Load Tester User Guide

October 8, 2020

Revision History

Version	Date	Author	Comments
1.0	11/14/2019	Deane Harding and Mark Hoerth	Initial release
1.1	11/18/2019	Deane Harding	Introduction of num_executors parameter and new analysis VDSs
1.2	01/06/2020	Deane Harding	Updated document format. Introduced new test_description parameter and automatic VDS creation functionality
1.3	02/10/2020	Deane Harding	Updated with new details for querycollector process.
1.4	02/13/2020	Deane Harding	Revised querycollector instructions
1.5	03/18/2020	Deane Harding	Update to include SSL setup steps
1.6	05/22/2020	Deane Harding	Minor update to queries.json scrubbing process.
1.7	09/22/2020	Deane Harding	Removed JMeter folder from VDS hierarchy
1.8	10/08/2020	Deane Harding	Removal of jar files for vds creation scripts. Now uses python instead.

Related Documents

Name	Version
semantic_layer_best_practices.pdf	1.0, September 2019

Supported Versions

Name	Version
Dremio	3.2.8 or later
Apache JMeter	5.1.1 or later
Java	1.8
Python	3.5 or later

Table of Contents

Introduction

Purpose

Dremio Load Tester enables organizations to understand and predict the performance of their cluster under current and future production workloads. Solution architects can easily understand when user demand will require system configuration changes or additional resources, and how to apply reflections to optimize their users' experience.

Using Dremio Load Tester, both the cache of input queries and the user concurrency are completely adjustable, giving system engineers the opportunity to model the results of what-if scenarios and new use cases with a high level of accuracy. Dremio Load Tester leverages Apache JMeter to measure individual query performance and then uses Dremio itself to drill down into the execution time of each query.

Architecture

The Dremio Load Tester is designed to be installed and executed on a standalone workstation or VM which has direct JDBC query access to the Dremio Coordinator host (or via a load balancer) on port 31010 and has access to the Dremio REST API on the same Coordinator host (or via a load balancer) via port 9047.

Users create a set of test queries that they want to execute against Dremio. When a test is executed, the first thing that happens is the set of test queries are placed into a circular queue, which is implemented as a sorted array. Each concurrent user defined in the test is represented as its own thread, which starts reading and sending queries to Dremio at an offset from the beginning of the sorted array. The offset (`k`) and subsequently the full set of queries that any particular thread will execute (`thisThreadFiles`) is derived from the following calculation:

```
for (int j=0; j < thisThreadFiles.length; j++) {
      k = (j + (spacing * (threadNum - 1))) % sourceFileCount;
      thisThreadFiles[j] = sqlFiles[k].getCanonicalFile();
}
```

Where spacing is defined as:

```
spacing = number of queries in test set / number of concurrent users in test
```

The fixed minimum value for the `spacing` is 3.

So for example if our test set had 100 queries (`sourceFileCount`=100) and we specified 10 concurrent users in our test then the value of `spacing` would be 10.

This means that thread 1 would start by reading and executing the query at array index 0, thread 2 starts at array index 10, thread 3 starts at array index 20 etc. Note that the circular queue is created with the modulus operator, we modulus the offset computation by the total length of the queue, which causes us to loop back around to the beginning of the array if we run off the end.

There are two fundamental reasons why the Load Tester is written this way:

1. Each concurrent user (thread) starts at a different position in the set of test queries, meaning we don't have many copies of the same query running all at the same time and we maintain a more even spread of executions across the entire set of test queries.
2. Repeatability. If the same test, with the same number of concurrent users and number of queries, were to be repeated over and over, then every thread will submit the same queries in the same order each time, meaning that results from multiple identical test are comparable between test runs. This is important when we start to tweak other factors such as queue concurrency limits, or the number of executors in the Dremio cluster.

Usage Patterns

The tool can be used to issue a sequential series of queries, one at a time, to enable benchmark analysis of the test queries. It can also be used to run sets of queries in parallel, simulating multiple concurrent users issuing queries simultaneously over a period of time and thus performing more traditional load tests against Dremio.

For every set of tests that is performed, it is important to note the Dremio configuration for those tests:

- Number of executors
- Direct and heap memory settings on all coordinators and executors
- Number of CPUs allocated to coordinator and executor nodes
- Total available memory and CPUs available to coordinator and executor nodes
- Queue concurrency limits

Benchmarking

For benchmark tests where we want to perform sequential runs of queries, the number of concurrent users defined in the test is set to 1 and the tool will always generate a single thread to process queries from the beginning of the sorted array of queries.

The number of queries that this one user will execute can be set to any number, however as a best practice Dremio recommends making this a multiple of the number of test queries that are available, so each query runs a fixed number of times and we can perform statistical analysis on the results (e.g. mean, max, min, standard deviation execution times per query).

Load Testing

For load test analysis, where the number of concurrent users is greater than one, each concurrent user will execute the same number of queries, however as described above, each user will start executing queries at a different offset position within the set of test queries, therefore simulating a mixed workload of queries running simultaneously, also ensuring the most complete coverage of all the queries in the test set.

When performing load tests, Dremio recommends always making the first test a benchmark test. Run each query in the test set sequentially 3 times to obtain a representative baseline of how long each query takes to execute when the system isn't under stress.

From there Dremio recommends that the level of concurrency at which queries are sent into Dremio from the Load Tester (simulating a number of concurrent users) is steadily increased on each subsequent test run, thus exerting increasing load onto the Dremio cluster. For each concurrency test the aim is to execute every query in the test set approximately 5 times, therefore Dremio recommends the number of queries fired into Dremio for each concurrent user in the Dremio Load Tester is calculated approximately as:

*Queries Per User = (Total Number of Queries in Test Set * 5) / Number of Concurrent Users*

This calculation needs to be adjusted to run each query more times if *Number of Concurrent Users* is greater than the *Total Number of Queries in Test Set*.

Monitoring

Dremio thoroughly recommends gathering CPU, IO and memory statistics for all coordinator and executor nodes in the cluster at 5 second intervals throughout the duration of every set of tests. This allows us to analyze trends in these statistics over time and enables us to correlate spikes in these statistics with details of which queries were being executed at that same time the spikes occurred.

Setup and Configuration

Prepare the test queries

Dremio Load Tester accepts an input list of queries to drive testing. The input queries should represent current and anticipated production workloads.

- Each SQL test query should be in a separate file containing only the query
- Each query should use fully-qualified references to existing PDS and VDS data sets on the Dremio cluster.
- Each PDS and VDS should be visible to the user configured to run the tests.
- The queries must not be terminated with a semicolon

Each test query file must be named `<number>.sql`. Dremio recommends using a naming scheme that organizes the test queries into sortable sets that also allows for future growth.

- Generally, a 3-digit numerical filename works well. This allows solution architects or test engineers to create several different query groups of up to 100 queries in each group. For example:
    - `000.sql to 099.sql`, the first query group
    - `100.sql to 199.sql`, the second query group
    - Etc., additional groups
- Additional groups of queries can be added in the future by incrementing the first digit and then numbering queries within the group from 00 to 99.

Apache JMeter Considerations

Dremio Load Tester users Apache JMeter to measure query performance. Apache JMeter attempts to store the entire result set for each query in memory until the query is complete. As a result, JMeter may crash with `Out Of Memory` errors when large data sets are returned. To mitigate this risk, Dremio Load Tester automatically wraps each result set in the query test list with a SQL `LIMIT 1000` clause to only return the first 1000 records back to JMeter.

TODO – Put in here a description of why even 1000 may be too large and where this can be edited if lots of queries are returning 1000 rows simultaneously during high concurrency tests

Prepare the host VM

Select a dedicated Linux VM to host the Dremio Load Tester. This VM needs connectivity to send queries to the Dremio Coordinator (or to a Load Balancer connected to the Dremio Coordinator) via JDBC on port 31010, and it should contain at least 1GB of free memory for running the JRE.

- On the host VM, create a user and group called `dremio`. Other user and group names are acceptable, however running the load test tool as root is not recommended.

```
groupadd -r dremio
useradd -r -g dremio dremio -m
usermod -aG wheel dremio
passwd dremio
```

  Enter and confirm a password for the `dremio` user when prompted.

- Install Java 8 on the host VM

```
yum install java-1.8.0-openjdk
```

- Copy the Dremio JDBC driver from the Dremo Coordinator to the host VM, where
  - The driver `dremio-jdbc-driver-<version>.jar` is located on the Dremio Coordinator at `/opt/dremio/jars/jdbc-driver/`
  - Copy the driver to the `dremio` users home directory, `/home/dremio`

```
# copy the driver from the Dremio Coordinator
scp -i your-keyfile.pem your-username@<your-cluster-
IP>:/opt/dremio/jars/jdbc-driver/dremio-jdbc-driver-3.2.7-
201907041656560959-7f867fb.jar .
# copy it to the host system
scp -i your-keyfile.pem dremio-jdbc-driver-3.2.7-201907041656560959-
7f867fb.jar your-username@<your-VM-host>:/home/your-username
```

- Dremio Load Tester will log into the Dremio cluster and run the test queries. Create a file called `local.pwd` containing the password associated with the username when logging into the cluster during testing. This is not necessarily the same as the user running the load tester. Change the permissions on the file so it is only readable by the `dremio` user.

```
cd /home/dremio/
vi local.pwd
chmod 400 local.pwd
```

Install Dremio Load Tester

Dremio Load Tester is delivered as a compressed file called `dremio-load-tester.tar.gz`.

- Copy the tar file to the host VM
- Unpack the file into the `dremio` user's home directory

```
tar xvzf dremio-load-tester.tar.gz -C /home/dremio
```

- Download and install Apache JMeter into the Dremio Load Tester home folder

```
curl -o apache-jmeter-5.1.1.tgz
https://archive.apache.org/dist/jmeter/binaries/apache-jmeter-5.1.1.tgz

tar xvzf apache-jmeter-5.1.1.tgz -C /home/dremio/dremio-load-tester
```

- Copy the Dremio JDBC driver into Dremio Load Tester

```
cp /home/dremio/dremio-jdbc-driver-<version>.jar /home/dremio/dremio-
load-tester/apache-jmeter-5.1.1/lib/
```

- Update Apache JMeter Logging Jars

```
curl -o /home/dremio/dremio-load-tester/apache-jmeter-
5.1.1/lib/slf4j-simple-1.7.25.jar
https://repo1.maven.org/maven2/org/slf4j/slf4j-simple/1.7.25/slf4j-
simple-1.7.25.jar

curl -o /home/dremio/dremio-load-tester/apache-jmeter-
5.1.1/lib/log4j-to-slf4j-2.9.0.jar
https://repo1.maven.org/maven2/org/apache/logging/log4j/log4j-to-
slf4j/2.9.0/log4j-to-slf4j-2.9.0.jar

cd /home/dremio/dremio-load-tester/apache-jmeter-5.1.1/lib/
mv log4j-1.2-api-2.11.1.jar log4j-1.2-api-2.11.1.jar.bak
mv log4j-slf4j-impl-2.11.1.jar log4j-slf4j-impl-2.11.1.jar.bak
```

- Copy the prepared test query files (the individual `<number>.sql` files) into load tester at `/home/dremio/dremio-load-tester/queries/`
- Verify that all files in the `dremio-load-tester` folder are owned by the current user

```
cd /home/dremio
chown -R dremio:dremio dremio-load-tester
```

- Make sure the scripts we need to run have the necessary execution privileges

```
cd /home/dremio/dremio-load-tester/scripts
chmod -R 755 *
```

Copy querycollector onto Dremio Coordinator

There is one component of the Dremio Load Tester tool, called `querycollector`, which is responsible for pre-processing the results of the load tests and copying them to a designated cloud storage location prior to the results being consumed by Dremio. This component needs copying from the VM onto the Dremio Coordinator node itself.
The component that needs copying is located in the `/home/dremio/dremio-load-tester/scripts/querycollector` directory on the VM. It is recommended to tar compress the contents of this directory prior to copying it to the Dremio Coordinator and then decompress the file into a `/home/dremio` directory on the Dremio Coordinator.

- On the VM, issue the following commands to compress the contents of the `querycollector` directory:

```
cd /home/dremio/dremio-load-tester/scripts
tar -zcvf querycollector.tar
```

- Copy the resulting `querycollector.tar` file into `/home/dremio` on the Dremio Coordinator, then issue the following commands on the Dremio Coordinator:

```
cd /home/dremio/
tar -xvf querycollector.tar
chown -R dremio:dremio querycollector
chmod -R 755 querycollector
```

- The tool also requires python in order to successfully execute. Therefore ensure python is present on the Dremio Coordinator; first issue the following command to see if python is already installed.

```
python -V
```

If you are not presented with details of the currently installed python version then you will need to install Python. Version 3.5 or above is recommended.

- The following site provides an explanation if required:

  https://docs.python-guide.org/starting/installation/

- (optional) If you are going to be using querycollector to copy the test results files to Amazon S3, then you may need to install and configure the AWS CLI on the Dremio Coordinator. First issue the following command to see if the AWS CLI is already installed:

```
aws --version
```

If you are not presented with details of the currently installed AWS CLI version then use the following links as guidance.

- To install the AWS CLI, follow these instructions: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html

- To configure the AWS CLI, follow these instructions: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html#cli-quick-configuration

Configure Dremio Load Tester

The primary file that is used to execute the tests is called `run_tests.sh`. This file is located at `/home/dremio/dremio-load-tester/scripts/loadrunner/run_tests.sh` and has several parameters that need to be specified before tests can be executed.

Specify the following in `run_tests.sh`

Variable	Description
`dremio_host`	The hostname or IP address of the Dremio Coordinator. If the Dremio Coordinator is accessed by consumers via a Load Balancer then this will be the hostname or IP address of the Load Balancer.
`num_executors`	Indicator of the number of executors being used in the test. This is information for reporting purposes only and does not affect the actual Dremio configuration.
`test_description`	A single-line description of the test. This description MUST be placed inside double-quotes.
`query_dir`	The location of the queries for execution during the test, pre-configured to be `/home/dremio/dremio-load-tester/queries/`
`user_name`	The user that will execute the test queries on the cluster.
`pwd`	The password for the user `user_name` above read from `local.pwd` or embedded directly. This password will be used to log into the cluster and run queries.

The file `run_tests.sh` also includes one or more `kick_concurr.sh` commands. Each `kick_concurr.sh` line creates a test run with

- The desired concurrency (number of simultaneous users)
- The desired number of queries executed per user

Dremio recommends that the first test establish a baseline set of query statistics using 1 concurrent user and 3 executions of each provided query. See the example below.

Example

The `run_tests.sh` file below configures the first test run with the following parameters:

Desired concurrency = 1

Desired number of queries per user = 150

Resulting in the command

```
./kick_concurr.sh $dremio_host $user_name $pwd $query_dir $log_file
$num_executors "$test_description" 1 150
```

The behavior of this command will depend on the number of test queries copied into the query directory.

- Assuming there are 50 queries in the query directory, each query would be run 3 times
- Assuming there are 300 queries in the query directory, only the first 150 queries would each be run once

The example also shows several additional test runs with increasing concurrency and differing queries per user values.

There is no restriction on the number of tests that can be specified in `run_tests.sh`.

```
# /home/dremio/dremio-load-tester/scripts/loadrunner/run_tests.sh

# hostname or IP address of the Dremio Coordinator
dremio_host=192.168.0.21

# Indicator of number of executors used in the test. Does not alter actual Dremio
configuration.
num_executors=4

#test_description must be a single-line description inside double-quotes
test_description="test"

# The location of individual query files
query_dir="../../queries"

# the username for query execution
user_name="dremio"

# Content of the local.pwd in the home directory of the user
# or the password can be entered explicitly in this file
pwd=$(cat ../../../local.pwd)

# Capture output of script - DO NOT EDIT
NOW=$(date +%Y%m%d%H%M%S)
log_file="../../logs/dremio-load-tester-$NOW.log"

# Test execution commands
echo "1 user, 150 queries"
./kick_concurr.sh $dremio_host $user_name $pwd $query_dir $log_file $num_executors 1
150
echo "==================================="

echo "5 users, 50 queries each"
./kick_concurr.sh $dremio_host $user_name $pwd $query_dir $log_file $num_executors 5
50
echo "==================================="

echo "10 users, 25 queries each"
./kick_concurr.sh $dremio_host $user_name $pwd $query_dir $log_file $num_executors
10 25
echo "==================================="

echo "20 users, 13 queries each"
./kick_concurr.sh $dremio_host $user_name $pwd $query_dir $log_file $num_executors
20 13
echo "==================================="

echo "30 users, 9 queries each"
./kick_concurr.sh $dremio_host $user_name $pwd $query_dir $log_file $num_executors
30 9
echo "==================================="
```

Example run_tests.sh

SSL Considerations

If you have a requirement to connect to Dremio securely then there are some additional steps to ensure the JDBC connection to Dremio is configured with the required SSL parameters. To successfully set up the SSL connection you need to ensure you have a copy of a trustStore

present on the same machine as the Dremio Load Tester that contains the certificate chain required to verify that we are communicating with Dremio. You also need the password that protects this file.

- Using a text editor, open the file `/home/dremio/dremio-load-tester/scripts/loadrunner/concurrent_repeatable.jmx`
- Search for `jdbc:dremio:direct` in this file in order to locate the Dremio JDBC Connection String. You should find a line that looks like the following:

  ```
  <stringProp
  name="dbUrl">jdbc:dremio:direct=${__P(dremio_host)}:31010</stringProp>
  ```

- Alter the connection string inside the property so that it looks like the following:

  ```
  jdbc:dremio:direct=${__P(dremio_host)}:31010;schema=Dremio;ssl=true;trustStore=
  </path/to/truststore.jks>;trustStorePassword=<password_for_truststore>
  ```

  Here you can see we need to set the ssl property to true, we set the trustStore property to point to the location of the trustStore jks file and we also set the trustStorePassword property to the password required to access the truststore.

Execution

To run the tests that have been configured in `run_tests.sh`

```
# Change user if necessary
su - dremio
cd /home/dremio/dremio-load-tester/scripts/loadrunner/
./run_tests.sh
```

A log file called `dremio-load-tester-<YYYYMMDDHH24MISS>.log` will be written into the `/home/dremio/dremio-load-tester/logs` directory. A summary of all query executions will also be written to a `summary.csv` file in the same directory.
Test execution progress is written to the console as well as the log file.

Results and Analysis

Results pre-processing

All the result data is stored in `queries.json` on the Dremio Coordinator. During the installation and setup phase of the Dremio Load Tester we installed the `querycollector` component of the tool on the Dremio Coordinator, this component is responsible for cleaning up ("scrubbing") the `queries.json` file prior to its consumption in Dremio, ensuring rules regarding lengths of data fields when they are imported into Dremio are adhered to. The `querycollector` component is also responsible for copying the scrubbed `queries.json` files into a user-defined cloud storage container (S3, ADLS, HDFS) that is accessible to Dremio in order to analyze the data. The script assumes that there will be two folders available in the designated storage location to store the scrubbed `queries.json` files, one called "`results`" and another called "`chunks`".

The driving script of the `querycollector` is called `gather_queries.sh`. Prior to executing the script, you must ensure the following variable inside the script is specified correctly:

Variable	Description
`DREMIO_LOG_DIR`	The directory on the Dremio Coordinator where all the Dremio log files are stored. Default value is "`/var/log/dremio`"

At execution time, `gather_queries.sh`, requires the following three parameters as inputs on the command line:

Parameter	Description
`storage_type`	The type of storage where we want to write the scrubbed `queries.json` file to. Valid values are currently **`s3, adls, hdfs`**. If no valid value is supplied, then the scrubbed files will remain on the Dremio Coordinator and will need to be manually copied to a storage location.
`storage_path`	The path on the storage to a "results" folder e.g. `s3://mybucket`
`num_archive_days`	The number of days of archived `queries.json` files to also scrub and copy into storage. If no value is supplied, then only `queries.json` files for the current day will be scrubbed and copied.

The results are available for analysis after the following steps.

- On the Dremio Coordinator, execute `gather_queries.sh` with the desired input parameter values for your chosen storage type, storage location and number of historical `queries.json` files that you wish to process. This script reads the desired `queries.json` and `queries.YYYY-MM-DD.N.json.gz` files and prepares them for analysis. e.g.:

```
cd /home/dremio/querycollector
./gather_queries.sh s3 s3://mybucket/dremio 2
```

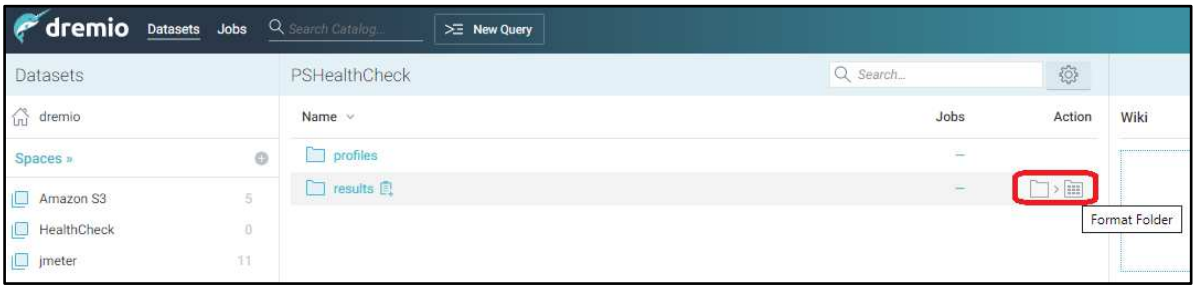
  `gather_queries.sh` copies the `queries.json` and `queries.YYYY-MM-DD.N.json.gz` files locally to the `querycollector` in `/home/dremio/querycollector/dremio_queries`, it then generates one or more files called `header.<filename>.json` and `chunks.<filename>.json` in the `/home/dremio/querycollector/dremio_queries/scrubbed/` directory. The scrubbed files then get automatically copied into the desired cloud storage location.

Results PDS

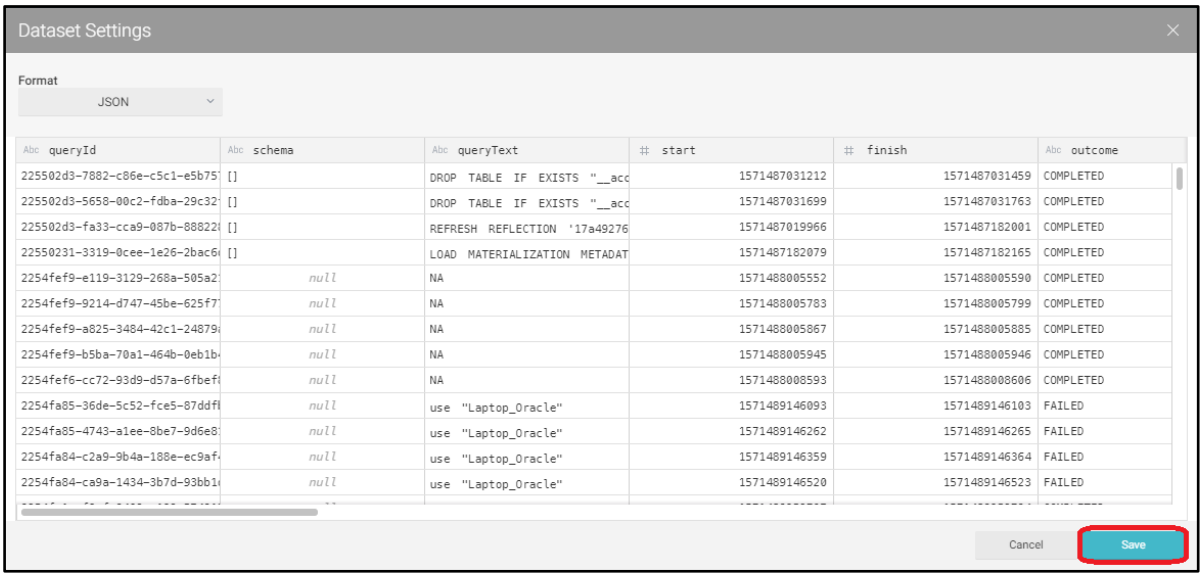
Configure a data source in Dremio to create a PDS which will drive the analysis

- In Dremio, create a new data source

- Call the data source `PSHealthCheck`.
- For the chosen data source, enter the relevant connection credentials.
- Navigate to the Advanced Options panel of the data source and enter the Root Path to be the **parent path** of the `results` folder on the storage device.
- Open the `PSHealthCheck` source. Next to the results folder, select the Format Folder option in the Actions column:



- In the resulting dialog, notice that Dremio automatically recognizes the format of the file inside this folder as JSON and has formatted the data table appropriately. Click Save.



Results VDSs

Create a set of VDSs that will be used to analyze the results of our tests. There are two approaches to this, automated VDS creation, or manual. Both options are described in the sections below. Dremio recommends the automated approach.

Automated VDS creation

The primary file that is used to automatically create the relevant spaces, folders and VDSs in Dremio is called `run_vdscreator.sh`. This file is located at `/home/dremio/dremio-load-tester/scripts/vdscreator/run_vdscreator.sh` and has several parameters that need to be specified before VDSs can be created.

- Specify the following in `run_vdscreator.sh`. All other parameters in the file must not be edited.

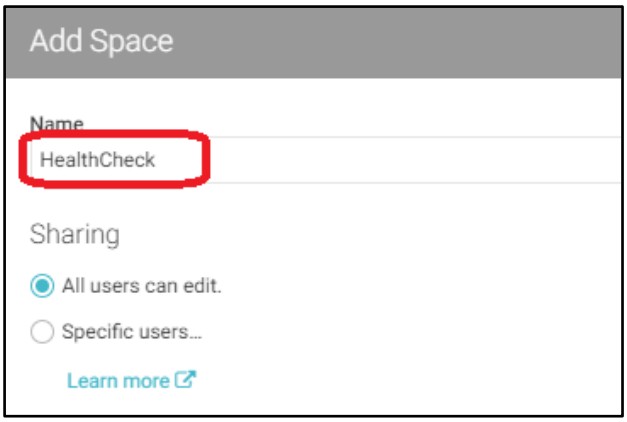
Variable	Description
`dremio_url`	Dremio UI URL for where the VDS definitions will be created e.g. http://localhost:9047
`user_name`	The name of an administrative user in Dremio that can create spaces, folders and VDSs.
`pwd`	The password for the user `user_name` above, read from `local.pwd` or embedded directly. This password will be used to log into Dremio via the REST API and issue calls to create the relevant objects.

- Execute `run_vdscreator.sh` to generate the required space, folders and VDSs in Dremio

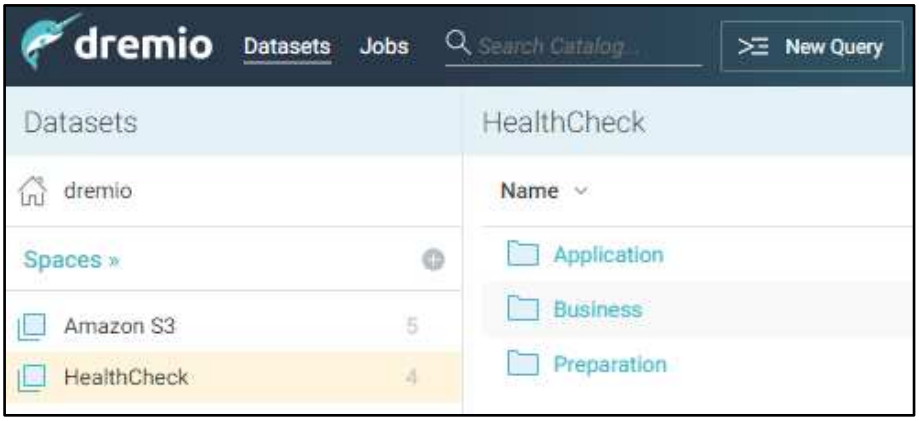
```
# Change user if necessary
su - dremio
cd /home/dremio/dremio-load-tester/scripts/vdscreator/
./run_vdscreator.sh
```

Manual VDS creation

- Create a new Space in Dremio called `HealthCheck`



- Inside the `HealthCheck` space create three new sub-folders: `Application, Business, and Preparation`



- On the Dremio Load Tester VM host, navigate to the `vdsdefinition` directory:

```
cd /home/dremio/dremio-load-tester/vdsdefinition
```

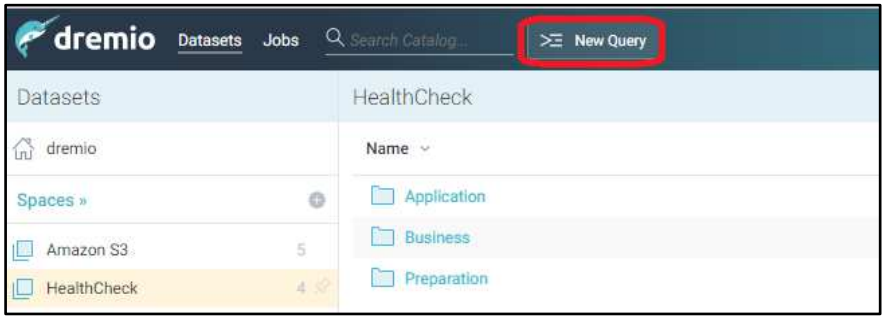
This directory contains the following `sql` files:

```
01_HealthCheck.Preparation.results.sql
02_HealthCheck.Business.TestQueryData.sql
03_HealthCheck.Application.LoadTestResults.sql
04_HealthCheck.Application.LoadTestResultsSummary.sql
05_HealthCheck.Application.LoadTestDescription.sql
06_HealthCheck.Application.LoadTestQueueUsage.sql
07_HealthCheck.Application.LoadTestPercentHighCostQueries.sql
08_HealthCheck.Application.LoadTestPercentAccelerated.sql
09_HealthCheck.Application.LoadTestFailedQueries.sql
10_HealthCheck.Application.LoadTestAvgQueueEnqueuedTime.sql
```

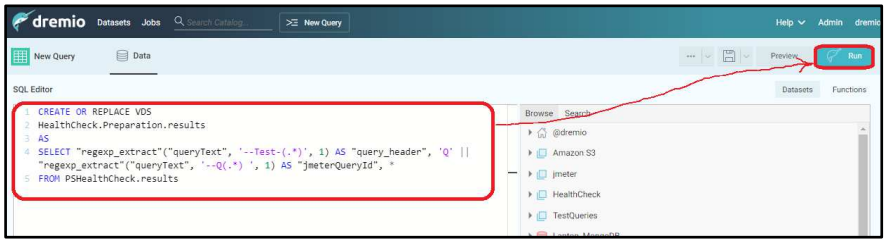
For each of the files above in numeric order do the following:

- Copy the SQL out of the file

- In the Dremio UI, click on New Query



- Paste the SQL into the SQL Editor and click Run. This will create the VDS.



Analyze Results

Dremio is now available to query the `HealthCheck.Application.LoadTestResults` VDS and `HealthCheck.Application.LoadTestResultsSummary` VDS to obtain execution timings for all of the queries executed during the test runs. `LoadTestResults` is the sanitized view of the results, `LoadTestResultsSummary` gives us an aggregated summary view of execution times of each query for each test.

Getting Support

Dremio Load Tester is provided by Dremio Professional Services for use in the context of a Dremio Professional Services engagement. Issues should be directed to Dremio Professional Services, who provide support during the engagement on a best-effort basis.