1 Saving Results to the Clipboard

The **Save results to clipboard** function lets you save your results to the clipboard to paste into another text editor or into Excel for further analysis.

##### 9.3.4.1.2 Saving Results to a Local File

The **Save results to local file** functions lets you save your search query results to a local file. Clicking **Save results to local file** downloads the contents of the Results panel to an Excel sheet. You can then use copy and paste this content into other editors as needed.

In the Results view you can also run parallel statements, as described in **Running Parallel Statements** below.

##### 9.3.4.1.3 Running Parallel Statements

While Studio's default functionality is to open a new tab for each executed statement, Studio supports running parallel statements in one statement tab. Running parallel statements requires using macros and is useful for advanced users.

The following shows the syntax for running parallel statements:

```
$ @@ parallel
$ $$
$ select 1;
$ select 2;
$ select 3;
$ $$
```

*Back to Viewing Statement and Query Results from the Results Panel*

#### 9.3.4.2 Execution Details View

The **Execution Details View** section describes the following:

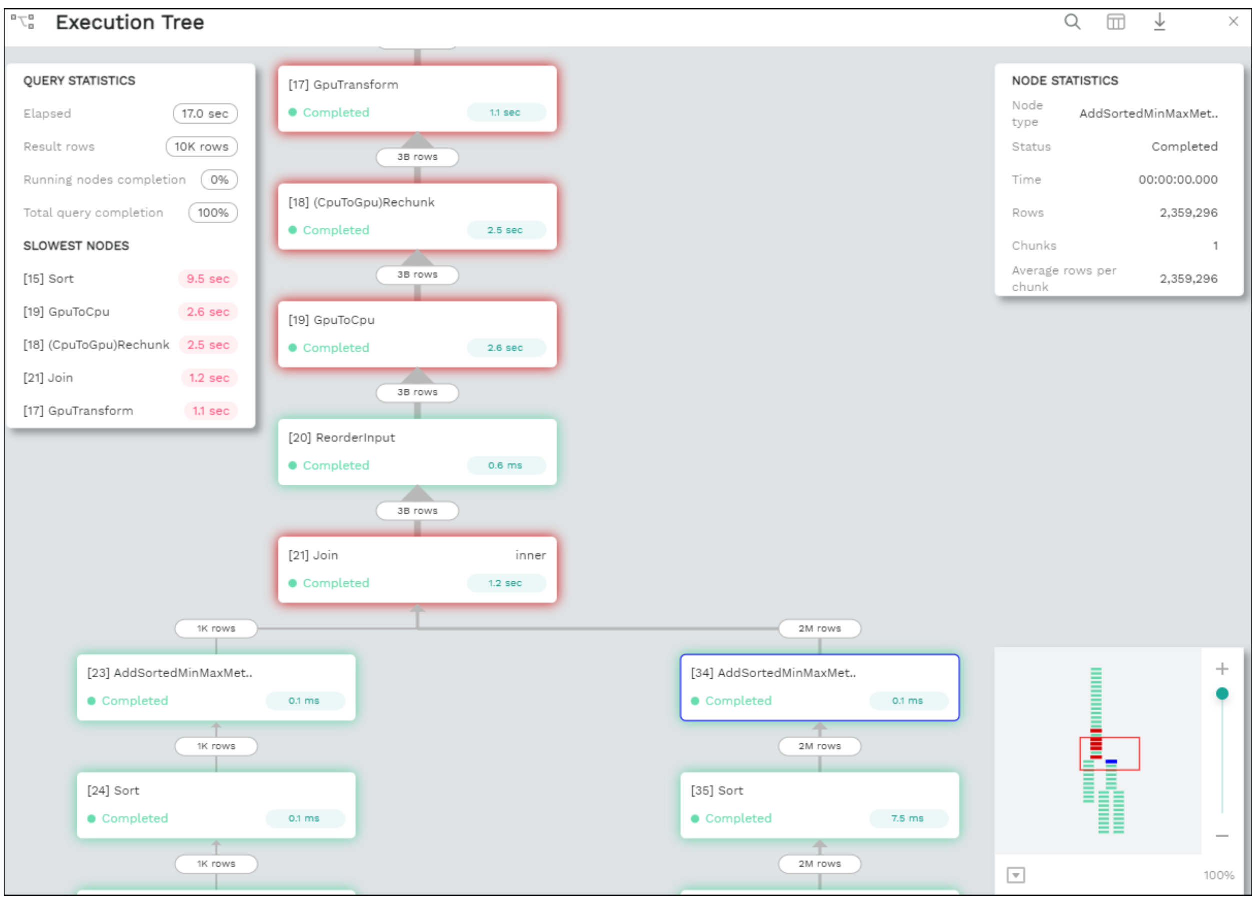


##### 9.3.4.2.1 Overview

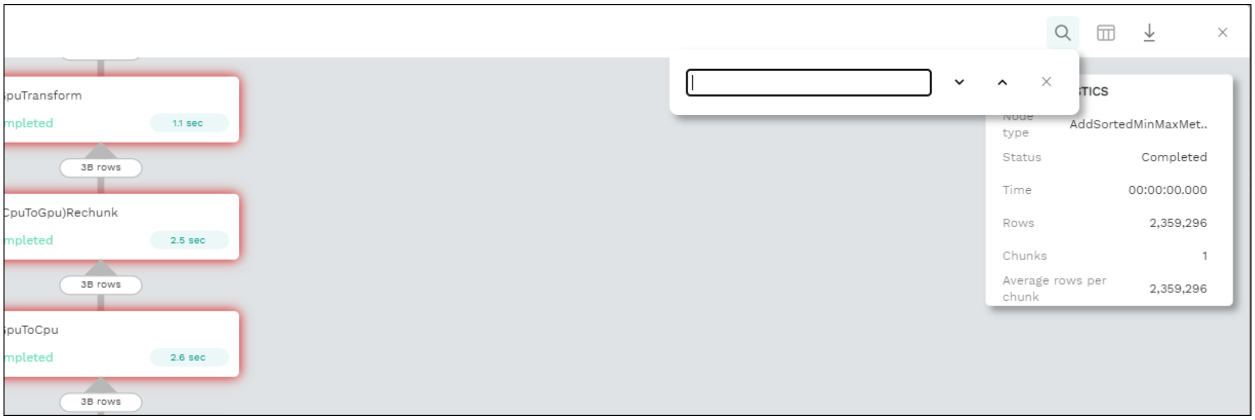
Clicking **Execution Details View** displays the **Execution Tree**, which is a chronological tree of processes that occurred to execute your queries. The purpose of the Execution Tree is to analyze all aspects of your query for troubleshooting and optimization purposes, such as resolving queries with an exceptionally long runtime.

**Note:** The **Execution Details View** button is enabled only when a query takes longer than five seconds.

From this screen you can scroll in, out, and around the execution tree with the mouse to analyze all aspects of your query. You can navigate around the execution tree by dragging or by using the mini-map in the bottom right corner.




You can also search for query data by pressing **Ctrl+F** or clicking the search icon in the search field in the top right corner and typing text.




Pressing **Enter** takes you directly to the next result matching your search criteria, and pressing **Shift + Enter** takes you directly to the previous result. You can also search next and previous results using the up and down arrows.

The nodes are color-coded based on the following:

- **Slow nodes** - red
- **In progress nodes** - yellow
- **Completed nodes** - green
- **Pending nodes** - white
- **Currently selected node** - blue
- **Search result node** - purple (in the mini-map)

The execution tree displays the same information as shown in the plain view in tree format.

The Execution Tree tracks each phase of your query in real time as a vertical tree of nodes. Each node refers to an operation that occurred on the GPU or CPU. When a phase is completed, the next branch begins to its right until the entire query is complete. Joins are displayed as two parallel branches merged together in a node called **Join**, as shown in the figure above. The nodes are connected by a line indicating the number of rows passed from one node to the next. The width of the line indicates the amount of rows on a logarithmic scale.

Each node displays a number displaying its **node ID**, its **type**, **table name** (if relevant), **status**, and **runtime**. The nodes are color-coded for easy identification. Green nodes indicate **completed nodes**, yellow indicates **nodes in progress**, and red indicates **slowest nodes**, typically joins, as shown below:

#### 9.3.4.2.2 Viewing Query Statistics

The following statistical information is displayed in the top left corner, as shown in the figure above:

- **Query Statistics**:
  - **Elapsed** - the total time taken for the query to complete.
  - **Result rows** - the amount of rows fetched.
  - **Running nodes completion**
  - **Total query completion** - the amount of the total execution tree that was executed (nodes marked green).
- **Slowest Nodes** information is displayed in the top right corner in red text. Clicking the slowest node centers automatically on that node in the execution tree.

You can also view the following **Node Statistics** in the top right corner for each individual node by clicking a node:

| Element | Description |
|---|---|
| Node type | Shows the node type. |
| Status | Shows the execution status. |
| Time | The total time taken to execute. |
| Rows | Shows the number of produced rows passed to the next node. |
| Chunks | Shows number of produced chunks. |
| Average rows per chunk | Shows the number of average rows per chunk. |
| Table (for **ReadTable** and joins only) | Shows the table name. |
| Write (for joins only) | Shows the total date size written to the disk. |
| Read (for **ReadTable** and joins only) | Shows the total data size read from the disk. |

Note that you can scroll the Node Statistics table. You can also download the execution plan table in .csv format by clicking the download arrow in the upper-right corner.

#### 9.3.4.2.3 Using the Plain View

You can use the **Plain View** instead of viewing the execution tree by clicking **Plain View** in the top right corner. The plain view displays the same information as shown in the execution tree in table format.

The plain view lets you view a query's execution plan for monitoring purposes and highlights rows based on how long they ran relative to the entire query.

This can be seen in the **timeSum** column as follows:

- **Rows highlighted red** - longest runtime
- **Rows highlighted orange** - medium runtime
- **Rows highlighted yellow** - shortest runtime

*Back to Viewing Statement and Query Results from the Results Panel*

#### 9.3.4.3 Viewing Wrapped Strings in the SQL View

The SQL View panel allows you to more easily view certain queries, such as a long string that appears on one line. The SQL View makes it easier to see by wrapping it so that you can see the entire string at once. It also reformats and organizes query syntax entered in the Statement panel for more easily locating particular segments of your queries. The SQL View is identical to the **Format SQL** feature in the Toolbar, allowing you to retain your originally constructed query while viewing a more intuitively structured snapshot of it.

*Back to Viewing Statement and Query Results from the Results Panel*

*Back to Executing Statements and Running Queries from the Editor*

## 9.4 Viewing Logs

The **Logs** screen is used for viewing logs and includes the following elements:

| Element | Description |
|---|---|
| *Filter area* | Lets you filter the data shown in the table. |
| *Query tab* | Shows basic query information logs, such as query number and the time the query was run. |
| *Session tab* | Shows basic session information logs, such as session ID and user name. |
| *System tab* | Shows all system logs. |

### 9.4.1 Filtering Table Data

From the Logs tab, from the **FILTERS** area you can also apply the **TIMESPAN**, **ONLY ERRORS**, and additional filters (**Add**). The **Timespan** filter lets you select a timespan. The **Only Errors** toggle button lets you show all queries, or only queries that generated errors. The **Add** button lets you add additional filters to the data shown in the table. The **Filter** button applies the selected filter(s).

Other filters require you to select an item from a dropdown menu:

- INFO
- WARNING
- ERROR
- FATAL
- SYSTEM

You can also export a record of all of your currently filtered logs in Excel format by clicking **Download** located above the Filter area.

*Back to Viewing Logs*

### 9.4.2 Viewing Query Logs

The **QUERIES** log area shows basic query information, such as query number and the time the query was run. The number next to the title indicates the amount of queries that have been run.

From the Queries area you can see and sort by the following:

- Query ID
- Start time
- Query
- Compilation duration
- Execution duration
- Total duration
- Details (execution details, error details, successful query details)

In the Queries table, you can click on the **Statement ID** and **Query** items to set them as your filters. In the **Details** column you can also access additional details by clicking one of the **Details** options for a more detailed explanation of the query.

*Back to Viewing Logs*

### 9.4.3 Viewing Session Logs

The **SESSIONS** tab shows the sessions log table and is used for viewing activity that has occurred during your sessions. The number at the top indicates the amount of sessions that have occurred.

From here you can see and sort by the following:

- Timestamp
- Connection ID
- Username
- Client IP
- Login (Success or Failed)
- Duration (of session)
- Configuration Changes

In the Sessions table, you can click on the **Timestamp**, **Connection ID**, and **Username** items to set them as your filters.

*Back to Viewing Logs*

### 9.4.4 Viewing System Logs

The **SYSTEM** tab shows the system log table and is used for viewing all system logs. The number at the top indicates the amount of sessions that have occurred. Because system logs occur less frequently than queries and sessions, you may need to increase the filter timespan for the table to display any system logs.

From here you can see and sort by the following:

- Timestamp
- Log type

- Message

In the Systems table, you can click on the **Timestamp** and **Log type** items to set them as your filters. In the **Message** column, you can also click on an item to show more information about the message.

*Back to Viewing Logs*

## 9.5 Creating, Assigning, and Managing Roles and Permissions

The **Creating, Assigning, and Managing Roles and Permissions** describes the following:



### 9.5.1 Overview

In the **Roles** area you can create and assign roles and manage user permissions.

The **Type** column displays one of the following assigned role types:

| Role Type | Description |
|---|---|
| Groups | Roles with no users. |
| Enabled users | Users with log-in permissions and a password. |
| Disabled users | Users with log-in permissions and with a disabled password. An admin may disable a user's password permissions to temporary disable access to the system. |

**Note:** If you disable a password, when you enable it you have to create a new one.

*Back to Creating, Assigning, and Managing Roles and Permissions*

### 9.5.2 Viewing Information About a Role

Clicking a role in the roles table displays the following information:

- **Parent Roles** - displays the parent roles of the selected role. Roles inherit all roles assigned to the parent.

- **Members** - displays all members that the role has been assigned to. The arrow indicates the roles that the role has inherited. Hovering over a member displays the roles that the role is inherited from.

- **Permissions** - displays the role's permissions. The arrow indicates the permissions that the role has inherited. Hovering over a permission displays the roles that the permission is inherited from.

*Back to Creating, Assigning, and Managing Roles and Permissions*

### 9.5.3 Creating a New Role

You can create a new role by clicking **New Role**.

An admin creates a **user** by granting login permissions and a password to a role. Each role is defined by a set of permissions. An admin can also group several roles together to form a **group** to manage them simultaneously. For example, permissions can be granted to or revoked on a group level.

Clicking **New Role** lets you do the following:

- Add and assign a role name (required)
- Enable or disable log-in permissions for the role.
- Set a password.
- Assign or delete parent roles.
- Add or delete permissions.
- Grant the selected user with superuser permissions.

From the New Role panel you view directly and indirectly (or inherited) granted permissions. Disabled permissions have no connect permissions for the referenced database and are displayed in gray text. You can add or remove permissions from the **Add permissions** field. From the New Role panel you can also search and scroll through the permissions. In the **Search** field you can use the **and** operator to search for strings that fulfill multiple criteria.

When adding a new role, you must select the **Enable login for this role** and **Has password** check boxes.

In the **Password** and **Confirm password** fields grant a password. When you click in the **Password** field, a tooltip is displayed listing the password requirements.

*Back to Creating, Assigning, and Managing Roles and Permissions*

### 9.5.4 Editing a Role

Once you've created a role, clicking the **Edit Role** button lets you do the following:

- Edit the role name.
- Enable or disable log-in permissions.
- Set a password.
- Assign or delete parent roles.
- Assign a role **administrator** permissions.
- Add or delete permissions.
- Grant the selected user with superuser permissions.

From the Edit Role panel you view directly and indirectly (or inherited) granted permissions. Disabled permissions have no connect permissions for the referenced database and are displayed in gray text. You can add or remove permissions from the **Add permissions** field. From the Edit Role panel you can also search and scroll through the permissions. In the **Search** field you can use the **and** operator to search for strings that fulfill multiple criteria.

*Back to Creating, Assigning, and Managing Roles and Permissions*

### 9.5.5 Deleting a Role

Clicking the **delete** icon displays a confirmation message with the amount of users and groups that will be impacted by deleting the role.

*Back to Creating, Assigning, and Managing Roles and Permissions*

# CHAPTER TEN

# SYSTEM ARCHITECTURE

This topic includes guides that walk an end-user, database administrator, or system architect through the main ideas behind SQream DB.

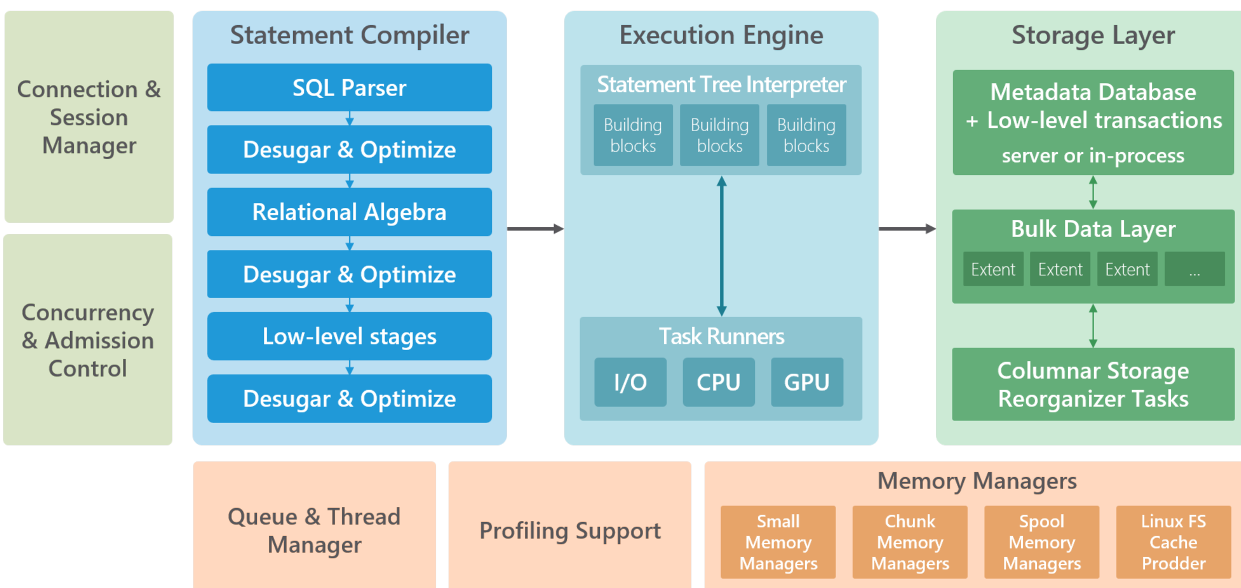
While SQream DB has many similarities to other database management systems, it has some unique and additional capabilities.

Explore the guides below for information about SQream DB's architecture.

## 10.1 Internals and architecture

### 10.1.1 SQream DB internals

Here is a high level architecture diagram of SQream DB's internals.



#### 10.1.1.1 Statement compiler

The statement compiler is written in Haskell. This takes SQL text and produces an optimised statement plan.

#### 10.1.1.2 Concurrency and concurrency control

The execution engine in SQream DB is built around thread workers with message passing. It uses threads to overlap different kinds of operations (including IO and GPU operations with CPU operations), and to accelerate CPU intensive operations.

#### 10.1.1.3 Transactions

SQream DB has serializable transactions, with these features:

- Serializable, with any kind of statement
- Run multiple SELECT queries concurrently with anything
- Run multiple inserts to the same table at the same time
- Cannot run multiple statements in a single transaction
- Other operations such as delete, truncate, and DDL use *coarse-grained exclusive locking*.

#### 10.1.1.4 Storage

The storage is split into the metadata layer and an append-only/ garbage collected bulk data layer.

##### 10.1.1.4.1 Metadata layer

The metadata layer uses LevelDB, and uses LevelDB's snapshot and write atomic features as part of the transaction system.

The metadata layer, together with the append-only bulk data layer help ensure consistency.

##### 10.1.1.4.2 Bulk data layer

The bulk data layer is comprised of extents, which are optimised for IO performance as much as possible. Inside the extents, are chunks, which are optimised for processing in the CPU and GPU. Compression is used in the extents and chunks.

When you run small inserts, you will get less optimised chunks and extents, but the system is designed to both be able to still run efficiently on this, and to be able to reorganise them transactionally in the background, without blocking DML operations. By writing small chunks in small inserts, then reorganising later, it supports both fast medium sized insert transactions and fast querying.

#### 10.1.1.5 Building blocks

The heavy lifting in SQream DB is done by single purpose C++/CUDA building blocks.

These are purposely designed to not be smart - they have to be instructed exactly what to do.

Most of the intelligence in piecing things together is in the statement compiler.

### 10.1.2 Columnar

Like many other analytical database management systems, SQream DB uses a column store for tables.

Column stores offer better I/O and performance with analytic workloads. Columns also compress much better, and lend themselves well to bulk data.

### 10.1.3 GPU usage

SQream DB uses GPUs for accelerating database operations. This acceleration brings additional benefit to columnar data processing.

SQream DB's GPU acceleration is integral to database operations. It is not an additional feature, but rather core to most data operations, e.g. `GROUP BY`, scalar functions, `JOIN`, `ORDER BY`, and more.

Using a GPU is an extended form of SIMD (Single-instruction, multiple data) intended for high throughput operations. When GPU acceleraiton is used, SQream DB uses special building blocks to take advantage of the high degree of parallelism of the GPU. This means that GPU operations use a single instruction that runs on multiple values.

## 10.2 Filesystem and usage

SQream DB writes and reads data from disk.

The SQream DB storage directory, sometimes refered to as a **storage cluster** is a collection of database objects, metadata database, and logs.

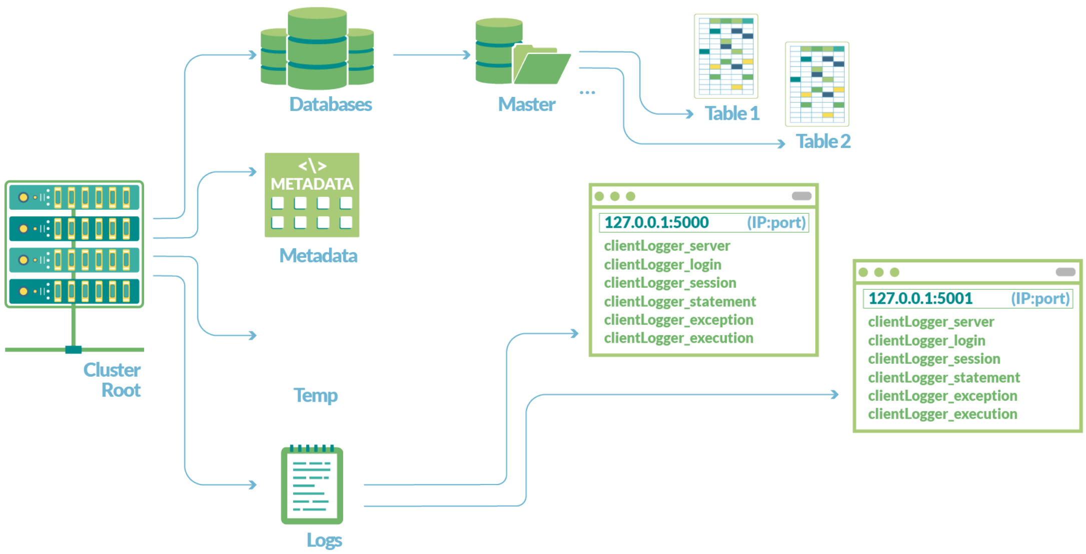
Each SQream DB worker and the metadata server must have access to the storage cluster in order to function properly.

### 10.2.1 Directory organization

The **cluster root** is the directory in which all data for SQream DB is stored.

**SQream DB storage cluster directories**

- *databases*
- *metadata or leveldb*
- *temp*
- *logs*




#### 10.2.1.1 `databases`

The databases directory houses all of the actual data in tables and columns.

Each database is stored as it's own directory. Each table is stored under it's respective database, and columns are stored in their respective table.

In the example above, the database named `retail` contains a table directory with a directory named `23`.

**Tip:** To find table IDs, use a catalog query:

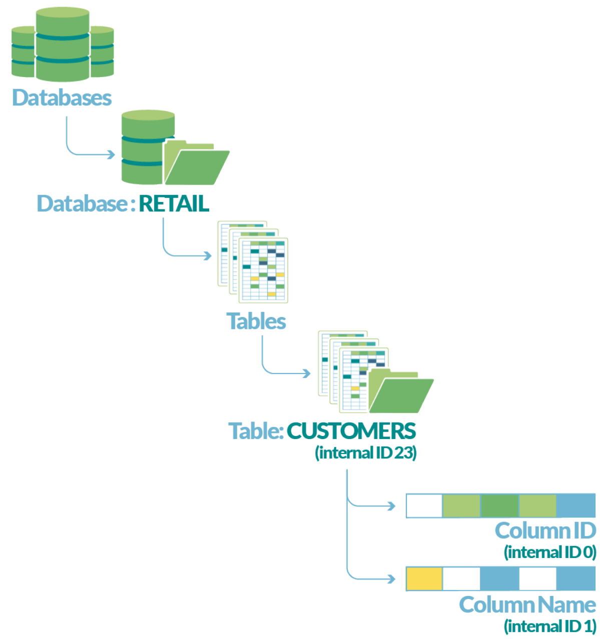
```
master=> SELECT table_name, table_id FROM sqream_catalog.tables WHERE table_name =
↪'customers';
table_name | table_id
-----------+---------
 customers |       23
```

Each table directory contains a directory for each physical column. An SQL column may be built up of several physical columns (e.g. if the data type is nullable).

**Tip:** To find column IDs, use a catalog query:

```
master=> SELECT column_id, column_name FROM sqream_catalog.columns WHERE table_id=23;
column_id | column_name
----------+------------
        0 | name@null
        1 | name@val
        2 | age@null
        3 | age@val
```

(continues on next page)




(continued from previous page)

```
        4 | email@null
        5 | email@val
```

Each column directory will contain extents, which are collections of chunks.

#### 10.2.1.2 `metadata` or `leveldb`

SQream DB's metadata is an embedded key-value store, based on LevelDB. LevelDB helps SQream DB ensure efficient storage for keys, handle atomic writes, snapshots, durability, and automatic recovery.

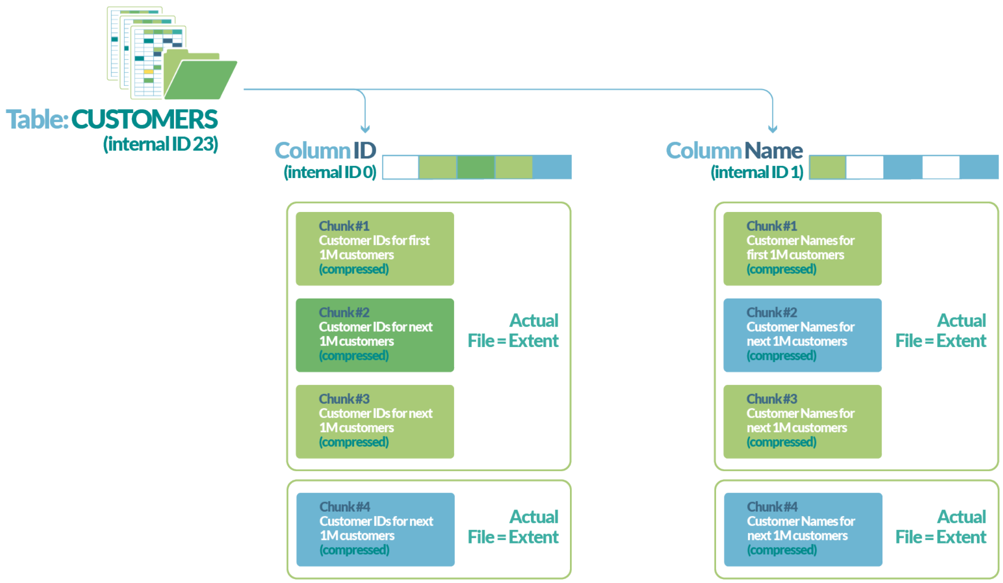
The metadata is where all database objects are stored, including roles, permissions, database and table structures, chunk mappings, and more.

#### 10.2.1.3 `temp`

The `temp` directory is where SQream DB writes temporary data.

The directory to which SQream DB writes temporary data can be changed to any other directory on the filesystem. SQream recommends remapping this directory to a fast local storage to get better performance when executing intensive larger-than-RAM operations like sorting. SQream recommends an SSD or NVMe drive, in mirrored RAID 1 configuration.

If desired, the `temp` folder can be redirected to a local disk for improved performance, by setting the `tempPath` setting in the configuration file.




#### 10.2.1.4 `logs`

The logs directory contains logs produced by SQream DB.

See more about the logs in the *Logging* guide.

CHAPTER

# ELEVEN

# CONFIGURATION GUIDES

The **Configuration Guides** page describes the following configuration information:

## 11.1 Configuring the Spooling Feature

The **Configuring the Spooling Feature** page includes the following topics:



### 11.1.1 Overview

From the SQream Acceleration Studio you can allocate the amount of memory (GB) available to the server for spooling using the `spoolMemoryGB` flag. SQream recommends setting the `spoolMemoryGB` flag to 90% of the `limitQueryMemoryGB` flag. The `limitQueryMemoryGB` flag is the total memory you've allocated for processing queries.

In addition, the `limitQueryMemoryGB` defines how much total system memory is used by each worker. SQream recommends setting `limitQueryMemoryGB` to 5% less than the total host memory divided by the amount of `sqreamd` workers on host.

Note that `spoolMemoryGB` must bet set to less than the `limitQueryMemoryGB`.

### 11.1.2 Example Configurations

The **Example Configurations** section shows the following example configurations:

#### 11.1.2.1 Example 1 - Recommended Settings

The following is an example of the recommended settings for a machine with 512GB of RAM and 4 workers:

```
limitQueryMemoryGB - ⌊(512 * 0.95 / 4)⌋ → ~ 486 / 4 → 121
spoolMemoryGB - ⌊( 0.9 * limitQueryMemoryGB )⌋ → ⌊( 0.9 * 121 )⌋ → 108
```

#### 11.1.2.2 Example 2 - Setting Spool Memory

The following is an example of setting spoolMemoryGB value in the current configuration method per-worker for 512GB of RAM and 4 workers:

```
{
    “cluster”: “/home/test_user/sqream_testing_temp/sqreamdb”,
    “gpu”:  0,
    “licensePath”: “home/test_user/SQream/tests/license.enc”,
    “machineIP”: “127.0.0.1”,
    “metadataServerIp”: “127.0.0.1”,
    “metadataServerPort”: “3105,
    “port”: 5000,
    “useConfigIP”” true,
    “limitQueryMemoryGB" : 121,
    “spoolMemoryGB" : 108
    “legacyConfigFilePath”: “home/SQream_develop/SqrmRT/utils/json/legacy_congif.json”
}
```

The following is an example of setting spoolMemoryGB value in the previous configuration method per-worker for 512GB of RAM and 4 workers:

```
“runtimeFlags”: {
“limitQueryMemoryGB” : 121,
“spoolMemoryGB” : 108
```

For more information about configuring the spoolMemoryGB flag, see the following:

- Current configuration method
- Previous configuration method

## 11.2 Configuring SQream

The **Configuring SQream** page describes the following configuration topics:

### 11.2.1 Configuration Levels

SQream’s configuration parameters are based on the following hierarchy:

- *Cluster-Based Configuration*
- *Worker-Based Configuration*
- *Session-Based Configuration*

#### 11.2.1.1 Cluster-Based Configuration

Cluster-based configuration lets you centralize configurations for all workers on the cluster. Only Regular and Cluster flag types can be modified on the cluster level. These modifications are persistent and stored at the metadata level, which are applied globally to all workers in the cluster.

**Note:** While cluster-based configuration was designed for configuring Workers, you can only configure Worker values set to the Regular or Cluster type.

#### 11.2.1.2 Worker-Based Configuration

Worker-based configuration lets you modify the configuration belong to individual workers from the worker configuration file.

For more information on making configurations from the worker configuration file, see Modifying Your Configuration Using a Legacy Configuration File.

#### 11.2.1.3 Session-Based Configuration

Session-based configurations are not persistent and are deleted when your session ends. This method enables you to modify all required configurations while avoiding conflicts between flag attributes modified on different devices at different points in time. The **SET flag_name** command is used to modify flag values on the session level. Any modifications you make with the **SET flag_name** command apply only to your open session, and are not saved when it ends.

For example, when the query below has completed executing, the values configured will be restored to its previous setting:

```
set spoolMemoryGB=700;
select * from table a where date='2021-11-11'
```

### 11.2.2 Flag Types

SQream uses three flag types, **Cluster**, **Worker**, and **Regular**. Each of these flag types is associated with one of three hierarchical configuration levels described earlier, making it easier to configure your system.

The highest level in the hierarchy is Cluster, which lets you set configurations across all workers in a given cluster. Modifying cluster values is **persistent**, meaning that any configurations you set are retained after shutting down your system. Configurations set at the Cluster level take the highest priority and override settings made on the Regular and Worker level **Comment** - *Confirm*. This is known as **cluster-based configuration**. Note that Cluster-based configuration lets you modify Cluster *and* Regular flag types. An example of a Cluster flag is **persisting your cache directory.**

The second level is Worker, which lets you configure individual workers. Modifying Worker values are also **persistent**. This is known as **worker-based configuration**. Some examples of Worker flags includes **setting total device memory usage** and **setting metadata server connection port**.

The lowest level is Regular, which means that modifying values of Regular flags affects only your current session and are not persistent. This means that they are automatically restored to their default value when the session ends. This is known as **session-based configuration**. Some examples of Regular flags includes **setting your bin size** and **setting CUDA memory**.

To see each flag's default value, see one of the following:

- The **Default Value** column in the All Configurations section.

- The flag's individual description page, such as Setting CUDA Memory.

### 11.2.3 Configuration Roles

SQream divides flags into the following roles, each with their own set of permissions:

- *Administration Flags* - can be modified by administrators on a session and cluster basis using the `ALTER SYSTEM SET` command: **Comment** - *I don't think we need to mention the command here, as it's described below, and also not mentioned for Generic Flags.*
  - Regular
  - Worker
  - Cluster
- *Generic Flags* - can be modified by standard users on a session basis:
  - Regular
  - Worker

### 11.2.4 Modification Methods

SQream provides two different ways to modify your configurations. The current method is based on hierarchical configuration as described above. This method is based on making modifications on the **worker configuration file**, while you can still make modifications using the previous method using the **legacy configuration file**, both described below:

- *Modifying Your Configuration Using the Worker Configuration File*
- *Modifying Your Configuration Using a Legacy Configuration File*

#### 11.2.4.1 Modifying Your Configuration Using the Worker Configuration File

You can modify your configuration using the **worker configuration file (config.json)**. Changes that you make to worker configuration files are persistent. Note that you can only set the attributes in your worker configuration file **before** initializing your SQream worker, and while your worker is active these attributes are read-only.

The following is an example of a worker configuration file:

```
{
    “cluster”: “/home/test_user/sqream_testing_temp/sqreamdb”,
    “gpu”:  0,
    “licensePath”: “home/test_user/SQream/tests/license.enc”,
    “machineIP”: “127.0.0.1”,
    “metadataServerIp”: “127.0.0.1”,
    “metadataServerPort”: “3105,
    “port”: 5000,
    “useConfigIP”” true,
    “legacyConfigFilePath”: “home/SQream_develop/SqrmRT/utils/json/legacy_congif.json”
}
```

You can access the legacy configuration file from the `legacyConfigFilePath` parameter shown above. If all (or most) of your workers require the same flag settings, you can set the `legacyConfigFilePath` attribute to the same legacy file.

#### 11.2.4.2 Modifying Your Configuration Using a Legacy Configuration File

You can modify your configuration using a legacy configuration file.

The Legacy configuration file provides access to the read/write flags used in SQream's previous configuration method. A link to this file is provided in the **legacyConfigFilePath** parameter in the worker configuration file.

The following is an example of the legacy configuration file:

```
{
   “developerMode”: true,
   “reextentUse”: false,
   “useClientLog”: true,
   “useMetadataServer”” false
}
```

For more information on using the previous configuration method, see *Configuring SQream Using the Previous Configuration Method*.

### 11.2.5 Configuring Your Parameter Values

The method you must use to configure your parameter values depends on the configuration level. Each configuration level has its own command or set of commands used to configure values, as shown below:

| Configuration Level | | |
|---|---|---|
| **Regular, Worker, and Cluster** | | |
| **Command** | **Description** | **Example** |
| `SET <flag_name>` | Used for modifying flag attributes. | `SET developerMode=true` |
| `SHOW <flag-name> / ALL` | Used to preset either a specific flag value or all flag values. | `SHOW <heartbeatInterval>` |
| `SHOW ALL LIKE` | Used as a wildcard character for flag names. | `SHOW <heartbeat*>` |
| `show_conf_UF` | Used to print all flags with the following attributes:<br>• Flag name<br>• Default value<br>• Is Developer Mode (Boolean)<br>• Flag category<br>• Flag type | `rechunkThreshold,90, true,RND,regular` |
| `show_conf_extended UF` | Used to print all information output by the show_conf UF command, in addition to description, usage, data type, default value and range. | `rechunkThreshold,90, true,RND,regular` |
| `show_md_flag UF` | Used to show a specific flag/all flags stored in the metadata file. | • Example 1: `* master=> ALTER SYSTEM SET heartbeatTimeout=111;`<br>• Example 2: `* master=> select show_md_flag('all'); heartbeatTimeout, 111`<br>• Example 3: `* master=> select show_md_flag('heartbeatTimeout'); heartbeatTimeout, 111` |
| **Worker and Cluster** | | |
| `ALTER SYSTEM SET <flag-name>` | Used for storing or modifying flag attributes in the metadata file. | `ALTER SYSTEM SET <heartbeatInterval=12;>` |
| `ALTER SYSTEM RESET <flag-name / ALL>` | Used to remove a flag or all flag attributes from the metadata file. | `ALTER SYSTEM RESET <heartbeatInterval ALTER SYSTEM RESET ALL>` |

## 11.2.6 Command Examples

This section includes the following command examples:



### 11.2.6.1 Running a Regular Flag Type Command

The following is an example of running a **Regular** flag type command:

```
SET spoolMemoryGB= 11;
executed
```

### 11.2.6.2 Running a Worker Flag Type Command

The following is an example of running a **Worker** flag type command:

```
SHOW spoolMemoryGB;
```

### 11.2.6.3 Running a Cluster Flag Type Command

The following is an example of running a **Cluster** flag type command:

```
ALTER SYSTEM RESET useMetadataServer;
executed
```

## 11.2.7 Showing All Flags in the Catalog Table

SQream uses the **sqream_catalog.parameters** catalog table for showing all flags, providing the scope (default, cluster and session), description, default value and actual value.

The following is the correct syntax for a catalog table query:

```
SELECT * FROM sqream_catalog.settings
```

The following is an example of a catalog table query:

```
externalTableBlobEstimate, 100, 100, default,
varcharEncoding, ascii, ascii, default, Changes the expected encoding for Varchar␣
↪columns
useCrcForTextJoinKeys, true, true, default,
hiveStyleImplicitStringCasts, false, false, default,
```

### 11.2.8 All Configurations

The following table describes all **Generic** and **Administration** configuration flags:

| Flag Name | Access Control | Modification Type | Description |
|---|---|---|---|
| binSizes | Admin | Regular | Sets the custom bin size in the cache to enable high g |
| cacheDiskDir | Generic | Regular | Sets the ondisk directory location for the spool to sav |
| cacheDiskGB | Generic | Regular | Sets the amount of memory (GB) to be used by Spoo |
| cacheEvictionMilliseconds | Generic | Regular | Sets how long the cache stores contents before being |
| cachePartitions | Generic | Regular | Sets the number of partitions that the cache is split in |
| cachePersistentDir | Generic | Regular | Sets the persistent directory location for the spool to s |
| cachePersistentGB | Generic | Regular | Sets the amount of data (GB) for the cache to store p |
| cacheRamGB | Generic | Regular | Sets the amount of memory (GB) to be used by Spoo |
| checkCudaMemory | Admin | Regular | Sets the pad device memory allocations with safety b |
| compilerGetsOnlyUFs | Admin | Regular | Sets the runtime to pass only utility functions names t |
| copyToRestrictUtf8 | Admin | Regular | Sets the custom bin size in the cache to enable high g |
| cpuReduceHashtableSize | Admin | Regular | Sets the hash table size of the CpuReduce. |
| csvLimitRowLength | Admin | Cluster | Sets the maximum supported CSV row length. |
| cudaMemcpyMaxSizeBytes | Admin | Regular | Sets the chunk size for copying from CPU to GPU. If |
| CudaMemcpySynchronous | Admin | Regular | Indicates if copying from/to GPU is synchronous. |
| cudaMemQuota | Admin | Worker | Sets the percentage of total device memory to be use |
| developerMode | Admin | Regular | Enables modifying R&D flags. |
| enableDeviceDebugMessages | Admin | Regular | Activates the Nvidia profiler (nvprof) markers. |
| enableLogDebug | Admin | Regular | Enables creating and logging in the clientLogger_deb |
| enableNvprofMarkers | Admin | Regular | Activates the Nvidia profiler (nvprof) markers. |
| endLogMessage | Admin | Regular | Appends a string at the end of every log line. |
| extentStorageFileSizeMB | Admin | Cluster | Sets the minimum size in mebibytes of extents for tab |
| flipJoinOrder | Generic | Regular | Reorders join to force equijoins and/or equijoins sort |
| gatherMemStat | Admin | Regular | Monitors all pinned allocations and all memcopies to/ |
| increaseChunkSizeBeforeReduce | Admin | Regular | Increases the chunk size to reduce query speed. |
| increaseMemFactors | Admin | Regular | Adds rechunker before expensive chunk producer. |
| leveldbWriteBufferSize | Admin | Regular | Sets the buffer size. |
| limitQueryMemoryGB | Generic | Worker | Prevents a query from processing more memory than |
| logSysLevel | Generic | Regular | Determines the client log level: 0 - L_SYSTEM, 1 - I |
| machineIP | Admin | Worker | Manual setting of reported IP. |
| maxAvgBlobSizeToCompressOnGpu | Generic | Regular | Sets the CPU to compress columns with size above (f |
| memoryResetTriggerMB | Admin | Regular | Sets the size of memory used during a query to trigge |
| metadataServerPort | Admin | Worker | Sets the port used to connect to the metadata server. |
| mtRead | Admin | Regular | Splits large reads to multiple smaller ones and execute |
| mtReadWorkers | Admin | Regular | Sets the number of workers to handle smaller concurr |
| orcImplicitCasts | Admin | Regular | Sets the implicit cast in orc files, such as int to tinyint |
| sessionTag | Generic | Regular | Sets the name of the session tag. |
| spoolMemoryGB | Generic | Regular | Sets the amount of memory (GB) to be used by the se |
| statementLockTimeout | Admin | Regular | Sets the timeout (seconds) for acquiring object locks |
| useConfigIP | Admin | Worker | Activates the machineIP (true). Setting to false ignore |
| useLegacyDecimalLiterals | Admin | Regular | Interprets decimal literals as Double instead of Nume |
| useLegacyStringLiterals | Admin | Regular | Interprets ASCII-only strings as VARCHAR instead o |
| varcharIdentifiers | Admin | Regular | Activates using varchar as an identifier. |

# 11.3 Configuration Flags

SQream provides two methods for configuration your instance of SQream. The current configuration method is based on cluster and session-based configuration, described in more detail below. Users can also use the previous configuration method done using a configuration file.

The **Configuration Methods** page describes the following configurations methods:

## 11.3.1 Administration Flags

The **Administration Flags** page describes the following flag types, which can be modified by administrators on a session and cluster basis using the `ALTER SYSTEM SET` command:

### 11.3.1.1 Regular Administration Flags

The **Regular Administration Flags** page describes **Regular** modification type flags, which can be modified by administrators on a session and cluster basis using the `ALTER SYSTEM SET` command:

- Setting Bin Size
- Setting CUDA Memory
- Limiting Runtime to Utility Functions
- Enabling High Bin Control Granularity
- Reducing CPU Hashtable Sizes
- Setting Chunk Size for Copying from CPU to GPU
- Indicating GPU Synchronicity
- Enabling Modification of R&D Flags
- Checking for Post-Production CUDA Errors
- Enabling Modification of clientLogger_debug File
- Activating the NVidia Profiler Markers
- Appending String at End of Log Lines
- Monitoring and Printing Pinned Allocation Reports
- Increasing Chunk Size to Reduce Query Speed
- Adding Rechunker before Expensing Chunk Producer
- Setting the Buffer Size
- Maximum Pinned Percentage of Total RAM
- Setting Memory Used to Abort Server
- Splitting Large Reads for Concurrent Execution
- Setting Worker Amount to Handle Concurrent Reads
- Setting Implicit Casts in ORC Files
- Setting Timeout Limit for Locking Objects before Executing Statements
- Interpreting Decimal Literals as Double Instead of Numeric

- Interpreting VARCHAR as TEXT
- VARCHAR Identifiers

#### 11.3.1.2 Cluster Administration Flags

The **Cluster Administration Flags** page describes **Cluster** modification type flags, which can be modified by administrators on a session and cluster basis using the `ALTER SYSTEM SET` command:

- Setting Maximum CSV Row Length

#### 11.3.1.3 Worker Administration Flags

The **Worker Administration Flags** page describes **Worker** modification type flags, which can be modified by administrators on a session and cluster basis using the `ALTER SYSTEM SET` command:

- Setting Total Device Memory Usage in SQream Instance
- Enabling Manually Setting Reported IP
- Setting Port Used for Metadata Server Connection
- Assigning Local Network IP
- Enabling the Query Healer
- Configuring the Query Healer
- Adjusting Permitted Log-in Attempts

### 11.3.2 Generic Flags

The **Generic Flags** page describes the following flag types, which can be modified by standard users on a session basis:

#### 11.3.2.1 Regular Generic Flags

The **Regular Generic Flags** page describes **Regular** modification type flags, which can be modified by standard users on a session basis:

- Flipping Join Order to Force Equijoins
- Determining Client Level
- Setting CPU to Compress Defined Columns
- Setting Query Memory Processing Limit
- Setting the Spool Memory
- Setting Cache Partitions
- Setting Cache Flushing
- Setting InMemory Spool Memory
- Setting Disk Spool Memory
- Setting Spool Saved File Directory Location
- Setting Data Stored Persistently on Cache

- Setting Persistent Spool Saved File Directory Location
- Setting Session Tag Name

#### 11.3.2.2 Worker Generic Flags

The **Worker Generic Flags** page describes **Worker** modification type flags, which can be modified by standard users on a session basis:

- Limits Available Query Processing Memory

## 11.4 Configuring SQream Using the Previous Configuration Method

The **Configuring SQream Using the Previous Configuration Method** page describes SQream's previous method for configuring your instance of SQream, and includes the following topics:

- *Frequently Set Parameters*
- *Recommended Configuration File*

By default, configuration files are stored in `/etc/sqream`.

A very minimal configuration file looks like this:

```
{
    "compileFlags": {
    },
    "runtimeFlags": {
    },
    "runtimeGlobalFlags": {
    },
    "server": {
        "gpu": 0,
        "port": 5000,
        "cluster": "/home/sqream/sqream_storage",
        "licensePath": "/etc/sqream/license.enc"
    }
}
```

- Each SQream DB worker (`sqreamd`) has a dedicated configuration file.
- The configuration file contains four distinct sections, `compileFlags`, `runtimeFlags`, `runtimeGlobalFlags`, and `server`.

In the example above, the worker will start on port 5000, and will use GPU #0.

### 11.4.1 Frequently Set Parameters

Table 2: Server flags

| Name | Section | Description | Default | Value range | Example |
|---|---|---|---|---|---|
| `gpu` | `server` | Controls the GPU ordinal to use | ✗ | 0 to (number of GPUs in the machine -1). Check with `nvidia-smi -L` | `"gpu": 0` |
| `port` | `server` | Controls the TCP port to listen on | ✗ | 1024 to 65535 | `"port" : 5000` |
| `ssl_port` | `server` | Controls the SSL TCP port to listen on. Must be different from `port` | ✗ | 1024 to 65535 | `"ssl_port" : 5100` |
| `cluster` | `server` | Specifies the cluster path root | ✗ | Valid local system path | `"cluster" : "/home/sqream/sqream_storage"` |
| `license_path` | `server` | Specifies the license file for this worker | ✗ | Valid local system path to license file | `"license_path" : "/etc/sqream/license.enc"` |

Table 3: Runtime global flags

| Name | Section | Description | Default | Value range | Example |
|---|---|---|---|---|---|
| spoolMemoryGb | runtimeGlobalFlags | Modifies RAM allocated for the worker for intermediate results. Statements that use more memory than this setting will spool to disk, which could degrade performance. We recommend not to exceed the amount of RAM in the machine. This setting must be set lower than the `limitQueryMemoryGB` setting. | `128` | 1 to maximum available RAM in gigabytes. | `"spoolMemoryGb": 250` |
| limitQueryMemoryGB | runtimeGlobalFlags | Modifies the maximum amount of RAM allocated for a query. The recommended value for this is `total host memory` / `sqreamd workers on host`. For example, for a machine with 512GB of RAM and 4 workers, the recommended setting is `512/4 → 128`. | `10000` | `1` to `10000` | `"limitQueryMemoryGB" : 128` |
| cudaMemQuota | runtimeGlobalFlags | Modifies the maximum amount of GPU RAM allocated for a worker. The recommended value is 99% for a GPU with a single worker, or 49% for a GPU with two workers. | `90 %` | `1` to `99` | `"cudaMemQuota" : 99` |
| showFullExceptionInfo | runtimeGlobalFlags | Shows complete error message with debug information. Use this for debugging. | `false` | `true` or `false` | `"showFullExceptionInfo" : true` |
| initialSubscribedServices | runtimeGlobalFlags | Comma separated list of *service queues* that the worker is subscribed to | `"sqream"` | Comma separated list of service names, with no spaces. Services that don't exist will be created. | `"initialSubscribedServices": "sqream, etl, management"` |
| logClientLevel | runtimeGlobalFlags | Used to control which log level should appear in the logs | `4` (`INFO`) | `0` SYSTEM (lowest) - `4` INFO (highest). See *information level table* for explanation about these log levels. | `"logClientLevel" : 3` |
| nodeInfoLoggingSec | runtimeGlobalFlags | Sets an interval for automatically logging long-running statements' show_node_info output. Output is written as a message type `200`. | `60` (every minute) | Positive whole number >=1. | `"nodeInfoLoggingSec" : 5` |
| useLogMaxFileSize | runtimeGlobalFlags | Defines whether SQream logs should be cycled when they reach `logMaxFileSizeMB` size. When `true`, set the `logMaxFileSizeMB` accordingly. | `false` | `false` or `true` | `"useLogMaxFileSize" : true` |
| logMaxFileSizeMB | runtimeGlobalFlags | Sets the size threshold in megabytes after which a new log file will be opened. | `20` | `1` to `1024` (1MB to 1GB) | `"logMaxFileSizeMB" : 250` |
| logFileRotateTimeFrequency | runtimeGlobalFlags | Control frequency of log rotation | `never` | `daily`, `weekly`, `monthly`, `never` | `"logClientLevel" : 3` |

Table 4: Runtime flags

| Name | Section | Description | Default | Value range | Example |
|---|---|---|---|---|---|
| insertParsers | runtimeFlags | Sets the number of CSV parsing threads launched during bulk load | 4 | 1 to 32 | "insertParsers" : 8 |
| insertCompressors | runtimeFlags | Sets the number of compressor threads launched during bulk load | 4 | 1 to 32 | "insertCompressors" : 8 |
| statementLockTimeout | runtimeGlobalFlags | Sets the delay in seconds before SQream DB will stop waiting for a lock and return an error | 3 | >=1 | "statementLockTimeout" : 10 |

**Warning:** JSON files can't contain any comments.

### 11.4.2 Recommended Configuration File

```
{
   "compileFlags":{
   },
   "runtimeFlags":{
      "insertParsers": 16,
      "insertCompressors": 8
   },
   "runtimeGlobalFlags":{
      "limitQueryMemoryGB" : 121,
      "spoolMemoryGB" : 108,
      "cudaMemQuota": 90,
      "initialSubscribedServices" : "sqream",
      "useMetadataServer": true,
      "metadataServerIp": "127.0.0.1",
      "useConfigIP": true,
      "machineIP": "127.0.0.1"
   },
   "server":{
      "gpu":0,
      "port":5000,
      "ssl_port": 5100,
      "cluster":"/home/sqream/sqream_storage",
      "licensePath":"/etc/sqream/license.enc"
   }
}
```

CHAPTER

TWELVE

# REFERENCE GUIDES

The **Reference Guides** section provides reference for using SQream DB's interfaces and SQL features.

## 12.1 SQL Statements and Syntax

This section provides reference for using SQream DB's SQL statements - *DDL commands*, *DML commands* and *SQL query syntax*.

### 12.1.1 SQL Syntax Features

SQream DB supports SQL from the ANSI 92 syntax and describes the following:

- keywords_and_identifiers
- literals
- scalar_expressions
- joins
- common_table_expressions
- window_functions
- subqueries
- null_handling

### 12.1.2 SQL Statements

The **SQL Statements** page describes the following commands:

- *Data Definition Commands (DDL)*
- *Data Manipulation Commands (DML)*
- *Utility Commands*
- *Workload Management*
- *Access Control Commands*

SQream supports commands from ANSI SQL.

#### 12.1.2.1 Data Definition Commands (DDL)

The following table shows the Data Definition commands:

| Command | Usage |
|---|---|
| ADD_COLUMN | Add a new column to a table |
| ALTER_DEFAULT_SCHEMA | Change the default schema for a role |
| ALTER_TABLE | Change the schema of a table |
| CLUSTER_BY | Change clustering keys in a table |
| CREATE_DATABASE | Create a new database |
| CREATE_FOREIGN_TABLE | Create a new foreign table in the database |
| CREATE_FUNCTION | Create a new user defined function in the database |
| CREATE_SCHEMA | Create a new schema in the database |
| CREATE_TABLE | Create a new table in the database |
| CREATE_TABLE_AS | Create a new table in the database using results from a select query |
| CREATE_VIEW | Create a new view in the database |
| DROP_CLUSTERING_KEY | Drops all clustering keys in a table |
| DROP_COLUMN | Drop a column from a table |
| DROP_DATABASE | Drop a database and all of its objects |
| DROP_FUNCTION | Drop a function |
| DROP_SCHEMA | Drop a schema |
| DROP_TABLE | Drop a table and its contents from a database |
| DROP_VIEW | Drop a view |
| RENAME_COLUMN | Rename a column |
| RENAME_TABLE | Rename a table |

#### 12.1.2.2 Data Manipulation Commands (DML)

The following table shows the Data Manipulation commands:

| Command | Usage |
|---|---|
| CREATE_TABLE_AS | Create a new table in the database using results from a select query |
| DELETE | Delete specific rows from a table |
| COPY_FROM | Bulk load CSV data into an existing table |
| COPY_TO | Export a select query or entire table to CSV files |
| INSERT | Insert rows into a table |
| SELECT | Select rows and column from a table |
| TRUNCATE | Delete all rows from a table |
| UPDATE | Modify the value of certain columns in existing rows without creating a table |
| VALUES | Return rows containing literal values |

#### 12.1.2.3 Utility Commands

The following table shows the Utility commands:

| Command | Usage |
|---|---|
| EXPLAIN | Returns a static query plan, which can be used to debug query plans |
| SELECT GET_LICENSE_INFO | View a user's license information |
| SELECT GET_DDL | View the `CREATE TABLE` statement for a table |
| SELECT GET_FUNCTION_DDL | View the `CREATE FUNCTION` statement for a UDF |
| SELECT GET_VIEW_DDL | View the `CREATE VIEW` statement for a view |
| SELECT RECOMPILE_VIEW | Recreate a view after schema changes |
| SELECT DUMP_DATABASE_DDL | View the `CREATE TABLE` statement for an current database |
| SHOW CONNECTIONS | Returns a list of active sessions on the current worker |
| SHOW LOCKS | Returns a list of locks from across the cluster |
| SHOW NODE INFO | Returns a snapshot of the current query plan, similar to `EXPLAIN ANALYZE` from other databases |
| SHOW SERVER STATUS | Returns a list of active sessions across the cluster |
| SHOW VERSION | Returns the system version for SQream DB |
| SHUTDOWN_SERVER | Sets your server to finish compiling all active queries before shutting down according to a user-defined time value |
| STOP STATEMENT | Stops or aborts an active statement |

#### 12.1.2.4 Workload Management

The following table shows the Workload Management commands:

| Command | Usage |
|---|---|
| subscribe_service | Add a SQream DB worker to a service queue |
| unsubscribe_service | Remove a SQream DB worker from a service queue |
| show_subscribed_instances | Return a list of service queues and workers |

#### 12.1.2.5 Access Control Commands

The following table shows the Access Control commands:

| Command | Usage |
|---|---|
| alter_default_permissions | Applies a change to defaults in the current schema |
| alter_role | Applies a change to an existing role |
| create_role | Creates a roles, which lets a database administrator control permissions on tables and databases |
| drop_role | Removes roles |
| get_role_permissions | Returns all permissions granted to a role in table format |
| get_role_global_ddl | Returns the definition of a global role in DDL format |
| get_role_database_ddl | Returns the definition of a database role in DDL format |
| get_statement_permissions | Returns a list of permissions required to run a statement or query |
| grant | Grant permissions to a role |
| revoke | Revoke permissions from a role |
| rename_role | Rename a role |

### 12.1.3 SQL Functions

SQream supports functions from ANSI SQL, as well as others for compatibility.

#### 12.1.3.1 Summary of Functions

##### 12.1.3.1.1 Built-In Scalar Functions

For more information about built-in scalar functions, see *Built-In Scalar Functions*.

###### 12.1.3.1.1.1 Bitwise Operations

The following table shows the **bitwise operations** functions:

| Function | Description |
|---|---|
| bitwise_and | Bitwise AND |
| bitwise_not | Bitwise NOT |
| bitwise_or | Bitwise OR |
| bitwise_shift_left | Bitwise shift left |
| bitwise_shift_right | Bitwise shift right |
| bitwise_xor | Bitwise XOR |

###### 12.1.3.1.1.2 Conditionals

The following table shows the **conditionals** functions:

| Function | Description |
|---|---|
| between | Value is in [ or not within ] the range |
| case | Test a conditional expression, and depending on the result, evaluate additional expressions. |
| coalesce | Evaluate first non-NULL expression |
| in | Value is in [ or not within ] a set of values |
| isnull | Alias for coalesce with two expressions |
| is_ascii | Test a `TEXT` for ASCII-only characters |
| is_null | Check for `NULL` [ or non-`NULL` ] values |

###### 12.1.3.1.1.3 Conversion

The following table shows the **conversion** functions:

| Function | Description |
|---|---|
| from_unixts | Converts a UNIX Timestamp to `DATE` or `DATETIME` |
| to_hex | Converts a number to a hexadecimal string representation |
| to_unixts | Converts a `DATE` or `DATETIME` to a UNIX Timestamp |

#### 12.1.3.1.1.4 Date and Time

The following table shows the **date and time** functions:

| Function | Description |
|---|---|
| curdate | Special syntax, equivalent to current_date |
| current_date | Returns the current date as `DATE` |
| current_timestamp | Equivalent to getdate |
| datepart | Extracts a date or time element from a date expression |
| dateadd | Adds an interval to a date expression |
| datediff | Calculates the time difference between two date expressions |
| eomonth | Calculates the last day of the month of a given date expression |
| extract | ANSI syntax for extracting date or time element from a date expression |
| getdate | Returns the current timestamp as `DATETIME` |
| sysdate | Equivalent to getdate |
| date_trunc | Truncates a date element down to a specified date or time element |

#### 12.1.3.1.1.5 Numeric

The following table shows the **arithmetic operators**:

Table 1: Arithmetic Operators

| Operator | Syntax | Description |
|---|---|---|
| + (unary) | `+a` | Converts a string to a numeric value. Identical to `a :: double` |
| + | `a + b` | Adds two expressions together |
| - (unary) | `-a` | Negates a numeric expression |
| - | `a - b` | Subtracts `b` from `a` |
| * | `a * b` | Multiplies `a` by `b` |
| / | `a / b` | Divides `a` by `b` |
| % | `a % b` | Modulu of `a` by `b`. See also mod |

For more information about arithmetic operators, see arithmetic_operators.

The following table shows the **arithmetic operator** functions:

Table 2: Arithemtic Operator Functions

| Function | Description |
|---|---|
| abs | Calculates the absolute value of an argument |
| acos | Calculates the inverse cosine of an argument |
| asin | Calculates the inverse sine of an argument |
| atan | Calculates the inverse tangent of an argument |
| atn2 | Calculates the inverse tangent for a point (y, x) |
| ceiling | Calculates the next integer for an argument |
| cos | Calculates the cosine of an argument |
| cot | Calculates the cotangent of an argument |
| crc64 | Calculates a CRC-64 hash of an argument |
| degrees | Converts a value from radian values to degrees |
| exp | Calcalates the natural exponent for an argument ($e^x$) |
| floor | Calculates the largest integer smaller than the argument |
| log | Calculates the natural log for an argument |
| log10 | Calculates the 10-based log for an argument |
| mod | Calculates the modulu (remainder) of two arguments |
| pi | Returns the constant value for π |
| power | Calculates x to the power of y ($x^y$) |
| radians | Converts a value from degree values to radians |
| round | Rounds an argument down to the nearest integer, or an arbitrary precision |
| sin | Calculates the sine of an argument |
| sqrt | Calculates the square root of an argument ($\sqrt{x}$) |
| square | Raises an argument to the power of 2 ($x^y$) |
| tan | Calculates the tangent of an argument |
| trunc | Rounds a number to its integer representation towards 0 |

#### 12.1.3.1.1.6 Strings

The following table shows the **string** functions:

| Function | Description |
|---|---|
| char_length | Calculates number of characters in an argument |
| charindex | Calculates the position where a string starts inside another string |
| concat | Concatenates two strings |
| decode | Decodes or extracts binary data from a textual input string |
| isprefixof | Matches if a string is the prefix of another string |
| left | Returns the first number of characters from an argument |
| len | Calculates the length of a string in characters |
| like | Tests if a string argument matches a pattern |
| lower | Converts an argument to a lower-case equivalent |
| ltrim | Trims whitespaces from the left side of an argument |
| octet_length | Calculates the length of a string in bytes |
| patindex | Calculates the position where a pattern matches a string |
| regexp_count | Calculates the number of matches of a regular expression match in an argument |
| regexp_instr | Returns the start position of a regular expression match in an argument |
| regexp_replace | Replaces and returns the text column substrings of a regular expression match in an argument |
| regexp_substr | Returns a substring of an argument that matches a regular expression |
| repeat | Repeats a string as many times as specified |
| replace | Replaces characters in a string |
| reverse | Reverses a string argument |
| right | Returns the last number of characters from an argument |
| rlike | Tests if a string argument matches a regular expression pattern |
| rtrim | Trims whitespace from the right side of an argument |
| substring | Returns a substring of an argument |
| trim | Trims whitespaces from an argument |
| upper | Converts an argument to an upper-case equivalent |

#### 12.1.3.1.2 User-Defined Scalar Functions

For more information about user-defined scalar functions, see scalar_sql_udf.

#### 12.1.3.1.3 Aggregate Functions

The following table shows the **aggregate** functions:

| Function | Aliases | Description |
|---|---|---|
| avg | | Calculates the average of all of the values |
| corr | | Calculates the Pearson correlation coefficient |
| count | | Calculates the count of all of the values or only distinct values |
| covar_pop | | Calculates population covariance of values |
| covar_samp | | Calculates sample covariance of values |
| max | | Returns maximum value of all values |
| min | | Returns minimum value of all values |
| sum | | Calculates the sum of all of the values or only distinct values |
| stddev_samp | `stdev`, `stddev` | Calculates sample standard deviation of values |
| stddev_pop | `stdevp` | Calculates population standard deviation of values |
| var_samp | `var`, `variance` | Calculates sample variance of values |
| var_pop | `varp` | Calculates population variance of values |

For more information about aggregate functions, see *Aggregate Functions*.

#### 12.1.3.1.4 Window Functions

The following table shows the **window** functions:

| Function | Description |
|---|---|
| lag | Calculates the value evaluated at the row that is before the current row within the partition |
| lead | Calculates the value evaluated at the row that is after the current row within the partition |
| max | Calculates the maximum value |
| min | Calculates the minimum value |
| sum | Calculates the sum of all of the values |
| rank | Calculates the rank of a row |
| first_value | Returns the value in the first row of a window |
| last_value | Returns the value in the last row of a window |
| nth_value | Returns the value in a specified `(n)` row of a window |
| dense_rank | Returns the rank of the current row with no gaps |
| percent_rank | Returns the relative rank of the current row |
| cume_dist | Returns the cumulative distribution of rows |
| ntile | Returns an integer ranging between `1` and the argument value, dividing the partitions as equally as possible |

For more information about window functions, see window_functions.

#### 12.1.3.1.5 Workload Management Functions

The following table shows the **workload management** functions:

| Function | Description |
|---|---|
| subscribe_service | Add a SQream DB worker to a service queue |
| unsubscribe_service | Remove a SQream DB worker to a service queue |
| show_subscribed_instances | Return a list of service queues and workers |

##### 12.1.3.1.5.1 Built-In Scalar Functions

The **Built-In Scalar Functions** page describes functions that return one value per call:

- AND
- NOT
- OR
- SHIFT_LEFT
- SHIFT_RIGHT
- XOR
- between
- case
- coalesce
- decode
- in
- is_ascii
- is_null
- isnull
- from_unixts
- to_hex
- to_unixts
- curdate
- current_date
- current_timestamp
- dateadd
- datediff
- datepart
- eomonth
- extract
- getdate
- sysdate
- trunc
- abs
- acos
- asin
- atan
- atn2
- ceiling
- cos
- cot
- crc64
- degrees
- exp
- floor
- log
- log10
- mod
- pi
- power
- radians
- round
- sin
- sqrt
- square
- tan
- trunc
- char_length

- charindex
- concat
- isprefixof
- left
- len
- like
- lower
- ltrim
- octet_length
- patindex
- regexp_count
- regexp_instr
- regexp_replace
- regexp_substr
- repeat
- replace
- reverse
- right
- rlike
- rtrim
- substring
- trim
- upper

#### 12.1.3.1.5.2 User-Defined Functions

The following user-defined functions are functions that can be defined and configured by users.

The **User-Defined Functions** page describes the following:

- Python user-defined functions
- Scalar SQL user-defined functions

#### 12.1.3.1.5.3 Aggregate Functions

#### 12.1.3.1.5.4 Overview

Aggregate functions perform calculations based on a set of values and return a single value. Most aggregate functions ignore null values. Aggregate functions are often used with the `GROUP BY` clause of the select statement.

#### 12.1.3.1.5.5 Available Aggregate Functions

The following list shows the available aggregate functions:

- AVG
- CORR
- COUNT
- COVAR_POP
- COVAR_SAMP
- MAX
- MIN
- MODE
- PERCENTILE_CONT
- PERCENTILE_DISC
- STDDEV_POP
- STDDEV_SAMP
- SUM
- VAR_POP
- VAR_SAMP

#### 12.1.3.1.5.6 Window Functions

Window functions are functions applied over a subset (known as a window) of the rows returned by a select query and describes the following:

- lag
- lead
- row_number
- rank
- first_value
- last_value
- nth_value
- dense_rank
- percent_rank
- cume_dist
- ntile

For more information, see window_functions in the *SQL Syntax Features* section.

## 12.2 Catalog Reference Guide

The **Catalog Reference Guide** describes the following:

### 12.2.1 Overview

The SQream database uses a schema called `sqream_catalog` that contains information about your database's objects, such tables, columns, views, and permissions. Some additional catalog tables are used primarily for internal analysis and which may be different across SQream versions.

- *What Information Does the Schema Contain?*
- *Catalog Tables*
- *Additional Tables*
- *Examples*

### 12.2.2 What Information Does the Schema Contain?

The schema includes tables designated and relevant for both external and internal use:

- *External Tables*
- *Internal Tables*

#### 12.2.2.1 External Tables

The following table shows the data objects contained in the `sqream_catalog` schema designated for external use:

Table 3: Database Objects

| Database Object | Table |
|---|---|
| *Clustering Keys* | `clustering_keys` |
| *Columns* | `columns`, `external_table_columns` |
| *Databases* | `databases` |
| *Permissions* | `table_permissions`, `database_permissions`, `schema_permissions`, `permission_types`, `udf_permissions`, `sqream_catalog.table_default_permissions` |
| *Queries* | `saved_queries` |
| *Roles* | `roles`, `roles_memeberships` |
| *Schemas* | `schemas` |
| *Sequences* | `identity_key` |
| *Tables* | `tables`, `external_tables` |
| *Views* | `views` |
| *User Defined Functions* | `user_defined_functions` |

#### 12.2.2.2 Internal Tables

The following table shows the data objects contained in the sqream_catalog schema designated for internal use:

Table 4: Storage Objects

| Database Object | Table |
|---|---|
| Extents | Shows extents. |
| Chunk columns | Shows chunks_columns. |
| Chunks | Shows chunks. |
| Delete predicates | Shows delete_predicates. For more information, see *Deleting Data*. |

### 12.2.3 Catalog Tables

The sqream_catalog includes the following tables:

- *Clustering Keys*
- *Columns*
- *Databases*
- *Permissions*
- *Queries*
- *Roles*
- *Schemas*
- *Sequences*
- *Tables*
- *Views*
- *User Defined Functions*

#### 12.2.3.1 Clustering Keys

The `clustering_keys` data object is used for explicit clustering keys for tables. If you define more than one clustering key, each key is listed in a separate row, and is described in the following table:

| Column | Description |
| --- | --- |
| `database_name` | Shows the name of the database containing the table. |
| `table_id` | Shows the ID of the table containing the column. |
| `schema_name` | Shows the name of the schema containing the table. |
| `table_name` | Shows the name of the table containing the column. |
| `clustering_key` | Shows the name of the column used as a clustering key for this table. |

#### 12.2.3.2 Columns

The **Columns** database object shows the following tables:

- *Columns*
- *External Table Columns*

##### 12.2.3.2.1 Columns

The `column` data object is used with standard tables and is described in the following table:

| Column | Description |
|---|---|
| `database_name` | Shows the name of the database containing the table. |
| `schema_name` | Shows the name of the schema containing the table. |
| `table_id` | Shows the ID of the table containing the column. |
| `table_name` | Shows the name of the table containing the column. |
| `column_id` | Shows the ordinal number of the column in the table (begins at **0**). |
| `column_name` | Shows the column's name. |
| `type_name` | Shows the column's data type. For more information see *Supported Data Types*. |
| `column_size` | Shows the maximum length in bytes. |
| `has_default` | Shows `NULL` if the column has no default value, `1` if the default is a fixed value, or `2` if the default is an identity. For more information, see identity. |
| `default_value` | Shows the column's default value. For more information, see Default Value Constraints. |
| `compression_strategy` | Shows the compression strategy that a user has overridden. |
| `created` | Shows the timestamp displaying when the column was created. |
| `altered` | Shows the timestamp displaying when the column was last altered. |

#### 12.2.3.2.2 External Table Columns

The `external_table_columns` is used for viewing data from foreign tables.

For more information on foreign tables, see CREATE FOREIGN TABLE.

### 12.2.3.3 Databases

The `databases` data object is used for displaying database information, and is described in the following table:

| Column | Description |
|---|---|
| `database_id` | Shows the database's unique ID. |
| `database_name` | Shows the database's name. |
| `default_disk_chunk_size` | Reserved for internal use. |
| `default_process_chunk_size` | Reserved for internal use. |
| `rechunk_size` | Reserved for internal use. |
| `storage_subchunk_size` | Reserved for internal use. |
| `compression_chunk_size_threshold` | Reserved for internal use. |

#### 12.2.3.4 Permissions

The `permissions` data object is used for displaying permissions information, such as roles (also known as **grantees**), and is described in the following tables:

- *Permission Types*
- *Default Permissions*
- *Table Permissions*
- *Database Permissions*
- *Schema Permissions*
- *UDF Permissions*

##### 12.2.3.4.1 Permission Types

The `permission_types` object identifies the permission names existing in the database.

The following table describes the `permission_types` data object:

| Column | Description |
|---|---|
| `permission_type_id` | Shows the permission type's ID. |
| `name` | Shows the name of the permission type. |

##### 12.2.3.4.2 Default Permissions

The commands included in the **Default Permissions** section describe how to check the following default permissions:

- *Default Table Permissions*
- *Default Schema Permissions*

###### 12.2.3.4.2.1 Default Table Permissions

The `sqream_catalog.table_default_permissions` command shows the columns described below:

| Column | Description |
|---|---|
| `database_name` | Shows the database that the default permission rule applies to. |
| `schema_id` | Shows the schema that the rule applies to, or `NULL` if the `ALTER` statement does not specify a schema. |
| `modifier_role_id` | Shows the role to apply the rule to. |
| `getter_role_id` | Shows the role that the permission is granted to. |
| `permission_type` | Shows the type of permission granted. |

#### 12.2.3.4.2.2 Default Schema Permissions

The sqream_catalog.schema_default_permissions command shows the columns described below:

| Column | Description |
|---|---|
| database_name | Shows the database that the default permission rule applies to. |
| modifier_role_id | Shows the role to apply the rule to. |
| getter_role_id | Shows the role that the permission is granted to. |
| permission_type | Shows the type of permission granted. |

For an example of using the sqream_catalog.table_default_permissions command, see Granting Default Table Permissions.

#### 12.2.3.4.3 Table Permissions

The table_permissions data object identifies all permissions granted to tables. Each role-permission combination displays one row.

The following table describes the table_permissions data object:

| Column | Description |
|---|---|
| database_name | Shows the name of the database containing the table. |
| table_id | Shows the ID of the table the permission applies to. |
| role_id | Shows the ID of the role granted permissions. |
| permission_type | Identifies the permission type. |

#### 12.2.3.4.4 Database Permissions

The database_permissions data object identifies all permissions granted to databases. Each role-permission combination displays one row.

The following table describes the database_permissions data object:

| Column | Description |
|---|---|
| database_name | Shows the name of the database the permission applies to |
| role_id | Shows the ID of the role granted permissions. |
| permission_type | Identifies the permission type. |

##### 12.2.3.4.5 Schema Permissions

The schema_permissions data object identifies all permissions granted to schemas. Each role-permission combination displays one row.

The following table describes the schema_permissions data object:

| Column | Description |
|---|---|
| database_name | Shows the name of the database containing the schema. |
| schema_id | Shows the ID of the schema the permission applies to. |
| role_id | Shows the ID of the role granted permissions. |
| permission_type | Identifies the permission type. |

##### 12.2.3.4.6 UDF Permissions

**Comment** - *No content.*

#### 12.2.3.5 Queries

The savedqueries data object identifies the saved_queries in the database, as shown in the following table:

| Column | Description |
|---|---|
| name | Shows the saved query name. |
| num_parameters | Shows the number of parameters to be replaced at run-time. |

For more information, see saved_queries.

#### 12.2.3.6 Roles

The roles data object is used for displaying role information, and is described in the following tables:

- *Roles*
- *Role Memberships*

##### 12.2.3.6.1 Roles

The roles data object identifies the roles in the database, as shown in the following table:

| Column | Description |
|---|---|
| role_id | Shows the role's database-unique ID. |
| name | Shows the role's name. |
| superuser | Identifies whether the role is a superuser (1 - superuser, 0 - regular user). |
| login | Identifies whether the role can be used to log in to SQream (1 - yes, 0 - no). |
| has_password | Identifies whether the role has a password (1 - yes, 0 - no). |
| can_create_function | Identifies whether role can create UDFs (1 - yes, 0 - no). |

#### 12.2.3.6.2 Role Memberships

The roles_memberships data object identifies the role memberships in the database, as shown below:

| Column | Description |
|---|---|
| role_id | Shows the role ID. |
| member_role_id | Shows the ID of the parent role that this role inherits from. |
| inherit | Identifies whether permissions are inherited (1 - yes, 0 - no). |

### 12.2.3.7 Schemas

The schemas data object identifies all the database's schemas, as shown below:

| Column | Description |
|---|---|
| schema_id | Shows the schema's unique ID. |
| schema_name | Shows the schema's name. |
| schema_owner | Shows the name of the role that owns the schema. |
| rechunker_ignore | Reserved for internal use. |

### 12.2.3.8 Sequences

The sequences data object is used for displaying identity key information, as shown below:

#### 12.2.3.8.1 Identity Key

**Comment** - *No content.*

#### 12.2.3.9 Tables

The `tables` data object is used for displaying table information, and is described in the following tables:

- *Tables*
- *Foreign Tables*

##### 12.2.3.9.1 Tables

The `tables` data object identifies proper (**Comment** - *What does "proper" mean?*) SQream tables in the database, as shown in the following table:

| Column | Description |
|---|---|
| `database_name` | Shows the name of the database containing the table. |
| `table_id` | Shows the table's database-unique ID. |
| `schema_name` | Shows the name of the schema containing the table. |
| `table_name` | Shows the name of the table. |
| `row_count_valid` | Identifies whether the `row_count` can be used. |
| `row_count` | Shows the number of rows in the table. |
| `rechunker_ignore` | Relevant for internal use. |

##### 12.2.3.9.2 Foreign Tables

The `external_tables` data object identifies foreign tables in the database, as shown below:

| Column | Description |
|---|---|
| `database_name` | Shows the name of the database containing the table. |
| `table_id` | Shows the table's database-unique ID. |
| `schema_name` | Shows the name of the schema containing the table. |
| `table_name` | Shows the name of the table. |
| `format` | Identifies the foreign data wrapper used. `0` for `csv_fdw`, `1` for `parquet_fdw`, `2` for `orc_fdw`. |
| `created` | Identifies the clause used to create the table. |

#### 12.2.3.10 Views

The `views` data object is used for displaying views in the database, as shown below:

| Column | Description |
|---|---|
| `view_id` | Shows the view's database-unique ID. |
| `view_schema_name` | Shows the name of the schema containing the view. |
| `view_name` | Shows the name of the view. |
| `view_data` | Reserved for internal use. |
| `view_query_text` | Identifies the `AS` clause used to create the view. |

#### 12.2.3.11 User Defined Functions

The `udf` data object is used for displaying UDFs in the database, as shown below:

| Column | Description |
|---|---|
| `database_name` | Shows the name of the database containing the view. |
| `function_id` | Shows the UDF's database-unique ID. |
| `function_name` | Shows the name of the UDF. |

### 12.2.4 Additional Tables

The Reference Catalog includes additional tables that can be used for performance monitoring and inspection. The definition for these tables described on this page may change across SQream versions.



#### 12.2.4.1 Extents

The `extents` storage object identifies storage extents, and each storage extents can contain several chunks.

**Note:** This is an internal table designed for low-level performance troubleshooting.

| Column | Description |
|---|---|
| database_name | Shows the name of the databse containing the extent. |
| table_id | Shows the ID of the table containing the extent. |
| column_id | Shows the ID of the column containing the extent. |
| extent_id | Shows the ID for the extent. |
| size | Shows the extent size in megabytes. |
| path | Shows the full path to the extent on the file system. |

### 12.2.4.2 Chunk Columns

The `chunk_columns` storage object lists chunk information by column.

| Column | Description |
|---|---|
| database_name | Shows the name of the databse containing the extent. |
| table_id | Shows the ID of the table containing the extent. |
| column_id | Shows the ID of the column containing the extent. |
| chunk_id | Shows the chunk ID. |
| extent_id | Shows the extent ID. |
| compressed_size | Shows the compressed chunk size in bytes. |
| uncompressed_size | Shows the uncompressed chunk size in bytes. |
| compression_type | Shows the chunk's actual compression scheme. |
| long_min | Shows the minimum numeric value in the chunk (if one exists). |
| long_max | Shows the maximum numeric value in the chunk (if one exists). |
| string_min | Shows the minimum text value in the chunk (if one exists). |
| string_max | Shows the maximum text value in the chunk (if one exists). |
| offset_in_file | Reserved for internal use. |

**Note:** This is an internal table designed for low-level performance troubleshooting.

#### 12.2.4.3 Chunks

The `chunks` storage object identifies storage chunks.

| Column | Description |
|---|---|
| `database_name` | Shows the name of the databse containing the chunk. |
| `table_id` | Shows the ID of the table containing the chunk. |
| `column_id` | Shows the ID of the column containing the chunk. |
| `rows_num` | Shows the amount of rows in the chunk. |
| `deletion_status` | Determines what data to logically delete from the table first, and identifies how much data to delete from the chunk. The value `0` is ued for no data, `1` for some data, and `2` to delete the entire chunk. |

**Note:** This is an internal table designed for low-level performance troubleshooting.

#### 12.2.4.4 Delete Predicates

The `delete_predicates` storage object identifies the existing delete predicates that have not been cleaned up.

Each DELETE command may result in several entries in this table.

| Column | Description |
|---|---|
| `database_name` | Shows the name of the databse containing the predicate. |
| `table_id` | Shows the ID of the table containing the predicate. |
| `max_chunk_id` | Reserved for internal use, this is a placeholder marker for the highest `chunk_id` logged during the `DELETE` operation. |
| `delete_predicate` | Identifies the DELETE predicate. |

**Note:** This is an internal table designed for low-level performance troubleshooting.

### 12.2.5 Examples

The **Examples** page includes the following examples:

- *Listing All Tables in a Database*
- *Listing All Schemas in a Database*
- *Listing Columns and Their Types for a Specific Table*
- *Listing Delete Predicates*
- *Listing Saved Queries*

#### 12.2.5.1 Listing All Tables in a Database

```
master=> SELECT * FROM sqream_catalog.tables;
database_name | table_id | schema_name | table_name     | row_count_valid | row_count␣
↪| rechunker_ignore
--------------+----------+-------------+----------------+-----------------+-----------
↪+-----------------
master        |        1 | public      | nba            | true            |       457␣
↪|                0
master        |       12 | public      | cool_dates     | true            |         5␣
↪|                0
master        |       13 | public      | cool_numbers   | true            |         9␣
↪|                0
master        |       27 | public      | jabberwocky    | true            |         8␣
↪|                0
```

#### 12.2.5.2 Listing All Schemas in a Database

```
master=> SELECT * FROM sqream_catalog.schemas;
schema_id | schema_name   | schema_owner | rechunker_ignore
----------+---------------+--------------+-----------------
        0 | public        | sqream       | false
        1 | secret_schema | mjordan      | false
```

#### 12.2.5.3 Listing Columns and Their Types for a Specific Table

```
SELECT column_name, type_name
FROM sqream_catalog.columns
WHERE table_name='cool_animals';
```

#### 12.2.5.4 Listing Delete Predicates

```
SELECT  t.table_name, d.*  FROM
sqream_catalog.delete_predicates AS d
INNER JOIN sqream_catalog.tables AS t
ON d.table_id=t.table_id;
```

#### 12.2.5.5 Listing Saved Queries

```
SELECT * FROM sqream_catalog.savedqueries;
```

For more information, see saved_queries.

## 12.3 Command line programs

SQream contains several command line programs for using, starting, managing, and configuring SQream DB clusters.

This topic contains the reference for these programs, as well as flags and configuration settings.

Table 5: User CLIs

| Command | Usage |
|---|---|
| *sqream sql* | Built-in SQL client |

Table 6: SQream DB cluster components

| Command | Usage |
|---|---|
| *sqreamd* | Start a SQream DB worker |
| *metadata_server* | The cluster manager/coordinator that enables scaling SQream DB. |
| *server_picker* | Load balancer end-point |

Table 7: SQream DB utilities

| Command | Usage |
|---|---|
| *SqreamStorage* | Initialize a cluster and set superusers |
| *upgrade_storage* | Upgrade metadata schemas when upgrading between major versions |

Table 8: Docker utilities

| Command | Usage |
|---|---|
| *sqream_console* | Dockerized convenience wrapper for operations |
| *sqream_installer* | Dockerized installer |

### 12.3.1 metadata_server

SQream DB's cluster manager/coordinator is called `metadata_server`.

In general, you should not need to run `metadata_server` manually, but it is sometimes useful for testing.

This page serves as a reference for the options and parameters.

#### 12.3.1.1 Positional command line arguments

```
$ metadata_server [ <logging path> [ <listen port> ] ]
```

| Argument | Default | Description |
|---|---|---|
| Logging path | Current directory | Path to store metadata logs into |
| Listen port | `3105` | TCP listen port. If used, log path must be specified beforehand. |

#### 12.3.1.2 Starting metadata server

##### 12.3.1.2.1 Starting temporarily

```
$ nohup metadata_server &
$ MS_PID=$!
```

Using `nohup` and `&` sends metadata server to run in the background.

**Note:**

- Logs are saved to the current directory, under `metadata_server_logs`.
- The default listening port is 3105

##### 12.3.1.2.2 Starting temporarily with non-default port

To use a non-default port, specify the logging path as well.

```
$ nohup metadata_server /home/rhendricks/metadata_logs 9241 &
$ MS_PID=$!
```

Using `nohup` and `&` sends metadata server to run in the background.

**Note:**

- Logs are saved to the `/home/rhendricks/metadata_logs` directory.
- The listening port is 9241

##### 12.3.1.2.3 Stopping metadata server

To stop metadata server:

```
$ kill -9 $MS_PID
```

**Tip:** It is safe to stop any SQream DB component at any time using `kill`. No partial data or data corruption should occur when using this method to stop the process.

## 12.3.2 sqreamd

SQream DB's main worker is called `sqreamd`.

This page serves as a reference for the options and parameters.

### 12.3.2.1 Starting SQream DB

#### 12.3.2.1.1 Start SQream DB temporarily

In general, you should not need to run `sqreamd` manually, but it is sometimes useful for testing.

```
$ nohup sqreamd -config ~/.sqream/sqream_config.json &
$ SQREAM_PID=$!
```

Using `nohup` and `&` sends SQream DB to run in the background.

To stop the active worker:

```
$ kill -9 $SQREAM_PID
```

**Tip:** It is safe to stop SQream DB at any time using `kill`. No partial data or data corruption should occur when using this method to stop the process.

### 12.3.2.2 Command line arguments

`sqreamd` supports the following command line arguments:

| Argument | Default | Description |
|---|---|---|
| `--version` | None | Outputs the version of SQream DB and immediately exits. |
| `-config` | `$HOME/.sqream/sqream_config.json` | Specifies the configuration file to use |
| `--port_ssl` | Don't use SSL | When specified, tells SQream DB to listen for SSL connections |

#### 12.3.2.2.1 Positional command arguments

`sqreamd` also supports positional arguments, when not using a configuration file.

This method can be used to temporarily start a SQream DB worker for testing.

```
$ sqreamd <Storage path> <GPU ordinal> <TCP listen port (unsecured)> <License path>
```

| Argument | Required | Description |
|---|---|---|
| Storage path | ✓ | Full path to a valid SQream DB persistant storage |
| GPU Ordinal | ✓ | Number representing the GPU to use. Check GPU ordinals with *nvidia-smi -L* |
| TCP listen port (unsecured) | ✓ | TCP port SQream DB should listen on. Recommended: 5000 |
| License path | ✓ | Full path to a SQream DB license file |

### 12.3.3 sqream-console

`sqream-console` is an interactive shell designed to help manage a dockerized SQream DB installation.

The console itself is a dockerized application.

This page serves as a reference for the options and parameters.

**In this topic:**

#### 12.3.3.1 Starting the console

`sqream-console` can be found in your SQream DB installation, under the name `sqream-console`.

Start the console by executing it from the shell

```
$ ./sqream-console
.......................................................................................
↪..............................

.......................................................................................
↪..............................


Welcome to SQream Console ver 1.7.6, type exit to log-out

usage: sqream [-h] [--settings] {master,worker,client,editor} ...

Run SQream Cluster

optional arguments:
  -h, --help            show this help message and exit
  --settings            sqream environment variables settings

subcommands:
  sqream services
```

(continues on next page)

(continued from previous page)

```
  {master,worker,client,editor}
                        sub-command help
    master              start sqream master
    worker              start sqream worker
    client              operating sqream client
    editor              operating sqream statement editor
sqream-console>
```

The console is now waiting for commands.

The console is a wrapper around a standard linux shell. It supports commands like `ls`, `cp`, etc.

All SQream DB-specific commands start with the keyword `sqream`.

#### 12.3.3.2 Operations and flag reference

##### 12.3.3.2.1 Commands

| Command | Description |
|---|---|
| `sqream --help` | Shows the initial usage information |
| `sqream master` | Controls the master node's operations |
| `sqream worker` | Controls workers' operations |
| `sqream client` | Access to *sqream sql* |
| `sqream editor` | Controls the statement editor's operations (web UI) |

##### 12.3.3.2.2 Master

The master node contains the *metadata server* and the *load balancer*.

###### 12.3.3.2.2.1 Syntax

```
sqream master <flags>
```

| Flag/command | Description |
|---|---|
| `--start [ --single-host ]` | Starts the master node. The `--single-host` modifier sets the mode to allow all containers to run on the same server. |
| `--stop [ --all ]` | Stops the master node and all connected *workers*. The `--all` modifier instructs the `--stop` command to stop all running services related to SQream DB |
| `--list` | Shows a list of all active master nodes and their workers |
| `-p <port>` | Sets the port for the load balancer. Defaults to `3108` |
| `-m <port>` | Sets the port for the metadata server. Defaults to `3105` |

##### 12.3.3.2.2.2 Common usage

##### 12.3.3.2.2.3 Start master node

```
sqream-console> sqream master --start
starting master server in single_host mode ...
sqream_single_host_master is up and listening on ports:   3105,3108
```

##### 12.3.3.2.2.4 Start master node on different ports

```
sqream-console> sqream master --start -p 4105 -m 4108
starting master server in single_host mode ...
sqream_single_host_master is up and listening on ports:   4105,4108
```

##### 12.3.3.2.2.5 Listing active master nodes and workers

```
sqream-console> sqream master --list
container name: sqream_single_host_worker_1, container id: de9b8aff0a9c
container name: sqream_single_host_worker_0, container id: c919e8fb78c8
container name: sqream_single_host_master, container id: ea7eef80e038
```

##### 12.3.3.2.2.6 Stopping all SQream DB workers and master

```
sqream-console> sqream master --stop --all
  shutting down 2 sqream services ...
 sqream_editor    stopped
 sqream_single_host_worker_1    stopped
 sqream_single_host_worker_0    stopped
 sqream_single_host_master    stopped
```

#### 12.3.3.2.3 Workers

Workers are *SQream DB daemons*, that connect to the master node.

##### 12.3.3.2.3.1 Syntax

```
sqream worker <flags>
```

| Flag/command | Description |
|---|---|
| `--start [ options [ ...] ]` | Starts worker nodes. See options table below. |
| `--stop [ <worker name> \| --all ]` | Stops the specified worker name. The `--all` modifier instructs the `--stop` command to stop all running workers. |

Start options are specified consecutively, separated by spaces.

Table 9: Start options

| Option | Description |
|---|---|
| `<n>` | Specifies the number of workers to start |
| `-j <config file> [ ...]` | Specifies configuration files to apply to each worker. When launching multiple workers, specify one file per worker, separated by spaces. |
| `-p <port> [ ...]` | Sets the ports to listen on. When launching multiple workers, specify one port per worker, separated by spaces. Defaults to 5000 - 5000+n. |
| `-g <gpu id> [ ...]` | Sets the GPU ordinal to assign to each worker. When launching multiple workers, specify one GPU ordinal per worker, separated by spaces. Defaults to automatic allocation. |
| `-m <spool memory>` | Sets the spool memory per node in gigabytes. |
| `--master-host` | Sets the hostname for the master node. Defaults to `localhost`. |
| `--master-port` | Sets the port for the master node. Defaults to `3105`. |
| `--stand-alone` | For testing only: Starts a worker without connecting to the master node. |

#### 12.3.3.2.3.2 Common usage

#### 12.3.3.2.3.3 Start 2 workers

After starting the master node, start workers:

```
sqream-console> sqream worker --start 2
started sqream_single_host_worker_0 on port 5000, allocated gpu: 0
started sqream_single_host_worker_1 on port 5001, allocated gpu: 1
```

#### 12.3.3.2.3.4 Stop a single worker

To stop a single worker, find its name first:

```
sqream-console> sqream master --list
container name: sqream_single_host_worker_1, container id: de9b8aff0a9c
container name: sqream_single_host_worker_0, container id: c919e8fb78c8
container name: sqream_single_host_master, container id: ea7eef80e038
```

Then, issue a stop command:

```
sqream-console> sqream worker --stop sqream_single_host_worker_1
stopped sqream_single_host_worker_1
```

##### 12.3.3.2.3.5 Start workers with a different spool size

If no spool size is specified, the RAM is equally distributed among workers. Sometimes a system engineer may wish to specify the spool size manually.

This example starts two workers, with a spool size of 50GB per node:

```
sqream-console> sqream worker --start 2 -m 50
```

##### 12.3.3.2.3.6 Starting multiple workers on non-dedicated GPUs

By default, SQream DB workers assign one worker per GPU. However, a system engineer may wish to assign multiple workers per GPU, if the workload permits it.

This example starts 4 workers on 2 GPUs, with 50GB spool each:

```
sqream-console> sqream worker --start 2 -g 0 -m 50
started sqream_single_host_worker_0 on port 5000, allocated gpu: 0
started sqream_single_host_worker_1 on port 5001, allocated gpu: 0
sqream-console> sqream worker --start 2 -g 1 -m 50
started sqream_single_host_worker_2 on port 5002, allocated gpu: 1
started sqream_single_host_worker_3 on port 5003, allocated gpu: 1
```

##### 12.3.3.2.3.7 Overriding default configuration files

It is possible to override default configuration settings by listing a configuration file for every worker.

This example starts 2 workers on the same GPU, with modified configuration files:

```
sqream-console> sqream worker --start 2 -g 0 -j /etc/sqream/configfile.json /etc/
↪sqream/configfile2.json
```

#### 12.3.3.2.4 Client

The client operation runs *sqream sql* in interactive mode.

**Note:** The dockerized client is useful for testing and experimentation. It is not the recommended method for executing analytic queries. See more about connecting a third party tool to SQream DB for data analysis.

##### 12.3.3.2.4.1 Syntax

```
sqream client <flags>
```

| Flag/command | Description |
|---|---|
| `--master` | Connects to the master node via the load balancer |
| `--worker` | Connects to a worker directly |
| `--host <hostname>` | Specifies the hostname to connect to. Defaults to `localhost`. |
| `--port <port>`, `-p <port>` | Specifies the port to connect to. Defaults to `3108` when used with `-master`. |
| `--user <username>`, `-u <username>` | Specifies the role's username to use |
| `--password <password>`, `-w <password>` | Specifies the password to use for the role |
| `--database <database>`, `-d <database>` | Specifies the database name for the connection. Defaults to `master`. |

#### 12.3.3.2.4.2 Common usage

#### 12.3.3.2.4.3 Start a client

Connect to default `master` database through the load balancer:

```
sqream-console> sqream client --master -u sqream -w sqream
Interactive client mode
To quit, use ^D or \q.

master=> _
```

#### 12.3.3.2.4.4 Start a client to a specific worker

Connect to database `raviga` directly to a worker on port 5000:

```
sqream-console> sqream client --worker -u sqream -w sqream -p 5000 -d raviga
Interactive client mode
To quit, use ^D or \q.

raviga=> _
```

#### 12.3.3.2.4.5 Start master node on different ports

```
sqream-console> sqream master --start -p 4105 -m 4108
starting master server in single_host mode ...
sqream_single_host_master is up and listening on ports:   4105,4108
```

##### 12.3.3.2.4.6 Listing active master nodes and worker nodes

```
sqream-console> sqream master --list
container name: sqream_single_host_worker_1, container id: de9b8aff0a9c
container name: sqream_single_host_worker_0, container id: c919e8fb78c8
container name: sqream_single_host_master, container id: ea7eef80e038
```

#### 12.3.3.2.5 Editor

The editor operation runs the web UI for the SQream DB Statement Editor.

The editor can be used to run queries from a browser.

##### 12.3.3.2.5.1 Syntax

```
sqream editor <flags>
```

| Flag/command | Description |
|---|---|
| `--start` | Start the statement editor |
| `--stop` | Shut down the statement editor |
| `--port <port>`, `-p <port>` | Specify a different port for the editor. Defaults to `3000`. |

##### 12.3.3.2.5.2 Common usage

##### 12.3.3.2.5.3 Start the editor UI

```
sqream-console> sqream editor --start
access sqream statement editor through Chrome http://192.168.0.100:3000
```

##### 12.3.3.2.5.4 Stop the editor UI

```
sqream-console> sqream editor --stop
 sqream_editor    stopped
```

### 12.3.3.3 Using the console to start SQream DB

The console is used to start and stop SQream DB components in a dockerized environment.

#### 12.3.3.3.1 Starting a SQream DB cluster for the first time

To start a SQream DB cluster, start the master node, followed by workers.

The example below starts 2 workers, running on 2 dedicated GPUs.

```
sqream-console> sqream master --start
starting master server in single_host mode ...
sqream_single_host_master is up and listening on ports:   3105,3108

sqream-console> sqream worker --start 2
started sqream_single_host_worker_0 on port 5000, allocated gpu: 0
started sqream_single_host_worker_1 on port 5001, allocated gpu: 1

sqream-console> sqream editor --start
access sqream statement editor through Chrome http://192.168.0.100:3000
```

SQream DB is now listening on port 3108 for any incoming statements.

A user can also access the web editor (running on port `3000` on the SQream DB machine) to connect and run queries.

### 12.3.4 sqream-installer

`sqream-installer` is an application that prepares and configures a dockerized SQream DB installation.

This page serves as a reference for the options and parameters.

**In this topic:**

#### 12.3.4.1 Operations and flag reference

##### 12.3.4.1.1 Command line flags

| Flag | Description |
|---|---|
| `-i` | Loads the docker images for installation |
| `-k` | Load new licenses from the `license` subdirectory |
| `-K` | Validate licenses |
| `-f` | Force overwrite any existing installation **and data directories currently in use** |
| `-c <path to read configuration from>` | Specifies a path to read and store configuration files in. Defaults to `/etc/sqream`. |
| `-v <storage cluster path>` | Specifies a path to the storage cluster. The path is created if it does not exist. |
| `-l <startup log path>` | Specifies a path to store system startup logs. Defaults to `/var/log/sqream` |
| `-d <path>` | Specifies a path to expose to SQream DB workers. To expose several paths, repeat the usage of this flag. |
| `-s` | Shows system settings |
| `-r` | Reset the system configuration. This flag can't be combined with other flags. |

#### 12.3.4.2 Usage

##### 12.3.4.2.1 Install SQream DB for the first time

Assuming license package tarball has been placed in the `license` subfolder.

- The path where SQream DB will store data is `/home/rhendricks/sqream_storage`.
- Logs will be stored in /var/log/sqream
- Source CSV, Parquet, and ORC files can be accessed from `/home/rhendricks/source_data`. All other directory paths are hidden from the Docker container.

```
# ./sqream-install -i -k -v /home/rhendricks/sqream_storage -l /var/log/sqream -c /
↪etc/sqream -d /home/rhendricks/source_data
```

**Note:** Installation commands should be run with `sudo` or root access.

##### 12.3.4.2.2 Modify exposed directories

To expose more directory paths for SQream DB to read and write data from, re-run the installer with additional directory flags.

```
# ./sqream-install -d /home/rhendricks/more_source_data
```

There is no need to specify the initial installation flags - only the modified exposed directory paths flag.

#### 12.3.4.2.3 Install a new license package

Assuming license package tarball has been placed in the `license` subfolder.

```
# ./sqream-install -k
```

#### 12.3.4.2.4 View system settings

This information may be useful to identify problems accessing directory paths, or locating where data is stored.

```
# ./sqream-install -s
SQREAM_CONSOLE_TAG=1.7.4
SQREAM_TAG=2020.1
SQREAM_EDITOR_TAG=3.1.0
license_worker_0=[...]
license_worker_1=[...]
license_worker_2=[...]
license_worker_3=[...]
SQREAM_VOLUME=/home/rhendricks/sqream_storage
SQREAM_DATA_INGEST=/home/rhendricks/source_data
SQREAM_CONFIG_DIR=/etc/sqream/
LICENSE_VALID=true
SQREAM_LOG_DIR=/var/log/sqream/
SQREAM_USER=sqream
SQREAM_HOME=/home/sqream
SQREAM_ENV_PATH=/home/sqream/.sqream/env_file
PROCESSOR=x86_64
METADATA_PORT=3105
PICKER_PORT=3108
NUM_OF_GPUS=8
CUDA_VERSION=10.1
NVIDIA_SMI_PATH=/usr/bin/nvidia-smi
DOCKER_PATH=/usr/bin/docker
NVIDIA_DRIVER=418
SQREAM_MODE=single_host
```

#### 12.3.4.2.5 Upgrading to a new version of SQream DB

When upgrading to a new version with Docker, most settings don't need to be modified.

The upgrade process replaces the existing docker images with new ones.

1. Obtain the new tarball, and untar it to an accessible location. Enter the newly extracted directory.
2. Install the new images

   ```
   # ./sqream-install -i
   ```

3. The upgrade process will check for running SQream DB processes. If any are found running, the installer will ask to stop them in order to continue the upgrade process. Once all services are stopped, the new version will be loaded.
4. After the upgrade, open *sqream-console* and restart the desired services.

## 12.3.5 server_picker

SQream DB's load balancer is called `server_picker`.

This page serves as a reference for the options and parameters.

### 12.3.5.1 Positional command line arguments

```
$ server_picker [ <Metadata server address> <Metadata server port> [ <TCP listen port>
↪ [ <SSL listen port> ] ]
```

| Argument | Default | Description |
|---|---|---|
| `Metadata server address` | | IP or hostname to an active *metadata server* |
| `Metadata server port` | | TCP port to an active *metadata server* |
| `TCP listen port` | `3108` | TCP port for server picker to listen on |
| `Metadata server port` | `3109` | SSL port for server picker to listen on |

### 12.3.5.2 Starting server picker

#### 12.3.5.2.1 Starting temporarily

In general, you should not need to run `server_picker` manually, but it is sometimes useful for testing.

Assuming we have a *metadata server* listening on the localhost, on port 3105:

```
$ nohup server_picker 127.0.0.1 3105 &
$ SP_PID=$!
```

Using `nohup` and `&` sends server picker to run in the background.

#### 12.3.5.2.2 Starting temporarily with non-default port

Tell server picker to listen on port 2255 for unsecured connections, and port 2266 for SSL connections.

```
$ nohup server_picker 127.0.0.1 3105 2255 2266 &
$ SP_PID=$!
```

Using `nohup` and `&` sends server picker to run in the background.

#### 12.3.5.2.3 Stopping server picker

```
$ kill -9 $SP_PID
```

**Tip:** It is safe to stop any SQream DB component at any time using `kill`. No partial data or data corruption should occur when using this method to stop the process.

### 12.3.6 SqreamStorage

You can use the **SqreamStorage** program to create a new *storage cluster*.

The **SqreamStorage** page serves as a reference for the options and parameters.

#### 12.3.6.1 Running SqreamStorage

The **SqreamStorage** program is located in the **bin** directory of your SQream installation..

#### 12.3.6.2 Command Line Arguments

The **SqreamStorage** program supports the following command line arguments:

| Argument | Shorthand | Description |
|---|---|---|
| `--create-cluster` | `-C` | Creates a storage cluster at a specified path |
| `--cluster-root` | `-r` | Specifies the cluster path. The path must **not** already exist. |

#### 12.3.6.3 Example

The **Examples** section describes how to create a new storage cluster at `/home/rhendricks/raviga_database`:

```
$ SqreamStorage --create-cluster --cluster-root /home/rhendricks/raviga_database
Setting cluster version to: 26
```

Alternatively, you can write this in shorthand as `SqreamStorage -C -r /home/rhendricks/raviga_database`. A message is displayed confirming that your cluster has been created.

### 12.3.7 Sqream SQL CLI Reference

SQream DB comes with a built-in client for executing SQL statements either interactively or from the command-line.

This page serves as a reference for the options and parameters. Learn more about using SQream DB SQL with the CLI by visiting the first_steps tutorial.

**In this topic:**

#### 12.3.7.1 Installing Sqream SQL

If you have a SQream DB installation on your server, `sqream sql` can be found in the `bin` directory of your SQream DB installation, under the name `sqream`.

**Note:** If you installed SQream DB via Docker, the command is named `sqream-client sql`, and can be found in the same location as the console.

Changed in version 2020.1: As of version 2020.1, `ClientCmd` has been renamed to `sqream sql`.

To run `sqream sql` on any other Linux host:

1. Download the `sqream sql` tarball package from the *Client Drivers for 2022.1* page.
2. Untar the package: `tar xf sqream-sql-v2020.1.1_stable.x86_64.tar.gz`
3. Start the client:

```
$ cd sqream-sql-v2020.1.1_stable.x86_64
$ ./sqream sql --port=5000 --username=jdoe --databasename=master
Password:

Interactive client mode
To quit, use ^D or \q.

master=> _
```

#### 12.3.7.1.1 Troubleshooting Sqream SQL Installation

Upon running sqream sql for the first time, you may get an error `error while loading shared libraries: libtinfo.so.5: cannot open shared object file: No such file or directory`.

Solving this error requires installing the ncruses or libtinfo libraries, depending on your operating system.

- Ubuntu:
  1. Install `libtinfo`:

     ```
     $ sudo apt-get install -y libtinfo
     ```

  2. Depending on your Ubuntu version, you may need to create a symbolic link to the newer libtinfo that was installed.

     For example, if `libtinfo` was installed as `/lib/x86_64-linux-gnu/libtinfo.so.6.2`:

     ```
     $ sudo ln -s /lib/x86_64-linux-gnu/libtinfo.so.6.2 /lib/x86_64-linux-gnu/libtinfo.so.5
     ```

- CentOS / RHEL:
  1. Install `ncurses`:

     ```
     $ sudo yum install -y ncurses-libs
     ```

  2. Depending on your RHEL version, you may need to create a symbolic link to the newer libtinfo that was installed.

     For example, if `libtinfo` was installed as `/usr/lib64/libtinfo.so.6`:

     ```
     $ sudo ln -s /usr/lib64/libtinfo.so.6 /usr/lib64/libtinfo.so.5
     ```

### 12.3.7.2 Using Sqream SQL

By default, sqream sql runs in interactive mode. You can issue commands or SQL statements.

#### 12.3.7.2.1 Running Commands Interactively (SQL shell)

When starting sqream sql, after entering your password, you are presented with the SQL shell.

To exit the shell, type `\q` or `Ctrl-d`.

```
$ sqream sql --port=5000 --username=jdoe --databasename=master
Password:

Interactive client mode
To quit, use ^D or \q.

master=> _
```

The database name shown means you are now ready to run statements and queries.

Statements and queries are standard SQL, followed by a semicolon (`;`). Statement results are usually formatted as a valid CSV, followed by the number of rows and the elapsed time for that statement.

```
master=> SELECT TOP 5 * FROM nba;
Avery Bradley           ,Boston Celtics        ,0,PG,25,6-2 ,180,Texas             ␣
↪,7730337
Jae Crowder             ,Boston Celtics        ,99,SF,25,6-6 ,235,Marquette        ␣
↪ ,6796117
John Holland            ,Boston Celtics        ,30,SG,27,6-5 ,205,Boston University ␣
↪ ,\N
R.J. Hunter             ,Boston Celtics        ,28,SG,22,6-5 ,185,Georgia State    ␣
↪ ,1148640
Jonas Jerebko           ,Boston Celtics        ,8,PF,29,6-10,231,\N,5000000
5 rows
time: 0.001185s
```

**Note:** Null values are represented as \N.

When writing long statements and queries, it may be beneficial to use line-breaks. The prompt for a multi-line statement will change from => to `.`, to alert users to the change. The statement will not execute until a semicolon is used.

```
$ sqream sql --port=5000 --username=mjordan -d master
Password:

Interactive client mode
To quit, use ^D or \q.

master=> SELECT "Age",
. AVG("Salary")
. FROM NBA
. GROUP BY 1
. ORDER BY 2 ASC
. LIMIT 5
. ;
38,1840041
19,1930440
23,2034746
21,2067379
36,2238119
5 rows
time: 0.009320s
```

#### 12.3.7.2.2 Executing Batch Scripts (`-f`)

To run an SQL script, use the `-f <filename>` argument.

For example,

```
$ sqream sql --port=5000 --username=jdoe -d master -f sql_script.sql --results-only
```

**Tip:** Output can be saved to a file by using redirection (>).

#### 12.3.7.2.3 Executing Commands Immediately (-c)

To run a statement from the console, use the `-c <statement>` argument.

For example,

```
$ sqream sql --port=5000 --username=jdoe -d nba -c "SELECT TOP 5 * FROM nba"
Avery Bradley           ,Boston Celtics        ,0,PG,25,6-2 ,180,Texas                
↪,7730337
Jae Crowder             ,Boston Celtics        ,99,SF,25,6-6 ,235,Marquette           
↪ ,6796117
John Holland            ,Boston Celtics        ,30,SG,27,6-5 ,205,Boston University   
↪ ,\N
R.J. Hunter             ,Boston Celtics        ,28,SG,22,6-5 ,185,Georgia State       
↪ ,1148640
Jonas Jerebko           ,Boston Celtics        ,8,PF,29,6-10,231,\N,5000000
5 rows
time: 0.202618s
```

**Tip:** Remove the timing and row count by passing the `--results-only` parameter

### 12.3.7.3 Examples

#### 12.3.7.3.1 Starting a Regular Interactive Shell

Connect to local server 127.0.0.1 on port 5000, to the default built-in database, *master*:

```
$ sqream sql --port=5000 --username=mjordan -d master
Password:

Interactive client mode
To quit, use ^D or \q.

master=>_
```

Connect to local server 127.0.0.1 via the built-in load balancer on port 3108, to the default built-in database, *master*:

```
$ sqream sql --port=3105 --clustered --username=mjordan -d master
Password:

Interactive client mode
To quit, use ^D or \q.

master=>_
```

#### 12.3.7.3.2 Executing Statements in an Interactive Shell

Note that all SQL commands end with a semicolon.

Creating a new database and switching over to it without reconnecting:

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d master
Password:

Interactive client mode
To quit, use ^D or \q.

master=> create database farm;
executed
time: 0.003811s
master=> \c farm
farm=>
```

```
farm=> create table animals(id int not null, name text(30) not null, is_angry bool␣
↪not null);
executed
time: 0.011940s

farm=> insert into animals values(1,'goat',false);
executed
time: 0.000405s

farm=> insert into animals values(4,'bull',true) ;
executed
time: 0.049338s

farm=> select * from animals;
1,goat                          ,0
4,bull                          ,1
2 rows
time: 0.029299s
```

#### 12.3.7.3.3 Executing SQL Statements from the Command Line

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d farm -c "SELECT * FROM␣
↪animals WHERE is_angry = true"
4,bull                          ,1
1 row
time: 0.095941s
```

#### 12.3.7.3.4 Controlling the Client Output

Two parameters control the dispay of results from the client:

- `--results-only` - removes row counts and timing information
- `--delimiter` - changes the record delimiter

##### 12.3.7.3.4.1 Exporting SQL Query Results to CSV

Using the `--results-only` flag removes the row counts and timing.

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d farm -c "SELECT * FROM␣
↪animals" --results-only > file.csv
$ cat file.csv
1,goat                          ,0
2,sow                           ,0
3,chicken                       ,0
4,bull                          ,1
```

##### 12.3.7.3.4.2 Changing a CSV to a TSV

The `--delimiter` parameter accepts any printable character.

**Tip:** To insert a tab, use `Ctrl-V` followed by `Tab ↹` in Bash.

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d farm -c "SELECT * FROM␣
↪animals" --delimiter '   ' > file.tsv
$ cat file.tsv
1  goat                             0
2  sow                              0
3  chicken                          0
4  bull                             1
```

#### 12.3.7.3.5 Executing a Series of Statements From a File

Assuming a file containing SQL statements (separated by semicolons):

```
$ cat some_queries.sql
   CREATE TABLE calm_farm_animals
  ( id INT IDENTITY(0, 1), name TEXT(30)
  );

INSERT INTO calm_farm_animals (name)
  SELECT name FROM   animals WHERE  is_angry = false;
```

```
$ sqream sql --port=3105 --clustered --username=oldmcd -d farm -f some_queries.sql
executed
time: 0.018289s
executed
time: 0.090697s
```

##### 12.3.7.3.6 Connecting Using Environment Variables

You can save connection parameters as environment variables:

```
$ export SQREAM_USER=sqream;
$ export SQREAM_DATABASE=farm;
$ sqream sql --port=3105 --clustered --username=$SQREAM_USER -d $SQREAM_DATABASE
```

##### 12.3.7.3.7 Connecting to a Specific Queue

When using the *dynamic workload manager* - connect to `etl` queue instead of using the default `sqream` queue.

```
$ sqream sql --port=3105 --clustered --username=mjordan -d master --service=etl
Password:

Interactive client mode
To quit, use ^D or \q.

master=>_
```

#### 12.3.7.4 Operations and Flag References

##### 12.3.7.4.1 Command Line Arguments

**Sqream SQL** supports the following command line arguments:

| Argument | Default | Description |
|---|---|---|
| `-c` or `--command` | None | Changes the mode of operation to single-command, non-interactive. Use this argument to run a statement and immediately exit. |
| `-f` or `--file` | None | Changes the mode of operation to multi-command, non-interactive. Use this argument to run a sequence of statements from an external file and immediately exit. |
| `--host` | `127.0.0.1` | Address of the SQream DB worker. |
| `--port` | `5000` | Sets the connection port. |
| `--databasename` or `-d` | None | Specifies the database name for queries and statements in this session. |
| `--username` | None | Username to connect to the specified database. |
| `--password` | None | Specify the password using the command line argument. If not specified, the client will prompt the user for the password. |
| `--clustered` | False | When used, the client connects to the load balancer, usually on port `3108`. If not set, the client assumes the connection is to a standalone SQream DB worker. |
| `--service` | `sqream` | *Service name (queue)* that statements will file into. |
| `--results-only` | False | Outputs results only, without timing information and row counts |
| `--no-history` | False | When set, prevents command history from being saved in `~/.sqream/clientcmdhist` |
| `--delimiter` | `,` | Specifies the field separator. By default, `sqream sql` outputs valid CSVs. Change the delimiter to modify the output to another delimited format (e.g. TSV, PSV). See the section *supported record delimiters* below for more information. |

**Tip:** Run `$ sqream sql --help` to see a full list of arguments

#### 12.3.7.4.1.1 Supported Record Delimiters

The supported record delimiters are printable ASCII values (32-126).

- Recommended delimiters for use are: `,`, `|`, tab character.
- The following characters are **not supported**: `\`, `N`, `-`, `:`, `"`, `\n`, `\r`, `.`, lower-case latin letters, digits (0-9)

#### 12.3.7.4.2 Meta-Commands

- Meta-commands in Sqream SQL start with a backslash (\)

**Note:** Meta commands do not end with a semicolon

| Command | Example | Description |
|---|---|---|
| `\q` or `\quit` | `master=> \q` | Quit the client. (Same as `Ctrl-d`) |
| `\c <database>` or `\connect <database>` | `master=> \c fox`<br>`fox=>` | Changes the current connection to an alternate database |

#### 12.3.7.4.3 Basic Commands

#### 12.3.7.4.4 Moving Around the Command Line

| Command | Description |
|---|---|
| `Ctrl-a` | Goes to the beginning of the command line. |
| `Ctrl-e` | Goes to the end of the command line. |
| `Ctrl-u` | Deletes from cursor to the beginning of the command line. |
| `Ctrl-k` | Deletes from the cursor to the end of the command line. |
| `Ctrl-w` | Delete from cursor to beginning of a word. |
| `Ctrl-y` | Pastes a word or text that was cut using one of the deletion shortcuts (such as the one above) after the cursor. |
| `Alt-b` | Moves back one word (or goes to the beginning of the word where the cursor is). |
| `Alt-f` | Moves forward one word (or goes to the end of word the cursor is). |
| `Alt-d` | Deletes to the end of a word starting at the cursor. Deletes the whole word if the cursor is at the beginning of that word. |
| `Alt-c` | Capitalizes letters in a word starting at the cursor. Capitalizes the whole word if the cursor is at the beginning of that word. |
| `Alt-u` | Capitalizes from the cursor to the end of the word. |
| `Alt-l` | Makes lowercase from the cursor to the end of the word. |
| `Ctrl-f` | Moves forward one character. |
| `Ctrl-b` | Moves backward one character. |
| `Ctrl-h` | Deletes characters located before the cursor. |
| `Ctrl-t` | Swaps a character at the cursor with the previous character. |

#### 12.3.7.4.5 Searching

| Command | Description |
| --- | --- |
| `Ctrl-r` | Searches the history backward. |
| `Ctrl-g` | Escapes from history-searching mode. |
| `Ctrl-p` | Searches the previous command in history. |
| `Ctrl-n` | Searches the next command in history. |

## 12.3.8 upgrade_storage

`upgrade_storage` is used to upgrade metadata schemas, when upgrading between major versions.

This page serves as a reference for the options and parameters.

### 12.3.8.1 Running upgrade_storage

`upgrade_storage` can be found in the `bin` directory of your SQream DB installation.

### 12.3.8.2 Command line arguments

`upgrade_storage` contains one positional argument:

```
$ upgrade_storage <storage path>
```

| Argument | Required | Description |
| --- | --- | --- |
| Storage path | ✓ | Full path to a valid storage cluster |

### 12.3.8.3 Results and error codes

| Result | Message | Description |
| --- | --- | --- |
| Success | `storage has been upgraded successfully to version 26` | Storage has been successfully upgraded |
| Success | `no need to upgrade` | Storage doesn't need an upgrade |
| Failure: can't read storage | `levelDB is in use by another application` | Check permissions, and ensure no SQream DB workers or *metadata_server* are running when performing this operation. |

### 12.3.8.4 Examples

#### 12.3.8.4.1 Upgrade SQream DB's storage cluster

```
$ ./upgrade_storage /home/rhendricks/raviga_database
get_leveldb_version path{/home/rhendricks/raviga_database}
current storage version 23
upgrade_v24
upgrade_storage to 24
```

(continues on next page)

(continued from previous page)

```
upgrade_storage to 24 - Done
upgrade_v25
upgrade_storage to 25
upgrade_storage to 25 - Done
upgrade_v26
upgrade_storage to 26
upgrade_storage to 26 - Done
validate_leveldb
storage has been upgraded successfully to version 26
```

This message confirms that the cluster has already been upgraded correctly.

## 12.4 SQL Feature Checklist

To understand which ANSI SQL and other SQL features SQream DB supports, use the tables below.

**In this topic:**

### 12.4.1 Data Types and Values

Read more about Yes data types.

Table 10: Data Types and Values

| Item | Supported | Further information |
|---|---|---|
| `BOOL` | Yes | Boolean values |
| `TINTINT` | Yes | Unsigned 1 byte integer (0 - 255) |
| `SMALLINT` | Yes | 2 byte integer (-32,768 - 32,767) |
| `INT` | Yes | 4 byte integer (-2,147,483,648 - 2,147,483,647) |
| `BIGINT` | Yes | 8 byte integer (-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807) |
| `REAL` | Yes | 4 byte floating point |
| `DOUBLE`, `FLOAT` | Yes | 8 byte floating point |
| `DECIMAL`, `NUMERIC` | Yes | Fixed-point numbers. |
| `TEXT` | Yes | Variable length string - UTF-8 encoded |
| `DATE` | Yes | Date |
| `DATETIME`, `TIMESTAMP` | Yes | Date and time |
| `NULL` | Yes | `NULL` values |
| `TIME` | No | Can be stored as a text string or as part of a `DATETIME` |

### 12.4.2 Contraints

Table 11: Contraints

| Item | Supported | Further information |
|---|---|---|
| `Not null` | Yes | `NOT NULL` |
| `Default values` | Yes | `DEFAULT` |
| `AUTO INCREMENT` | Yes (different name) | `IDENTITY` |

### 12.4.3 Transactions

SQream DB treats each statement as an auto-commit transaction. Each transaction is isolated from other transactions with serializable isolation.

If a statement fails, the entire transaction is cancelled and rolled back. The database is unchanged.

Read more about *transactions in SQream DB*.

### 12.4.4 Indexes

SQream DB has a range-index collected on all columns as part of the metadata collection process.

SQream DB does not support explicit indexing, but does support clustering keys.

Read more about clustering keys and our metadata system.

### 12.4.5 Schema Changes

Table 12: Schema Changes

| Item | Supported | Further information |
|---|---|---|
| `ALTER TABLE` | Yes | alter_table - Add column, alter column, drop column, rename column, rename table, modify clustering keys |
| Rename database | No | |
| Rename table | Yes | rename_table |
| Rename column | Yes | rename_column |
| Add column | Yes | add_column |
| Remove column | Yes | drop_column |
| Alter column data type | No | |
| Add / modify clustering keys | Yes | cluster_by |
| Drop clustering keys | Yes | drop_clustering_key |
| Add / Remove constraints | No | |
| Rename schema | No | |
| Drop schema | Yes | drop_schema |
| Alter default schema per user | Yes | alter_default_schema |

### 12.4.6 Statements

Table 13: Statements

| Item | Supported | Further information |
|---|---|---|
| SELECT | Yes | select |
| CREATE TABLE | Yes | create_table |
| CREATE FOREIGN / EXTERNAL TABLE | Yes | create_foreign_table |
| DELETE | Yes | *Deleting Data* |
| INSERT | Yes | insert, copy_from |
| TRUNCATE | Yes | truncate |
| UPDATE | No | |
| VALUES | Yes | values |

### 12.4.7 Clauses

Table 14: Clauses

| Item | Supported | Further information |
|---|---|---|
| LIMIT / TOP | Yes | |
| LIMIT with OFFSET | No | |
| WHERE | Yes | |
| HAVING | Yes | |
| OVER | Yes | |

### 12.4.8 Table Expressions

Table 15: Table Expressions

| Item | Supported | Further information |
|---|---|---|
| Tables, Views | Yes | |
| Aliases, AS | Yes | |
| JOIN - INNER, LEFT [ OUTER ], RIGHT [ OUTER ], CROSS | Yes | |
| Table expression subqueries | Yes | |
| Scalar subqueries | No | |

### 12.4.9 Scalar Expressions

Read more about scalar_expressions.

Table 16: Scalar Expressions

| Item | Supported | Further information |
|---|---|---|
| Common functions | Yes | CURRENT_TIMESTAMP, SUBSTRING, TRIM, EXTRACT, etc. |
| Comparison operators | Yes | <, <=, >, >=, =, <>, !=, IS, IS NOT |
| Boolean operators | Yes | AND, NOT, OR |
| Conditional expressions | Yes | CASE .. WHEN |
| Conditional functions | Yes | COALESCE |
| Pattern matching | Yes | LIKE, RLIKE, ISPREFIXOF, CHARINDEX, PATINDEX |
| REGEX POSIX pattern matching | Yes | RLIKE, REGEXP_COUNT, REGEXP_INSTR, REGEXP_SUBSTR, |
| EXISTS | No | |
| IN, NOT IN | Partial | Literal values only |
| Bitwise arithmetic | Yes | &, \|, XOR, ~, >>, << |

### 12.4.10 Permissions

Read more about *Access Control* in SQream DB.

Table 17: Permissions

| Item | Supported | Further information |
|---|---|---|
| Roles as users and groups | Yes | |
| Object default permissions | Yes | |
| Column / Row based permissions | No | |
| Object ownership | No | |

### 12.4.11 Extra Functionality

Table 18: Extra Functionality

| Item | Supported | Further information |
|---|---|---|
| Information schema | Yes | *Catalog Reference Guide* |
| Views | Yes | create_view |
| Window functions | Yes | window_functions |
| CTEs | Yes | common_table_expressions |
| Saved queries, Saved queries with parameters | Yes | saved_queries |
| Sequences | Yes | identity |

CHAPTER

# THIRTEEN

# DATA TYPE GUIDES

This section describes the following:

## 13.1 Converting and Casting Types

SQream supports explicit and implicit casting and type conversion. The system may automatically add implicit casts when combining different data types in the same expression. In many cases, while the details related to this are not important, they can affect the query results of a query. When necessary, an explicit cast can be used to override the automatic cast added by SQream DB.

For example, the ANSI standard defines a `SUM()` aggregation over an `INT` column as an `INT`. However, when dealing with large amounts of data this could cause an overflow.

You can rectify this by casting the value to a larger data type, as shown below:

```
SUM(some_int_column :: BIGINT)
```

SQream supports the following three data conversion types:

- `CAST(<value> TO <data type>)`, to convert a value from one type to another. For example, `CAST('1997-01-01' TO DATE)`, `CAST(3.45 TO SMALLINT)`, `CAST(some_column TO TEXT(30))`.
- `<value> :: <data type>`, a shorthand for the `CAST` syntax. For example, `'1997-01-01' :: DATE`, `3.45 :: SMALLINT`, `(3+5) :: BIGINT`.
- See the *SQL functions reference* for additional functions that convert from a specific value which is not an SQL type, such as from_unixts, etc.

**Note:** SQream interprets integer constants exceeding the maximum bigint value as float constants, which may cause precision loss.

## 13.2 Supported Data Types

The **Supported Data Types** page describes SQream's supported data types:

The following table shows the supported data types.

| Name | Description | Data Size (Not Null, Uncompressed) | Example | Alias |
|---|---|---|---|---|
| `BOOL` | Boolean values (`true`, `false`) | 1 byte | `true` | `BIT` |
| `TINYINT` | Unsigned integer (0 - 255) | 1 byte | `5` | NA |
| `SMALLINT` | Integer (-32,768 - 32,767) | 2 bytes | `-155` | NA |
| `INT` | Integer (-2,147,483,648 - 2,147,483,647) | 4 bytes | `1648813` | `INTEGER` |
| `BIGINT` | Integer (-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807) | 8 bytes | `36124441255243` | `NUMBER` |
| `REAL` | Floating point (inexact) | 4 bytes | `3.141` | NA |
| `DOUBLE` | Floating point (inexact) | 8 bytes | `0.000003` | `FLOAT`/`DOUBLE PRECISION` |
| `TEXT (n)` | Variable length string - UTF-8 unicode | Up to `4` bytes | `'Kiwis have tiny wings, but cannot fly.'` | `CHAR VARYING`, `CHAR`, `CHARACTER VARYING`, `CHARACTER`, `NATIONAL CHARACTER VARYING`, `NATIONAL CHARACTER`, `NCHAR VARYING`, `NCHAR` |
| `NUMERIC` | 38 digits | 16 bytes | `0.123245678901234567890123456789012345678` | `DECIMAL` |
| `DATE` | Date | 4 bytes | `'1955-11-05'` | NA |
| `DATETIME` | Date and time pairing in UTC | 8 bytes | `'1955-11-05 01:24:00.000'` | `TIMESTAMP`, `DATETIME2` |

**Note:** SQream compresses all columns and types. The data size noted is the maximum data size allocation for uncompressed data.

## 13.3 Supported Casts

The **Supported Casts** section describes supported casts for the following types:

### 13.3.1 Numeric

The **Numeric** data type (also known as **Decimal**) is recommended for values that tend to occur as exact decimals, such as in Finance. While Numeric has a fixed precision of `38`, higher than `REAL` (`9`) or `DOUBLE` (`17`), it runs calculations more slowly. For operations that require faster performance, using Floating Point is recommended.

The correct syntax for Numeric is `numeric(p, s)`, where `p` is the total number of digits (`38` maximum), and `s` is the total number of decimal digits.

#### 13.3.1.1 Numeric Examples

The following is an example of the Numeric syntax:

```
$ create or replace table t(x numeric(20, 10), y numeric(38, 38));
$ insert into t values(1234567890.1234567890, 0.
↪12324567890123456789012345678901234567 8);
$ select x + y from t;
```

The following table shows information relevant to the Numeric data type:

| Description | Data Size (Not Null, Uncompressed) | Example |
|---|---|---|
| 38 digits | 16 bytes | `0.12324567890123456789012345678901234` |

Numeric supports the following operations:

- All join types.
- All aggregation types (not including Window functions).
- Scalar functions (not including some trigonometric and logarithmic functions).

### 13.3.2 Boolean

The following table describes the Boolean data type.

| Values | Syntax | Data Size (Not Null, Uncompressed) |
|---|---|---|
| `true`, `false` (case sensitive) | When loading from CSV, `BOOL` columns can accept `0` as `false` and `1` as `true`. | 1 byte, but resulting average data sizes may be lower after compression. |

#### 13.3.2.1 Boolean Examples

The following is an example of the Boolean syntax:

```
CREATE TABLE animals (name TEXT, is_angry BOOL);

INSERT INTO animals VALUES ('fox',true), ('cat',true), ('kiwi',false);

SELECT name, CASE WHEN is_angry THEN 'Is really angry!' else 'Is not angry' END FROM␣
↪animals;
```

The following is an example of the correct output:

```
"fox","Is really angry!"
"cat","Is really angry!"
"kiwi","Is not angry"
```

#### 13.3.2.2 Boolean Casts and Conversions

The following table shows the possible Boolean value conversions:

| Type | Details |
|---|---|
| `TINYINT`, `SMALLINT`, `INT`, `BIGINT` | `true` → `1`, `false` → `0` |
| `REAL`, `DOUBLE` | `true` → `1.0`, `false` → `0.0` |

### 13.3.3 Integer

Integer data types are designed to store whole numbers.

For more information about identity sequences (sometimes called auto-increment or auto-numbers), see identity.

#### 13.3.3.1 Integer Types

The following table describes the Integer types.

| Name | Details | Data Size (Not Null, Uncompressed) | Example |
|---|---|---|---|
| `TINYINT` | Unsigned integer (0 - 255) | 1 byte | `5` |
| `SMALLINT` | Integer (-32,768 - 32,767) | 2 bytes | `-155` |
| `INT` | Integer (-2,147,483,648 - 2,147,483,647) | 4 bytes | `1648813` |
| `BIGINT` | Integer (-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807) | 8 bytes | `36124441255243` |

The following table describes the Integer data type.

| Syntax | Data Size (Not Null, Uncompressed) |
|---|---|
| An integer can be entered as a regular literal, such as `12`, `-365`. | Integer types range between 1, 2, 4, and 8 bytes - but resulting average data sizes could be lower after compression. |

#### 13.3.3.2 Integer Examples

The following is an example of the Integer syntax:

```
CREATE TABLE cool_numbers (a INT NOT NULL, b TINYINT, c SMALLINT, d BIGINT);

INSERT INTO cool_numbers VALUES (1,2,3,4), (-5, 127, 32000, 45000000000);

SELECT * FROM cool_numbers;
```

The following is an example of the correct output:

```
1,2,3,4
-5,127,32000,45000000000
```

#### 13.3.3.3 Integer Casts and Conversions

The following table shows the possible Integer value conversions:

| Type | Details |
|---|---|
| `REAL`, `DOUBLE` | `1` → `1.0`, `-32` → `-32.0` |
| `TEXT` (All numberic values must fit in the string length) | `1` → `'1'`, `2451` → `'2451'` |

### 13.3.4 Floating Point

The **Floating Point** data types (`REAL` and `DOUBLE`) store extremely close value approximations, and are therefore recommended for values that tend to be inexact, such as Scientific Notation. While Floating Point generally runs faster than Numeric, it has a lower precision of `9` (`REAL`) or `17` (`DOUBLE`) compared to Numeric's `38`. For operations that require a higher level of precision, using Numeric is recommended.

The floating point representation is based on IEEE 754.

#### 13.3.4.1 Floating Point Types

The following table describes the Floating Point data types.

| Name | Details | Data Size (Not Null, Uncompressed) | Example |
|---|---|---|---|
| `REAL` | Single precision floating point (inexact) | 4 bytes | `3.141` |
| `DOUBLE` | Double precision floating point (inexact) | 8 bytes | `0.000003` |

The following table shows information relevant to the Floating Point data types.

| Aliases | Syntax | Data Size (Not Null, Uncompressed) |
|---|---|---|
| `DOUBLE` is also known as `FLOAT`. | A double precision floating point can be entered as a regular literal, such as `3.14`, `2.718`, `.34`, or `2.71e-45`. To enter a `REAL` floating point number, cast the value. For example, `(3.14 :: REAL)`. | Floating point types are either 4 or 8 bytes, but size could be lower after compression. |

#### 13.3.4.2 Floating Point Examples

The following are examples of the Floating Point syntax:

```
CREATE TABLE cool_numbers (a REAL NOT NULL, b DOUBLE);

INSERT INTO cool_numbers VALUES (1,2), (3.14159265358979, 2.718281828459);

SELECT * FROM cool_numbers;
```

```
1.0,2.0
3.1415927,2.718281828459
```

**Note:** Most SQL clients control display precision of floating point numbers, and values may appear differently in some clients.

#### 13.3.4.3 Floating Point Casts and Conversions

The following table shows the possible Floating Point value conversions:

| Type | Details |
|---|---|
| `BOOL` | `1.0` → `true`, `0.0` → `false` |
| `TINYINT`, `SMALLINT`, `INT`, `BIGINT` | `2.0` → `2`, `3.14159265358979` → `3`, `2.718281828459` → `2`, `0.5` → `0`, `1.5` → `1` |

**Note:** As shown in the above examples, casting `real` to `int` rounds down.

### 13.3.5 String

`TEXT` is designed for storing text or strings of characters.

SQream UTF-8 representations (`TEXT`).

#### 13.3.5.1 Length

When using `TEXT`, specifying a size is optional. If not specified, the text field carries no constraints. To limit the size of the input, use `TEXT(n)`, where `n` is the permitted number of characters.

The following apply to setting the String type length:

- If the data exceeds the column length limit on `INSERT` or `COPY` operations, SQream DB will return an error.
- When casting or converting, the string has to fit in the target. For example, `'Kiwis are weird birds' :: TEXT(5)` will return an error. Use `SUBSTRING` to truncate the length of the string.

#### 13.3.5.2 Syntax

String types can be written with standard SQL string literals, which are enclosed with single quotes, such as `'Kiwi bird'`. To include a single quote in the string, use double quotations, such as `'Kiwi bird''s wings are tiny'`. String literals can also be dollar-quoted with the dollar sign `$`, such as `$$Kiwi bird's wings are tiny$$` is the same as `'Kiwi bird''s wings are tiny'`.

#### 13.3.5.3 Size

`TEXT(n)` can occupy up to *4*n* bytes. However, the size of strings is variable and is compressed by SQream.

#### 13.3.5.4 String Examples

The following is an example of the String syntax:

```
CREATE TABLE cool_strings (a TEXT NOT NULL, b TEXT);

INSERT INTO cool_strings VALUES ('hello world', 'Hello to kiwi birds specifically');

INSERT INTO cool_strings VALUES ('This is ASCII only', 'But this column can contain □□□□');

SELECT * FROM cool_strings;
```

The following is an example of the correct output:

```
hello world  ,Hello to kiwi birds specifically
This is ASCII only,But this column can contain □□□□
```

**Note:** Most clients control the display precision of floating point numbers, and values may appear differently in some clients.

#### 13.3.5.5 String Casts and Conversions

The following table shows the possible String value conversions:

| Type | Details |
|---|---|
| `BOOL` | `'true'` → `true`, `'false'` → `false` |
| `TINYINT`, `SMALLINT`, `INT`, `BIGINT` | `'2'` → `2`, `'-128'` → `-128` |
| `REAL`, `DOUBLE` | `'2.0'` → `2.0`, `'3.141592'` → `3.141592` |
| `DATE`, `DATETIME` | Requires a supported format, such as `'1955-11-05` → `date '1955-11-05'`, `'1955-11-05 01:24:00.000'` → `'1955-11-05 01:24:00.000'` |

### 13.3.6 Date

`DATE` is a type designed for storing year, month, and day. `DATETIME` is a type designed for storing year, month, day, hour, minute, seconds, and milliseconds in UTC with 1 millisecond precision.

#### 13.3.6.1 Date Types

The following table describes the Date types:

Table 1: Date Types

| Name | Details | Data Size (Not Null, Uncompressed) | Example |
|---|---|---|---|
| `DATE` | Date | 4 bytes | `'1955-11-05'` |
| `DATETIME` | Date and time pairing in UTC | 8 bytes | `'1955-11-05 01:24:00.000'` |

#### 13.3.6.2 Aliases

`DATETIME` is also known as `TIMESTAMP` or `DATETIME2`.

#### 13.3.6.3 Syntax

`DATE` values are formatted as string literals.

The following is an example of the DATETIME syntax:

```
'1955-11-05'
```

```
date '1955-11-05'
```

`DATETIME` values are formatted as string literals conforming to ISO 8601.

The following is an example of the DATETIME syntax:

```
'1955-11-05 01:26:00'
```

SQream attempts to guess if the string literal is a date or datetime based on context, for example when used in date-specific functions.

#### 13.3.6.4 Size

A `DATE` column is 4 bytes in length, while a `DATETIME` column is 8 bytes in length.

However, the size of these values is compressed by SQream DB.

#### 13.3.6.5 Date Examples

The following is an example of the Date syntax:

```
CREATE TABLE important_dates (a DATE, b DATETIME);

INSERT INTO important_dates VALUES ('1997-01-01', '1955-11-05 01:24');

SELECT * FROM important_dates;
```

The following is an example of the correct output:

```
1997-01-01,1955-11-05 01:24:00.0
```

The following is an example of the Datetime syntax:

```
SELECT a :: DATETIME, b :: DATE FROM important_dates;
```

The following is an example of the correct output:

```
1997-01-01 00:00:00.0,1955-11-05
```

**Warning:** Some client applications may alter the `DATETIME` value by modifying the timezone.

#### 13.3.6.6 Date Casts and Conversions

The following table shows the possible `DATE` and `DATETIME` value conversions:

| Type | Details |
|---|---|
| `TEXT` | `'1997-01-01'` → `'1997-01-01'`, `'1955-11-05 01:24'` → `'1955-11-05 01:24:00.000'` |

CHAPTER

# FOURTEEN

# RELEASE NOTES

| Version | Release Date |
|---|---|
| *Release Notes 2022.1* | July 19, 2022 |
| *Release Notes 2021.2* | September 13, 2021 |
| *Release Notes 2021.1* | June 13, 2021 |
| *Release Notes 2020.3* | October 8, 2020 |
| *Release Notes 2020.2* | July 22, 2020 |
| *Release Notes 2020.1* | January 15, 2020 |

## 14.1 Release Notes 2022.1

The 2022.1 Release Notes describe the following releases:

### 14.1.1 Release Notes 2022.1.2

The 2022.1.2 release notes were released on 8/24/2022 and describe the following:

- *Version Content*
- *New Features*
- *Resolved Issues*
- *Operations and Configuration Changes*
- *Naming Changes*
- *Deprecated Features*
- *End of Support*
- *Upgrading to v2022.1.2*

#### 14.1.1.1 Version Content

The 2022.1.2 Release Notes describes the following:

- Automatic schema identification.

- Optimized queries on external Parquet tables.

#### 14.1.1.2 New Features

The 2022.1.2 Release Notes include the following new features:

- *Parquet Read Optimization*

##### 14.1.1.2.1 Parquet Read Optimization

Querying Parquet foreign tables has been optimized and is now up to 20x faster than in previous versions.

#### 14.1.1.3 Resolved Issues

The following table lists the issues that were resolved in Version 2022.1.2:

| SQ No. | Description |
|---|---|
| SQ-10892 | An incorrect error message was displayed when users ran the `UPDATE` command on foreign tables. |
| SQ-11273 | Clustering optimization only occurs when copying data from CSV files. |

#### 14.1.1.4 Operations and Configuration Changes

No configuration changes were made.

#### 14.1.1.5 Naming Changes

No relevant naming changes were made.

#### 14.1.1.6 Deprecated Features

No features were deprecated for Version 2022.1.2.

#### 14.1.1.7 End of Support

The End of Support section is not relevant to Version 2022.1.2.

#### 14.1.1.8 Upgrading to v2022.1.2

1. Generate a back-up of the metadata by running the following command:

```
$ select backup_metadata('out_path');
```

**Tip:** SQream recommends storing the generated back-up locally in case needed.

SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

```
$ ./upgrade_storage <levelDB path>
```

**Note:** Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

### 14.1.2 Release Notes 2022.1.1

The 2022.1.1 release notes were released on 7/19/2022 and describe the following:

#### 14.1.2.1 Version Content

The 2022.1.1 Release Notes describes the following:

- Enhanced security features
- For more information, see SQream Acceleration Studio 5.4.7.

#### 14.1.2.2 Storage Version

The storage version presently in effect is version 40.

#### 14.1.2.3 New Features

The 2022.1.1 Release Notes include the following new features:



##### 14.1.2.3.1 Password Security Compliance

In compliance with GDPR standards, SQream now requires a strong password policy when accessing the CLI or Studio. For more information, see Password Policy.

#### 14.1.2.4 Known Issues

There were no known issues in Version 2022.1.1.

#### 14.1.2.5 Resolved Issues

The following table lists the issues that were resolved in Version 2022.1.1:

| SQ No. | Description |
|---|---|
| SQ-6419 | An internal compiler error occurred when casting Numeric literals in an aggregation function. |
| SQ-10873 | Inserting 100K bytes into a text column resulted in an unclear error message. |
| SQ-10955 | Unneeded reads were occurring when filtering by date. |

#### 14.1.2.6 Operations and Configuration Changes

The `login_max_retries` configuration flag is required for adjusting the permitted log-in attempts.

For more information, see Adjusting the Permitted Log-In Attempts.

#### 14.1.2.7 Naming Changes

No relevant naming changes were made.

#### 14.1.2.8 Deprecated Features

In SQream Acceleration Studio 5.4.7, the **Configuration** section has been temporarily disabled and will be enabled at a later date. In addition, the **Log Lines** tab in the **Log** section has been removed.

#### 14.1.2.9 End of Support

The End of Support section is not relevant to Version 2022.1.1.

#### 14.1.2.10 Upgrading to v2022.1.1

1. Generate a back-up of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path');
   ```

   **Tip:** SQream recommends storing the generated back-up locally in case needed.

   SQream runs the Garbage Collector and creates a clean backup tarball package.

2. Shut down all SQream services.

3. Extract the recently created back-up file.

4. Replace your current metadata with the metadata you stored in the back-up file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

   ```
   $ ./upgrade_storage <levelDB path>
   ```

   **Note:** Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

### 14.1.3 Release Notes 2022.1

The 2022.1 release notes were released on 7/19/2022 and describe the following:



#### 14.1.3.1 Version Content

The 2022.1 Release Notes describe the following:

- Enhanced security features.
- New data manipulation command.
- Additional data ingestion format.

#### 14.1.3.2 Storage Version

The storage version presently in effect is version 40.

#### 14.1.3.3 New Features

The 2022.1 Release Notes include the following new features:

##### 14.1.3.3.1 Data Encryption

SQream now supports data encryption mechanisms in accordance with **General Data Protection Regulation (GDPR)** standards.

Using the data encryption feature may lead to a maximum of a 10% increase in performance degradation.

For more information, see Data Encryption.

##### 14.1.3.3.2 Update Feature

SQream now supports the DML **Update** feature, which is used for modifying the value of certain columns in existing rows.

For more information, see UPDATE.

##### 14.1.3.3.3 Avro Ingestion

SQream now supports ingesting data from Avro files.

For more information, see Inserting Data from Avro.

#### 14.1.3.4 Known Issues

The following table lists the known issues for Version 2022.1:

| SQ No. | Description |
|---|---|
| SQ-7732 | Reading numeric columns from an external Parquet file generated an error. |
| SQ-9889 | Running a query including Thai characters generated an internal runtime error. |
| SQ-10071 | Error on existing subqueries with TEXT and VARCHAR equality condition |
| SQ-10191 | The `ALTER DEFAULT SCHEMA` command was not functioning correctly. |
| SQ-10629 | Inserting data into a table significantly slowed down running queries. |
| SQ-10659 | Using a comment generated a compile error. |

#### 14.1.3.5 Resolved Issues

The following table lists the issues that were resolved in Version 2022.1:

| SQ No. | Description |
|---|---|
| SQ-10111 | Reading numeric columns from an external Parquet file generated an error. |

#### 14.1.3.6 Operations and Configuration Changes

No relevant operations and configuration changes were made.

#### 14.1.3.7 Naming Changes

No relevant naming changes were made.

#### 14.1.3.8 Deprecated Features

In SQream version 2022.1 the `VARCHAR` data type has been deprecated and replaced with `TEXT`. SQream will maintain `VARCHAR` in all previous versions until completing the migration to `TEXT`, at which point it will be deprecated in all earlier versions. SQream also provides an automated and secure tool to facilitate and simplify migration from `VARCHAR` to `TEXT`.

If you are using an earlier version of SQream, see the Using Legacy String Literals configuration flag.

#### 14.1.3.9 End of Support

The End of Support section is not relevant to Version 2022.1.

#### 14.1.3.10 Upgrading to v2022.1

1. Generate a backup of the metadata by running the following command:

   ```
   $ select backup_metadata('out_path', 'single_file');
   ```

   **Tip:** SQream recommends storing the generated backup locally in case needed.

   SQream runs the Garbage Collector and creates a multi-file directory as specified in the `out_path`.

2. Shut down all SQream services.

3. Extract the recently created backup file.

4. Replace your current metadata with the metadata you stored in the backup file.

5. Navigate to the new SQream package bin folder.

6. Run the following command:

   ```
   $ ./upgrade_storage <levelDB path>
   ```

   **Note:** Upgrading from a major version to another major version requires you to follow the **Upgrade Storage** step. This is described in Step 7 of the Upgrading SQream Version procedure.

## 14.2 Release Notes 2021.2

The 2021.2 Release Notes describe the following releases:

### 14.2.1 Release Notes 2021.2.1.24

The 2021.2.1.24 release notes were released on 7/28/2022 and describe the following:



#### 14.2.1.1 Version Content

The 2021.2.1.24 Release Notes includes a query maintenance feature.

#### 14.2.1.2 New Features

The 2021.2.1.24 Release Notes include the following new features:



##### 14.2.1.2.1 Query Healer

The new **Query Healer** feature periodically examines the progress of running statements, and is used for query maintenance.

For more information, see Query Healer.

#### 14.2.1.3 Resolved Issues

The following table lists the resolved issues for Version 2021.2.1.24:

| SQ No. | Description |
|---|---|
| SQ-10606 | Queries were getting stuck in the queue for a prolonged time. |
| SQ-10691 | The DB schema identifier was causing an error when running queries from joins suite. |
| SQ-10918 | The Workload Manager was only assigning jobs sequentially, delaying user SQLs assigned to workers running very large jobs. |
| SQ-10955 | Metadata filters were not being applied when users filtered by nullable dates using `dateadd` |

#### 14.2.1.4 Known Issues

The following table lists the known issues for Version 2021.2.1.24:

| SQ No. | Description |
|---|---|
| SQ-10071 | An error occurred on existing subqueries with `TEXT` and `VARCHAR` equality conditions. |
| SQ-10902 | Inserting a null value into non-null column was causing SQream to crash. |
| SQ-11088 | Specific workers caused low performance during compilation. |

#### 14.2.1.5 Operations and Configuration Changes

The following configuration flags were added:

- is_healer_on
- healer_max_inactivity_hours
- login_max_retries

#### 14.2.1.6 Naming Changes

No relevant naming changes were made.

#### 14.2.1.7 Deprecated Features

Version 2021.2.1.24 includes no deprecated features.

#### 14.2.1.8 End of Support

The End of Support section is not relevant to Version 2021.2.1.24.

### 14.2.2 Release Notes 2021.2.1

The 2021.2.1 release notes were released on 15/12/2021 and describes the following:



#### 14.2.2.1 New Features

The 2021.2.1 Release Notes include the following new features:



##### 14.2.2.1.1 CREATE TABLE

SQream now supports duplicating the column structure of an existing table using the `LIKE` clause.

For more information, see Duplicating the Column Structure of an Existing Table.

##### 14.2.2.1.2 PERCENTILE FUNCTIONS

SQream now supports the following aggregation functions:

- PERCENTILE_CONT
- PERCENTILE_DISC
- MODE

##### 14.2.2.1.3 REGEX REPLACE

SQream now supports the `REGEXP_REPLACE` function for finding and replacing text column substrings.

For more information, see REGEX_REPLACE.

##### 14.2.2.1.4 Delete Optimization

The `DELETE` statement can now delete values that contain multi-table conditions.

For more information, see Deleting Values that Contain Multi-Table Conditions.

For more information, see REGEX_REPLACE.

#### 14.2.2.2 Performance Enhancements

The **Performance Enhancements** section is not relevant to Version 2021.2.1.

#### 14.2.2.3 Resolved Issues

The following table lists the issues that were resolved in Version 2021.2.1:

| SQ No. | Description |
|---|---|
| SQ-8267 | A method has been provided for including the `GROUP BY` and `DISTINCT COUNT` statements. |

#### 14.2.2.4 Known Issues

The **Known Issues** section is not relevant to 2021.2.1.

#### 14.2.2.5 Naming Convention Modifications

The **Naming Convention Modifications** section is not relevant to Version 2021.2.1.

#### 14.2.2.6 End of Support

The **End of Support** section is not relevant to Version 2021.2.1.

#### 14.2.2.7 Deprecated Features

The **Deprecated Components** section is not relevant to Version 2021.2.1.

### 14.2.3 Release Notes 2021.2

The 2021.2 release notes were released on 13/9/2021.



#### 14.2.3.1 New Features

The 2021.2 Release Notes include the following new features:



##### 14.2.3.1.1 New Driver Compatibility

The 2021.2 release supports the following drivers:

- **JDBC** - new driver version (JDBC 4.5) with important bug fixes.
- **ODBC** - ODBC 4.1.1. available on request.
- **NodeJS** - all versions starting with NodeJS 4.0. SQream recommends the latest version (NodeJS 4.2.4).
- **Dot Net** - SQream recommends version version 3.02 (compatible with DotNet version 48).
- **Pysqream** - pysqream 3.1.2

##### 14.2.3.1.2 Centralized Configuration System

SQream now uses a new configuration system based on centralized configuration accessible from SQream Studio.

For more information, see the following:

- Configuration - describes how to configure your instance of SQream from a centralized location.
- SQream Studio 5.4.3 - configure your instance of SQream from Studio.

#### 14.2.3.1.3 Qualifying Schemas Without Providing an Alias

When running queries, SQream now supports qualifying schemas without providing an alias.

For more information, see CREATE_SCHEMA.

#### 14.2.3.1.4 Double-Quotations Supported When Importing and Exporting CSVs

When importing and exporting CSVs, SQream now supports using quotation characters other than double quotation marks (`"`).

For more information, see the following:

- COPY_FROM
- COPY_TO

Note the following:

- Leaving *<x>* unspecified uses the default value of standard double quotations `”`.
- The quotation character must be a single, 1-byte printable ASCII character. The same octal syntax of the copy command can be used.
- The quote character cannot be contained in the field delimiter, record delimiter, or null marker.
- Double-quotations can be customized when the `csv_fdw` value is used with the `COPY FROM` and `CREATE FOREIGN TABLE` statements.
- The default escape character always matches the quote character, and can be overridden by using the `ESCAPE = {'\\' | E'\XXX')` syntax as shown in the following examples:

```
copy t from wrapper csv_fdw options (location = '/tmp/file.csv', escape='\\');
```

```
copy t from wrapper csv_fdw options (location = '/tmp/file.csv', escape=E'\017');
```

```
copy t to wrapper csv_fdw options (location = '/tmp/file.csv', escape='\\');
```

For more information, see the following statements:

- COPY_FROM
- Foreign Tables

#### 14.2.3.2 Performance Enhancements

In Version 2021.2, an advanced smart spooling mechanism splits spool memory based on required CP usage.

#### 14.2.3.3 Resolved Issues

The following table lists the issues that were resolved in Version 2021.2:

| SQ No. | Description |
|---|---|
| SQ-8294 | Quote qualifiers were not present in exported file, preventing it from being reloaded. |
| SQ-8288 | Saved `TEXT` query parameters were not supported. |
| SQ-8266 | A data loading issue occurred related to column order. |

#### 14.2.3.4 Known Issues

The **Known Issues** section is not relevant to Version 2021.2.

#### 14.2.3.5 Naming Convention Modifications

The **Naming Convention Modifications** describes SQream features, such as data types or statements, that have been renamed.

##### 14.2.3.5.1 NVARCHAR Data Type Renamed TEXT

The `NVARCHAR` data type has been renamed `TEXT`.

For more information on the `TEXT` data type, see String (TEXT)

#### 14.2.3.6 End of Support

The **End of Support** section is not relevant to Version 2021.2.

#### 14.2.3.7 Deprecated Features

The **Deprecated Components** section is not relevant to Version 2021.2.

#### 14.2.3.8 Upgrading Your SQream Version

The **Upgrading Your SQream Version** section describes the following:



##### 14.2.3.8.1 Upgrading Your Storage Version

When upgrading from a SQream version earlier than 2021.2 you must upgrade your storage version, as shown in the following example:

```
$ cat /etc/sqream/sqream1_config.json |grep cluster
$ ./upgrade_storage <cluster path>
```

For more information on upgrading your SQream version, see Upgrading SQream Version.

##### 14.2.3.8.2 Upgrading Your Client Drivers

For more information on the client drivers for version 2021.2, see Client Drivers for 2021.2.

##### 14.2.3.8.3 Configuring Your Instance of SQream

A new configuration method is used starting with Version 2021.2.

For more information about configuring your instance of SQream, see Configuring SQream.

## 14.3 Release Notes 2021.1

The 2021.1 Release Notes describe the following releases:

### 14.3.1 Release Notes 2021.1.2

The 2021.1.2 release notes were released on 8/9/2021 and describe the following:

#### 14.3.1.1 New Features

The 2021.1.2 Release Notes include the following new features:



##### 14.3.1.1.1 Aliases Added to SUBSTRING Function and Length Argument

The following aliases have been added:

- length - `len`
- substring - `substr`

##### 14.3.1.1.2 Data Type Aliases Added

The following data type aliases have been added:

- INTEGER - `int`
- DECIMAL - `numeric`
- DOUBLE PRECISION - `double`
- CHARACTER/CHAR - `text`
- NATIONAL CHARACTER/NATIONAL CHAR/NCHAR - `text`
- CHARACTER VARYING/CHAR VARYING - `text`
- NATIONAL CHARACTER VARYING/NATIONAL CHAR VARYING/NCHAR VARYING - `text`

##### 14.3.1.1.3 String Literals Containing ASCII Characters Interepreted as TEXT

SQream now interprets all string literals, including those containing ASCII characters, as `text`.

For more information, see Data Types.

##### 14.3.1.1.4 Decimal Literals Interpreted as Numeric Columns

SQream now interprets literals containing decimal points as `numeric` instead of as `double`.

For more information, see Data Types.

##### 14.3.1.1.5 Roles Area Added to Studio Version 5.4.3

The **Roles** area has been added to Studio version 5.4.3. From the Roles area users can create and assign roles and manage user permissions.

#### 14.3.1.2 Resolved Issues

The following list describes the resolved issues:

- In Parquet files, `float` columns could not be mapped to SQream `double` columns. This was fixed.
- The `REPLACE` function only supported constant values as arguments. This was fixed.
- The `LIKE` function did not check for incorrect patterns or handle escape characters. This was fixed.

### 14.3.2 Release Notes 2021.1.1

The 2021.1.1 release notes were released on 7/27/2021 and describe the following:



#### 14.3.2.1 New Features

The 2021.1.1 Release Notes include the following new features:



##### 14.3.2.1.1 Complete Ranking Function Support

SQream now supports the following new ranking functions:

| Function | Return Type | Description |
|---|---|---|
| first_value | Same type as value | Returns the value in the first row of a window. |
| last_value | Same type as value | Returns the value in the last row of a window. |
| nth_value | Same type as value | Returns the value in a specified (`n`) row of a window. if the specified row does not exist, this function returns `NULL`. |
| dense_rank | bigint | Returns the rank of the current row with no gaps. |
| percent_rank | double | Returns the relative rank of the current row. |
| cume_dist | double | Returns the cumulative distribution of rows. |
| ntile(buckets) | integer | Returns an integer ranging between `1` and the argument value, dividing the partitions as equally as possible. |

For more information, navigate to Windows Functions and scroll to the Ranking Functions table.

#### 14.3.2.2 Resolved Issues

The following list describes the resolved issues:

- SQream did not support exporting and reading **Int64** columns as **bigint** in Parquet. This was fixed.
- The Decimal column was not supported when inserting data from Parquet files. This was fixed.
- Values in Parquet Numeric columns were not being converted correctly. This was fixed.
- Converting `string` data type to `datetime` was not working correctly. This was fixed.
- Casting `datetime` to `text` truncated the time. This was fixed.

### 14.3.3 Release Notes 2021.1

The 2021.1 release notes were released on 6/13/2021 and describe the following:

#### 14.3.3.1 Version Content

The 2021.1 Release Notes describes the following:

- Major feature release targeted for all on-premises customers.
- Basic Cloud functionality.

#### 14.3.3.2 New Features

The 2021.1 Release Notes include the following new features:



##### 14.3.3.2.1 SQream DB on Cloud

SQream DB can now be run on AWS, GCP, and Azure.

##### 14.3.3.2.2 Numeric Data Types

SQream now supports Numeric Data types for the following operations:

- All join types.
- All aggregation types (not including Window functions).
- Scalar functions (not including some trigonometric and logarithmic functions).

For more information, see Numeric Data Types.

##### 14.3.3.2.3 Text Data Type

SQream now supports TEXT data types in all operations, which is default string data type for new projects.

- Sqream supports VARCHAR functionalty, but recommends using TEXT.
- TEXT data enhancements introduced in Release Notes version 2020.3.1:
  - Support text columns in queries with multiple distinct aggregates.
  - Text literal support for all functions.

For more information, see String Types.

#### 14.3.3.2.4 Supports Scalar Subqueries

SQream now supports running initial scalar subqueries.

For more information, see Subqueries.

#### 14.3.3.2.5 Literal Arguments

SQream now supports literal arguments for functions in all cases where column/scalar arguments are supported.

#### 14.3.3.2.6 Simple Scalar SQL UDFs

SQream now supports simple scalar SQL UDF's.

For more information, see Simple Scalar SQL UDF's.

#### 14.3.3.2.7 Logging Enhancements

The following log information has been added for the following events:

- Compilation start time.
- When the first metadata callback in the compiler (if relevant).
- When the last metadata callback in the compiler (if relevant).
- When the log started attempting to apply locks.
- When a statement entered the queue.
- When a statement exited the queue.
- When a client has connected to an instance of **sqreamd** (if it reconnects).
- When the log started executing.

#### 14.3.3.2.8 Improved Presented License Information

SQream now displays information related to data size limitations, expiration date, type of license shown by the new UF. The **Utility Function (UF)** name is `get_license_info()`.

For more information, see GET_LICENSE_INFO.

#### 14.3.3.2.9 Optimized Foreign Data Wrapper Export

Sqream now supports exporting to multiple files concurrently. This is useful when you need to reduce file size to more easily export multiple files.

The following is the correct syntax for exporting multiple files concurrently:

```
COPY table_name TO fdw_name OPTIONS(max_file_size=size_in_bytes,enforce_single_file=
↪{TRUE|FALSE});
```

The following is an example of the correct syntax for exporting multiple files concurrently:

```
COPY my_table1 TO my_ext_table OPTIONS(max_file_size=500000,enforce_single_file=TRUE);
```

The following apply:

- Both of the parameters in the above example are optional.
- The `max_file_size` value is specified in bytes and can be any positive value. The default value is `16*2^20` (16MB).
- When the `enforce_single_file` value is set to `TRUE`, only one file is created, and its size is not limited by the `max_file_size` value. Its default value is `TRUE`.

### 14.3.3.3 Main Features

The following list describes the main features:

- SQreamDB available on AWS.
- SQreamDB available on GCP.
- SQreamDB available on Azure.
- SQream usages storage located on Object Store (as opposed to local disks) for the above three cloud providers.
- SQream now supports Microstrategy.
- Supports MVP licensing system.
- A new literal syntax containing character escape semantics for string literals has been added.
- Supports optimizing exporting foreign data wrappers.
- Supports truncating Numeric values when ingested from ORC and CSV files.
- Supports catalog Utility Function that accepts valid SQL patterns and escape characters.
- Supports creating a basic random data foreign data wrapper for non-text types.
- The new foreign data wrapper `random_fdw` has been introduced for non-text types.
- Supports simple scalar SQL UDF's.
- SQream parses its own logs as CSV's.

#### 14.3.3.4 Resolved Issues

The following list describes the resolved issues:

- Copying text from a CSV file to the TEXT column without closing quotes caused SQream to crash. This was fixed.
- Using an unsupported function call generated an incorrect insert error. This was fixed.
- Using the `insert into` function from `table_does_not_exist` generated an incorrect error.
- SQream treated inserting `*` in `select_distinct` as one column. This was fixed.
- Using certain encodeKey functions generated errors. This was fixed.
- Compile errors occurred while running decimal datatype sets. This was fixed.
- Running the `select table_name,row_count from sqream_catalog.tables order by row_count limit 5` query generated an internal runtime error.
- Using wildcards (such as `*.x.y`) did not work in parquet files. This was fixed.
- Executing `log*(x,y)` generated an incorrect error message. This was fixed.
- The `internal runtime error` type doesn't have a fixed size when doing max on text on develop.
- The `min` and `max` on `TEXT` were significantly slower than `varchar`. This was fixed.
- Running `regexp_instr` generated an empty regular expression. This was fixed.
- Schemas with foreign tables could be dropped. This was fixed.

#### 14.3.3.5 Operations and Configuration Changes

##### 14.3.3.5.1 Recommended SQream Configuration on Cloud

For more information about AWS, see Amazon S3.

##### 14.3.3.5.2 Optimized Foreign Data Wrapper Export Configuration Flag

SQream now has a new `runtimeGlobalFlags` flag called `WriteToFileThreads`.

This flag configures the number of threads in the **WriteToFile** function. The default value is `16`.

For more information about the `runtimeGlobalFlags` flag, see the **Runtime Global Flags** table in Configuration.

#### 14.3.3.6 Naming Changes

No relevant naming changes were made.

#### 14.3.3.7 Deprecated Features

No features were depecrated.

#### 14.3.3.8 Known Issues and Limitations

The the list below describes the following known issues and limitations:

- In cases when selecting top 1 from foreign table using the Parquet format with an hdfs path, SQream experienced an error.
- Internal Runtime Error occurred when SQream was unable to find column in reorder columns.
- Casting datetime to text truncates the time segment.
- In the **select** list, the compiler generates an error when a count is used as an alias.
- Performance degradation occurred when joins made on small tables.
- SQream causes a logging error when using copy from logs.
- Deploying S3 requires setting the `ObjectStoreClients` parameter to `40`.

#### 14.3.3.9 Upgrading to v2021.1

Due to the known issue of a limitation on the amount of access requests that can be simultaneously sent to AWS, deploying S3 requires setting the `ObjectStoreClients` parameter to `40`.

## 14.4 Release Notes 2020.3

The 2020.3 Release Notes describe the following releases:

### 14.4.1 Release Notes 2020.3.2.1

The 2020.3.2.1 release notes were released on October 8, 2020 and describe the following:

#### 14.4.1.1 Overview

SQream DB v2020.3.2.1 contains major performance improvements and some bug fixes.

#### 14.4.1.2 Performance Enhancements

- Metadata on Demand optimization resulting in reduced latency and improved overall performance.

#### 14.4.1.3 Known Issues and Limitations

- Multiple count distinct operations is enabled for all data types.

#### 14.4.1.4 Upgrading to v2020.3.2.1

Versions are available for IBM POWER9, RedHat (CentOS) 7, Ubuntu 18.04, and other OSs via Docker.

Contact your account manager to get the latest release of SQream DB.

### 14.4.2 What's new in 2020.3.2

SQream DB v2020.3.2 contains major performance improvements and some bug fixes.

#### 14.4.2.1 Performance Enhancements

- Metadata on Demand optimization resulting in reduced latency and improved overall performance

#### 14.4.2.2 Known Issues & Limitations

- Bug with STDDEV_SAMP,STDDEV_POP and STDEV functions
- Window function query returns wrong results
- rank() in window function sometimes returns garbage
- Window function on null value could have bad result
- Window function lead() on varchar can have garbage results
- Performance degradation when using "groupby" or outer_join

#### 14.4.2.3 Upgrading to v2020.3.2

Versions are available for IBM POWER9, RedHat (CentOS) 7, Ubuntu 18.04, and other OSs via Docker.

Contact your account manager to get the latest release of SQream DB.

### 14.4.3 Release Notes 2020.3.1

The 2020.3.1 release notes were released on October 8, 2020 and describe the following:



#### 14.4.3.1 New Features

The following list describes the new features:

- **TEXT data type:**
  - Full support for `MIN` and `MAX` aggregate functions on `TEXT` columns in `GROUP BY` queries.
  - Support Text-type as window partition keys (e.g., select distinct name, max(id) over (partition by name) from `textTable;`).
  - Support Text-type fields in windows order by keys.
  - Support join on `TEXT` columns (such as `t1.x = t2.y` where `x` and `y` are columns of type `TEXT`).
  - Complete the implementation of `LIKE` on `TEXT` columns (previously limited to prefix and suffix).
  - Support for cast fromm `TEXT` to `REAL/FLOAT`.
  - New string function - `REPEAT` for repeating a string value for a specified number of times.
- Support mapping `DECIMAL ORC` columns to SQream's floating-point types.
- Support `LIKE` on non-literal patterns (such as columns and complex expressions).
- Catch OS signals and save the signal along with the stack trace in the SQream debug log.
- Support equijoin conditions on columns with different types (such as `tinyint`, `smallint`, `int` and `bigint`).
- `DUMP_DATABASE_DDL` now includes foreign tables in the output.
- New utility function - `TRUNCATE_IF_EXISTS`.

#### 14.4.3.2 Performance Enhancements

The following list describes the performance enhancements:

- Introduced the "MetaData on Demand" feature which results in signicant proformance improvements.
- Implemented regex functions (`RLIKE`, `REGEXP_COUNT`, `REGEXP_INSTR`, `REGEXP_SUBSTR`, `PATINDEX`) for `TEXT` columns on GPU.

#### 14.4.3.3 Resolved Issues

The following list describes the resolved issues:

- Multiple distinct aggregates no longer need to be used with developerMode flag.
- In some scenarios, the `statement_id` and `connection_id values` are incorrectly recorded as `-1` in the log.
- `NOT RLIKE` is not supported for `TEXT` in the compiler.
- Casting from `TEXT` to `date/datetime` returns an error when the `TEXT` column contains `NULL`.

#### 14.4.3.4 Known Issues and Limitations

No known issues and limitations.

#### 14.4.3.5 Upgrading to v2020.3.1

Versions are available for IBM POWER9, RedHat (CentOS) 7, Ubuntu 18.04, and other OSs via Docker.

Contact your account manager to get the latest release of SQream DB.

### 14.4.4 Release Notes 2020.3

The 2020.3 release notes were released on October 8, 2020 and describes the following:



#### 14.4.4.1 Overview

SQream DB v2020.3 contains new features, performance enhancements, and resolved issues.

#### 14.4.4.2 New Features

The following list describes the new features:

- Parquet and ORC files can now be exported to local storage, S3, and HDFS with copy_to and foreign data wrappers.
- New error tolerance features when loading data with foreign data wrappers.
- `TEXT` is ramping up with new features (previously only available with VARCHARs):
  - substring, lower, ltrim, charindex, replace, etc.
  - Binary operators - concat, like, etc.

  - Casts to and from `TEXT`

- sqream_studio v5.1
  - New log viewer helps you track and debug what's going on in SQream DB.
  - Dashboard now also available for non-k8s deployments.
  - The editor contains a new query concurrency tool for date and numeric ranges.

#### 14.4.4.3 Performance Enhancements

The following list describes the performance enhancements:

- Error handling for CSV FDW.
- Enable logging errors - ORC, Parquet, CSV.
- Add limit and offset options to `csv_fdw` import.
- Enable logging errors to an external file when skipping CSV, Parquet, and ORC errors.
- Option to specify date format to the CSV FDW.
- Support all existing `VARCHAR` functions with `TEXT` on GPU.
- Support `INSERT INTO` + `ORDER BY` optimization for non-clustered tables.
- Performance improvements with I/O.

#### 14.4.4.4 Resolved Issues

The following list describes the resolved issues:

- Better error message when passing the max errors limit. This was fixed.
- `showFullExceptionInfo` is no longer restricted to Developer Mode. This was fixed.
- An `StreamAggregateA` reduction error occured when performing aggregation on a `NULL` column. This was fixed.
- Insert into query fails with ""Error at Sql phase during Stages ""rewriteSqlQuery"". This was fixed.
- Casting from `VARCHAR` to `TEXT` does not remove the spaces. This was fixed.
- An `Internal Runtime Error t1.size() == t2.size()` occurs when querying the `sqream_catalog.delete_predicates`. This was fixed.
- `spoolMemoryGB` and `limitQueryMemoryGB` show incorrectly in the **runtime global** section of `show_conf.` This was fixed.
- Casting empty text to `int` causes illegal memory access. This was fixed.
- Copying from the `TEXT` field is 1.5x slower than the `VARCHAR` equivalent. This was fixed.
- `TPCDS 10TB - Internal runtime error (std::bad_alloc: out of memory)` occurs on 2020.1.0.2. This was fixed.
- An unequal join on non-existing `TEXT` caused a system crash. This was fixed.
- An `Internal runtime time error` occured when using `TEXT (tpcds).` This was fixed.
- Copying CSV with a quote in the middle of a field to a `TEXT` field does not produce the required error. This was fixed.

- Cannot monitor long network insert loads with SQream. This was fixed.
- Upper and like performance on `TEXT`. This was fixed.
- Insert into from 4 instances would get stuck (hanging). This was fixed.
- An invalid formatted CSV would cause an insufficient memory error on a `COPY FROM` statement if a quote was not closed and the file was much larger than system memory. This was fixed.
- `TEXT` columns cannot be used with an outer join together with an inequality check (!= , <>). This was fixed.

#### 14.4.4.5 Known Issues And Limitations

The following list describes the known issues and limitations:

- Cast from `TEXT` to a `DATE` or `DATETIME` errors when the `TEXT` column contains `NULL`
- Casting an empty `TEXT` field to an `INT` type returns `0` instead of erroring
- Multiple `COUNT( distinct ... )` operations on the `TEXT` data type are currently unsupported
- Multiple `COUNT( distinct ... )` operations within the same query are limited to "developer mode" due to an instability that was identified. If you rely on this feature, contact your SQream account manager to enable this feature.

#### 14.4.4.6 Upgrading to v2020.3

Versions are available for IBM POWER9, RedHat (CentOS) 7, Ubuntu 18.04, and other OSs via Docker.

Contact your account manager to get the latest release of SQream.

## 14.5 Release Notes 2020.2

SQream v2020.2 contains some new features, improved performance, and bug fixes.

This version has new window ranking function and a new editor UI to empower data users to analyze more data with less friction.

As always, the latest release improves reliability and performance, and makes getting more data into SQream easier than ever.

### 14.5.1 New Features

#### 14.5.1.1 UI

- New sqream_studio replaces the previous Statement Editor.

#### 14.5.1.2 Integrations

- Our Python driver (pysqream) now has an SQLAlchemy dialect. Customers can write high-performance Python applications that make full use of SQream - connect, query, delete, and insert data. Data scientists can use pysqream with Pandas, Numpy, and AI/ML frameworks like TensorFlow for direct queries of huge datasets.

#### 14.5.1.3 SQL Support

- Added lag/lead ranking functions to our window_functions support. We will have more features coming in the next version.
- New syntax preview for external_tables. Foreign tables replace external tables, with improved functionality.

  You can keep using the existing foreign table syntax for now, but it may be deprecated in the future.

```
CREATE FOREIGN TABLE orc_example
(
   name varchar(40),
   Age tinyint,
   Salary float
 )
WRAPPER orc_fdw
OPTIONS
( LOCATION = 'hdfs://hadoop-nn.piedpiper.com:8020/demo-data/example.orc' );
```

### 14.5.2 Improvements and Fixes

SQream v2020.2 includes hundreds of small new features and tunable parameters that improve performance, reliability, and stability.

- ~100 bug fixes, including:
  - Fixed CSV handling for DOS newlines
  - Fixed "out of bounds" message when several layers of nested `substring`, `cast`, and `to_hex` were used to produce one value.
  - Fixed "Illegal memory access" that would occur in extremely rare situations on all-text tables
  - Window functions can now be used with all aggregations
  - Fixed situation where a single worker may use more than one GPU that isn't allocated to it
  - Text columns can now be added to existing tables with alter_table
- New data_clustering syntax that can improve query performance for unsorted data

### 14.5.3 Operations

- When upgrading from a previous version of SQream (for example, v2019.2), the storage version must be upgraded using the *upgrade_storage* utility: `./bin/upgrade_storage /path/to/storage/sqreamdb/`
- A change in memory allocation behaviour in this version sees the introduction of a new setting, `limitQueryMemoryGB`. This is an addition to the previous `spoolMemoryGB` setting.

  A good rule-of-thumb is to allow 5% system memory for other processes. The spool memory allocation should be around 90% of the total memory allocated.

   - `limitQueryMemoryGB` defines how much total system memory is used by the worker. The recommended setting is (`total host memory` - 5%) / `sqreamd workers on host`.
   - `spoolMemoryGB` defines how much memory is set aside for spooling, out of the total system memory allocated in `limitQueryMemoryGB`. The recommended setting is 90% of the `limitQueryMemoryGB`.

This setting must be set lower than the `limitQueryMemoryGB` setting.

For example, for a machine with 512GB of RAM and 4 workers, the recommended settings are:

   - `limitQueryMemoryGB` - `⌊(512 * 0.95 / 4)⌋ → ~ 486 / 4 → 121`.
   - `spoolMemoryGB` - `⌊( 0.9 * limitQueryMemoryGB )⌋ → ⌊( 0.9 * 121 )⌋ → 108`

Example settings per-worker, for 512GB of RAM and 4 workers:

```
"runtimeFlags": {
   "limitQueryMemoryGB" : 121,
   "spoolMemoryGB" : 108
```

### 14.5.4 Known Issues and Limitations

- An invalid formatted CSV can cause an `insufficient memory` error on a copy_from statement if a quote isn't closed and the file is much larger than system memory.
- Multiple `COUNT( distinct ... )` operations within the same query are limited to "developer mode" due to an instability that was identified. If you rely on this feature, contact your SQream account manager to enable this feature.
- `TEXT` columns can't be used with an outer join together with an inequality check (`!= , <>`)

### 14.5.5 Upgrading to Version 2020.2

Versions are available for IBM POWER9, RedHat (CentOS) 7, Ubuntu 18.04, and other OSs via Docker.

Contact your account manager to get the latest release of SQream.

## 14.6 Release Notes 2020.1

SQream DB v2020.1 contains lots of new features, improved performance, and bug fixes.

This is the first release of 2020, with a strong focus on integration into existing environments. The release includes connectivity to Hadoop and other legacy data warehouse ecosystems. We're also bringing lots of new capabilities to our analytics engine, to empower data users to analyze more data with less friction.

The latest release vastly improves reliability and performance, and makes getting more data into SQream DB easier than ever.

The core of SQream DB v2020.1 contains new integration features, more analytics capabilities, and better drivers and connectors.

### 14.6.1 New features

#### 14.6.1.1 Integrations

- Load files directly from *S3 buckets*. Customers with columnar data in S3 data lakes can now access the data directly. All that is needed is to simply point an external table to an S3 bucket with Parquet, ORC, or CSV objects. This feature is available on all deployments of SQream DB – in the cloud and on-prem.
- Load files directly from *HDFS*. SQream DB now comes with built-in, native HDFS support for directly loading data from Hadoop-based data lakes. Our focus on helping Hadoop customers do more with their data led us to develop this feature, which works out of the box. As a result, SQream DB can now not only read but also write data, and intermediate results back to HDFS for HIVE and other data consumers. SQream DB now fits seamlessly into a Hadoop data pipeline.
- Import *ORC files*, through external_tables. ORC files join Parquet as files that can be natively accessed and inserted into SQream DB tables.
- Python driver (pysqream) is now DB-API v2.0 compliant. Customers can write high-performance Python applications that make full use of SQream DB - connect, query, delete, and insert data. Data scientists can use pysqream with Pandas, Numpy, and AI/ML frameworks like TensorFlow for direct queries of huge datasets.
- Certified Tableau JDBC connector (taco), now also supported on MacOS. Users are encouraged to install the new JDBC connector.
- All logs are now unified into one log, which can be analyzed with SQream DB directly. See *Logging* for more information.

#### 14.6.1.2 SQL support

- Added frames and frame exclusions to window_functions. This is available for preview, with more features coming in the next version.

  The new frames and frame exclusionsfeature adds complex analytics capabilities to the already powerful window functions.
- New datatype - `TEXT`, which replaces `NVARCHAR` directly with UTF-8 support and improved performance.

  Unlike `VARCHAR`, the new `TEXT` data type has no restrictions on size, and carries no performance overhead as the text sizes grow.
- `TEXT` join keys are now supported
- Added lots of new *aggregate functions*, including `VAR_SAMP`, `VAR_POP`, `COVAR_POP`, etc.

### 14.6.2 Improvements and fixes

SQream DB v2020.1 includes hundreds of small new features and tunable parameters that improve performance, reliability, and stability. Existing SQream DB users can expect to see a general speedup of around 10% on most statements and queries!

- 207 bug fixes, including:
  - Improved performance of both inner and outer joins
  - Fixed wrong results on STDDEV (0 instead of `NULL`)
  - Fixed wrong results on nested Parquet files
  - Fixed failing cast from `VARCHAR` to `FLOAT`

  - Fix `INSERT` that would fail on nullable values and non-nullable columns in some scenarios
  - Improved memory consumption, so `Out of GPU memory` errors should not occur anymore
  - Reduced long compilation times for very complex queries
  - Improved ODBC reliability
  - Fixed situation where some logs would clip very long queries
  - Improved error messages when dropping a schema with many objects
  - Fixed situation where Spotfire would not show table names
  - Fixed situation where some queries with UTF-8 literals wouldn't run through Tableau over ODBC
  - Significantly improved cache freeing and memory allocation
  - Fixed situation in which a malformed time (`24:00:00`) would get incorrectly inserted from a CSV
  - Fixed race condition in which loading thousands of small files from HDFScaused a memory leak
- The saved query feature can now be used with insert statements
- Faster "Deferred gather" algorithm for joins with text keys
- Faster filtering when using datepart
- Faster metadata tagging during load
- Fixed situation where some queries would get compiled twice
- saved_queries now support insert statements
- `highCardinalityColumns` can be configured to tell the system about *high selectivity* columns
- *sqream sql* starts up faster, can run on any Linux machine
- Additional CSV date formats (date parsers) added for compatibility

### 14.6.3 Behaviour changes

- `ClientCmd` is now known as *sqream sql*
- `NVARCHAR` columns are now known as `TEXT` internally
- Deprecated the ability to run `SELECT` and `COPY` at the same time on the same worker. This change is designed to protect against `out of GPU memory` issues. This comes with a configuration change, namely the `limitQueryMemoryGB` setting. See the operations section for more information.
- All logs are now unified into one log. See *Logging* for more information
- Compression changes:
  - The latest version of SQream DB could select a different compression scheme if data is reloaded, compared to previous versions of SQream DB. This internal change improves performance.
  - With `LZ4` compression, the maximum chunk size is limited to 2.1GB. If the chunk size is bigger, another compression may be selected - primarily `SNAPPY`.
- The following configuration flags have been deprecated:
  - `addStatementRechunkerAfterGpuToHost`
  - `increasedChunkSizeFactor`
  - `gpuReduceMergeOutputFactor`

- `fullSortInputMemFactor`
- `reduceInputMemFactor`
- `distinctInputMemFactor`
- `useAutoMemFactors`
- `autoMemFactorsVramFactor`
- `catchNotEnoughVram`
- `useNetworkRechunker`
- `useMemFactorInJoinOutput`

### 14.6.4 Operations

- The client-server protocol has been updated to support faster data flow, and more reliable memory allocations on the client side. End users are required to use only the latest *sqream sql*, *JDBC*, and *ODBC* drivers delivered with this version. See the *client driver download page* for the latest drivers and connectors.
- When upgrading from a previous version of SQream DB (for example, v2019.2), the storage version must be upgraded using the *upgrade_storage* utility: `./bin/upgrade_storage /path/to/storage/sqreamdb/`
- A change in memory allocation behaviour in this version sees the introduction of a new setting, `limitQueryMemoryGB`. This is an addition to the previous `spoolMemoryGB` setting.

  A good rule-of-thumb is to allow 5% system memory for other processes. The spool memory allocation should be around 90% of the total memory allocated.

  - `limitQueryMemoryGB` defines how much total system memory is used by the worker. The recommended setting is (`total host memory` - 5%) / `sqreamd workers on host`.
  - `spoolMemoryGB` defines how much memory is set aside for spooling, out of the total system memory allocated in `limitQueryMemoryGB`. The recommended setting is 90% of the `limitQueryMemoryGB`.

  This setting must be set lower than the `limitQueryMemoryGB` setting.

  For example, for a machine with 512GB of RAM and 4 workers, the recommended settings are:

  - `limitQueryMemoryGB` - `⌊(512 * 0.95 / 4)⌋ → ~ 486 / 4 → 121`.
  - `spoolMemoryGB` - `⌊( 0.9 * limitQueryMemoryGB )⌋ → ⌊( 0.9 * 121 )⌋ → 108`

  Example settings per-worker, for 512GB of RAM and 4 workers:

```
"runtimeGlobalFlags": {
   "limitQueryMemoryGB" : 121,
   "spoolMemoryGB" : 108
```

### 14.6.5 Known Issues & Limitations

- An invalid formatted CSV can cause an `insufficient memory` error on a copy_from statement if a quote isn't closed and the file is much larger than system memory.
- `TEXT` columns cannot be used in a window functions' partition
- Parsing errors are sometimes hard to read - the location points to the wrong part of the statement
- LZ4 compression may not be applied correctly on very large `VARCHAR` columns, which decreases performance

- Using `SUM` on very large numbers in window functions can error (`overflow`) when not used with an `ORDER BY` clause
- Slight performance decrease with dateadd in this version (<4%)
- Operations on Snappy-compressed ORC files are slower than their Parquet equivalents.

### 14.6.6 Upgrading to v2020.1

Versions are available for IBM POWER9, RedHat (CentOS) 7, Ubuntu 18.04, and other OSs via Docker.

Contact your account manager to get the latest release of SQream DB.

CHAPTER

# FIFTEEN

# TROUBLESHOOTING

The **Troubleshooting** page describes solutions to the following issues:

## 15.1 Remedying Slow Queries

The **Remedying Slow Queries** page describes how to troubleshoot the causes of slow queries.

The following table is a checklist you can use to identify the cause of your slow queries:

| Step | Description | Results |
|---|---|---|
| 1 | A single query is slow | If a query isn't performing as you expect, follow the *Query best practices* part of the *Optimization and Best Practices* guide.<br>If all queries are slow, continue to step 2. |
| 2 | All queries on a specific table are slow | 1. If all queries on a specific table aren't performing as you expect, follow the *Table design best practices* part of the *Optimization and Best Practices* guide.<br>2. Check for active delete predicates in the table. Consult the *Deleting Data* guide for more information.<br>If the problem spans all tables, continue to step 3. |
| 3 | Check that all workers are up | Use `SELECT show_cluster_nodes();` to list the active cluster workers.<br>If the worker list is incomplete, follow the cluster troubleshooting section below.<br>If all workers are up, continue to step 4. |
| 4 | Check that all workers are performing well | 1. Identify if a specific worker is slower than others by running the same query on different workers. (e.g. by connecting directly to the worker or through a service queue)<br>2. If a specific worker is slower than others, investigate performance issues on the host using standard monitoring tools (e.g. `top`).<br>3. Restart SQream DB workers on the problematic host.<br>If all workers are performing well, continue to step 5. |
| 5 | Check if the workload is balanced across all workers | 1. Run the same query several times and check that it appears across multiple workers (use `SELECT show_server_status()` to monitor)<br>2. If some workers have a heavier workload, check the service queue usage. Refer to the *Workload Manager* guide.<br>If the workload is balanced, continue to step 6. |
| 6 | Check if there are long running statements | 1. Identify any currently running statements (use `SELECT show_server_status()` |

## 15.2 Resolving Common Issues

The **Resolving Common Issues** page describes how to resolve the following common issues:

### 15.2.1 Troubleshooting Cluster Setup and Configuration

1. Note any errors - Make a note of any error you see, or check the *logs* for errors you might have missed.
2. If SQream DB can't start, start SQream DB on a new storage cluster, with default settings. If it still can't start, there could be a driver or hardware issue. *Contact SQream support*.
3. Reproduce the issue with a standalone SQream DB - starting up a temporary, standalone SQream DB can isolate the issue to a configuration issue, network issue, or similar.
4. Reproduce on a minimal example - Start a standalone SQream DB on a clean storage cluster and try to replicate the issue if possible.

### 15.2.2 Troubleshooting Connectivity Issues

1. Verify the correct login credentials - username, password, and database name.
2. Verify the host name and port
3. Try connecting directly to a SQream DB worker, rather than via the load balancer
4. Verify that the driver version you're using is supported by the SQream DB version. Driver versions often get updated together with major SQream DB releases.
5. Try connecting directly with *the built in SQL client*. If you can connect with the local SQL client, check network availability and firewall settings.

### 15.2.3 Troubleshooting Query Performance

1. Use show_node_info to examine which building blocks consume time in a statement. If the query has finished, but the results are not yet materialized in the client, it could point to a problem in the application's data buffering or a network throughput issue..
2. If a problem occurs through a 3rd party client, try reproducing it directly with *the built in SQL client*. If the performance is better in the local client, it could point to a problem in the application or network connection.
3. Consult the *Optimization and Best Practices* guide to learn how to optimize queries and table structures.

### 15.2.4 Troubleshooting Query Behavior

1. Consult the *SQL Statements and Syntax* reference to verify if a statement or syntax behaves correctly. SQream DB may have some differences in behavior when compared to other databases.
2. If a problem occurs through a 3rd party client, try reproducing it directly with *the built in SQL client*. If the problem still occurs, file an issue with SQream support.

### 15.2.5 File an issue with SQream support

To file an issue, follow our *Gathering Information for SQream Support* guide.

## 15.3 Examining Logs

See the *Collecting Logs and Metadata Database* section of the *Gathering Information for SQream Support* guide for information about collecting logs for support.

## 15.4 Identifying Configuration Issues

The **Troubleshooting Common Issues** page describes how to troubleshoot the following common issues:

Starting a SQream DB temporarily (not as part of a cluster, with default settings) can be helpful in identifying configuration issues.

Example:

```
$ sqreamd /home/rhendricks/raviga_database 0 5000 /home/sqream/.sqream/license.enc
```

**Tip:**

- Using `nohup` and `&` sends SQream DB to run in the background.
- It is safe to stop SQream DB at any time using `kill`. No partial data or data corruption should occur when using this method to stop the process.

  ```
  $ kill -9 $SQREAM_PID
  ```

## 15.5 Lock Related Issues

Sometimes, a rare situation can occur where a lock is never freed.

The workflow for troubleshooting locks is:

1. Identify which statement has obtained locks
2. Understand if the statement is itself stuck, or waiting for another statement
3. Try to abort the offending statement
4. Force the stale locks to be removed

For example, we will assume that the statement from the previous example is stuck (statement #`287`). We can attempt to abort it using stop_statement:

```
t=> SELECT STOP_STATEMENT(287);
executed
```

If the locks still appear in the show_locks utility, we can force remove the stale locks:

```
t=> SELECT RELEASE_DEFUNCT_LOCKS();
executed
```

**Warning:** This operation can cause some statements to fail on the specific worker on which they are queued. This is intended as a "last resort" to solve stale locks.

## 15.6 Log Related Issues

The **Log Related Issues** page describes how to resolve the following common issues:

### 15.6.1 Loading Logs with Foreign Tables

Assuming logs are stored at `/home/rhendricks/sqream_storage/logs/`, a database administrator can access the logs using the external_tables concept through SQream DB.

```
CREATE FOREIGN TABLE logs
(
  start_marker      TEXT(4),
  row_id            BIGINT,
  timestamp         DATETIME,
  message_level     TEXT,
  thread_id         TEXT,
  worker_hostname   TEXT,
  worker_port       INT,
  connection_id     INT,
  database_name     TEXT,
  user_name         TEXT,
  statement_id      INT,
  service_name      TEXT,
  message_type_id   INT,
  message           TEXT,
  end_message       TEXT(5)
)
WRAPPER csv_fdw
OPTIONS
  (
     LOCATION = '/home/rhendricks/sqream_storage/logs/**/sqream*.log',
     DELIMITER = '|',
     CONTINUE_ON_ERROR = true
  )
;
```

For more information, see Loading Logs with Foreign Tables.

### 15.6.2 Counting Message Types

```
t=> SELECT message_type_id, COUNT(*) FROM logs GROUP BY 1;
message_type_id | count
----------------+----------
              0 |         9
              1 |      5578
              4 |      2319
             10 |      2788
             20 |       549
             30 |       411
             31 |      1720
             32 |      1720
            100 |      2592
            101 |      2598
            110 |      2571
            200 |        11
            500 |       136
           1000 |        19
           1003 |        19
           1004 |        19
           1010 |         5
```

### 15.6.3 Finding Fatal Errors

```
t=> SELECT message FROM logs WHERE message_type_id=1010;
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/leveldb/LOCK: Resource temporarily unavailable
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/leveldb/LOCK: Resource temporarily unavailable
Mismatch in storage version, upgrade is needed,Storage version: 25, Server version␣
↪is: 26
Mismatch in storage version, upgrade is needed,Storage version: 25, Server version␣
↪is: 26
Internal Runtime Error,open cluster metadata database:IO error: lock /home/rhendricks/
↪sqream_storage/LOCK: Resource temporarily unavailable
```

### 15.6.4 Countng Error Events Within a Certain Timeframe

```
t=> SELECT message_type_id,
.          COUNT(*)
.    FROM logs
.    WHERE message_type_id IN (1010,500)
.    AND timestamp BETWEEN '2019-12-20' AND '2020-01-01'
.    GROUP BY 1;
message_type_id | count
----------------+------
            500 |    18
           1010 |     3
```

### 15.6.5 Tracing Errors to Find Offending Statements

If we know an error occured, but don't know which statement caused it, we can find it using the connection ID and statement ID.

```
t=> SELECT connection_id, statement_id, message
.     FROM logs
.     WHERE message_level = 'ERROR'
.     AND timestamp BETWEEN '2020-01-01' AND '2020-01-06';
connection_id | statement_id | message
--------------+--------------+----------------------------------------------------
↪---------------------------------------------------------------------------------
↪----------------------
          79 |           67 | Column type mismatch, expected UByte, got INT64 on␣
↪column Number, file name: /home/sqream/nba.parquet
```

Use the `connection_id` and `statement_id` to narrow down the results.

```
t=>   SELECT database_name, message FROM logs
.       WHERE connection_id=79 AND statement_id=67 AND message_type_id=1;
database_name | message
--------------+--------------------------
master        | Query before parsing
master        | SELECT * FROM nba_parquet
```

## 15.7 Core Dumping Related Issues

The **Core Dumping Related Issues** page describes the troubleshooting procedure to be followed if all parameters have been configured correctly, but the cores have not been created.

**To troubleshoot core dumping:**

1. Reboot the server.
2. Verify that you have folder permissions:

```
$ sudo chmod -R 777 /tmp/core_dumps
```

3. Verify that the limits have been set correctly:

```
$ ulimit -c
```

If all parameters have been configured correctly, the correct output is:

```
$ unlimited
```

4. If all parameters have been configured correctly, but running **ulimit -c** outputs **0**, run the following:

```
$ sudo vim /etc/profile
```

5. Search for line and tag it with the **hash** symbol:

```
$ ulimit -S -c 0 > /dev/null 2>&1
```

6. If the line is not found in **/etc/profile** directory, do the following:

   a. Run the following command:

```
$ sudo vim /etc/init.d/functions
```

b. Search for the following:

```
$ ulimit -S -c ${DAEMON_COREFILE_LIMIT:-0} >/dev/null 2>&1
```

c. If the line is found, tag it with the **hash** symbol and reboot the server.

## 15.8 Gathering Information for SQream Support

SQream Support is ready to answer any questions, and help solve any issues with SQream DB.

### 15.8.1 Getting Support and Reporting Bugs

When contacting SQream Support, we recommend reporting the following information:

- What is the problem encountered?
- What was the expected outcome?
- How can SQream reproduce the issue?

When possible, please attach as many of the following:

- Error messages or result outputs
- DDL and queries that reproduce the issue
- *Log files*
- Screen captures if relevant

### 15.8.2 How SQream Debugs Issues

#### 15.8.2.1 Reproduce

If we are able to easily reproduce your issue in our testing lab, this greatly improves the speed at which we can fix it.

Reproducing an issue consists of understanding:

1. What was SQream DB doing at the time?
2. How is the SQream DB cluster configured?
3. How does the schema look?
4. What is the query or statement that exposed the problem?
5. Were there any external factors? (e.g. Network disconnection, hardware failure, etc.)

See the *Collecting a Reproducible Example of a Problematic Statement* section ahead for information about collecting a full reproducible example.

#### 15.8.2.2 Logs

The logs produced by SQream DB contain a lot of information that may be useful for debugging.

Look for *error messages in the log and the offending statements*. SQream's support staff are experienced in correlating logs to workloads, and finding possible problems.

See the *Collecting Logs and Metadata Database* section ahead for information about collecting a set of logs that can be analyzed by SQream support.

#### 15.8.2.3 Fix

Once we have a fix, this can be issued as a hotfix to an existing version, or as part of a bigger major release.

Your SQream account manager will keep you up-to-date about the status of the issue.

### 15.8.3 Collecting a Reproducible Example of a Problematic Statement

SQream DB contains an SQL utility that can help SQream support reproduce a problem with a query or statement.

This utility compiles and executes a statement, and collects the relevant data in a small database which can be used to recreate and investigate the issue.

#### 15.8.3.1 SQL Syntax

```
SELECT EXPORT_REPRODUCIBLE_SAMPLE(output_path, query_stmt [, ... ])
;

output_path ::=
   filepath
```

#### 15.8.3.2 Parameters

| Parameter | Description |
|---|---|
| `output_path` | Path for the output archive. The output file will be a tarball. |
| `query_stmt [, ...]` | Statements to analyze. |

#### 15.8.3.3 Example

```
SELECT EXPORT_REPRODUCIBLE_SAMPLE('/home/rhendricks', 'SELECT * FROM t', $$SELECT
↪"Name", "Team" FROM nba$$);
```

## 15.8.4 Collecting Logs and Metadata Database

SQream DB comes bundled with a data collection utility and an SQL utility intended for collecting logs and additional information that can help SQream support drill down into possible issues.

See more information in the *Collect logs from your cluster* section of the *Logging* guide.

### 15.8.4.1 Examples

Write an archive to `/home/rhendricks`, containing log files:

```
SELECT REPORT_COLLECTION('/home/rhendricks', 'log')
;
```

Write an archive to `/home/rhendricks`, containing log files and metadata database:

```
SELECT REPORT_COLLECTION('/home/rhendricks', 'db_and_log')
;
```

## 15.8.5 Using the Command Line Utility:

```
$ ./bin/report_collection /home/rhendricks/sqream_storage /home/rhendricks db_and_log
```

# CHAPTER SIXTEEN

# GLOSSARY

The following table shows the **Glossary** descriptions:

| Term | Description |
|---|---|
| Authentication | The process of verifying identity by validating a user or role identity using a username and password. |
| Authorization | Defines the set of actions that an authenticaed role can perform after gaining access to the system. |
| Catalog | A set of views containing metadata information about objects in a database. |
| Cluster | A SQream deployment containing several workers running on one or more nodes. |
| Custom connector | When SQream is integrated with Power BI, used for running direct queries. |
| Direct query | A Power BI data extraction method that retrieves data from a remote source instead of from a local repository. |
| Import | A Power BI data extraction method that retrieves data to local repository to be visualized at a later point. |
| Metadata | SQream's internal storage containing details about database objects. |
| Node | A machine used to run SQream workers. |
| Role | A group or a user. For more information see SQream Studio. |
| Storage cluster | The directory where SQream stores data. |
| Worker | A SQream application that responds to statements. Several workers running on one or more nodes form a cluster. |