# 

[**Preface**](#_u9uc2sq1qxqp) **1**

[**Technical Requirements**](#_y063puuoyl4r) **1**

[JMX to Prometheus exporter](#_az56b81qxf4j) 2

[Download the JAR](#_qe3xmg7ampbm) 2

[Configure the coordinators](#_tqbiagg4i52s) 2

[Restart the coordinators](#_jhl3jbun4bnf) 2

[Prometheus Push Gateway](#_g4so4kocc8rl) 2

[Download the Package](#_pvl6mrom5t33) 2

[Configure Push Gateway](#_81h9sec16suv) 3

[Prometheus](#_g6vpiibpg6th) 4

[Download the Package](#_gb58tviptoia) 4

[Configure Prometheus](#_ulsmonl2ecko) 4

[Grafana](#_ngcuqod347fg) 6

[Configure the YUM repo](#_vgsj4wmghtn7) 6

[Install Grafana](#_6uew2s47of01) 7

[Start the service](#_evdj27r7a9px) 7

[Prometheus Python Client](#_j72efcto0gn4) 7

[Install](#_441flakalv3v) 7

# Preface

Standard Chartered Bank (SCB) wants to monitor multiple Dremio clusters that will be deployed in the System integration testing (SIT) environment, running on the Hadoop clusters in YARN deployment mode. This document details the steps that need to be taken to stand up the tech stack to enable the solution provided by Dremio. Solution itself is detailed in another document, which will be shared with SCB.

# Technical Requirements

Monitoring solution will use JMX to Prometheus exporter to export metrics exposed by JMX interface on the Dremio coordinator node. For the metrics that are not available on JMX interface, we propose to push them using Prometheus Push Gateway. These metrics will be generated using Dremio API and SQL queries against Dremio’s system tables.

## JMX to Prometheus exporter

This module exposes Dremio JMX metrics, so they’re available for scraping by Prometheus. This exporter is intended to be run as a Java Agent, exposing a HTTP server and serving metrics of the local JVM.

### Download the JAR

Download the latest jar available [here](https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.12.0/jmx_prometheus_javaagent-0.12.0.jar) and copy it to a location on both active and standby coordinators in a HA setup. We recommend copying this to <DREMIO\_INSTALL\_DIR>/jars/3rdparty location. This is typically /opt/dremio/jars/3rdparty.

### Configure the coordinators

We’ll need to configure both active and standby coordinator nodes (in a HA setup). Edit DREMIO\_JAVA\_EXTRA\_OPTS option in the <DREMIO\_CONF\_DIR>/dremio-env file. This is typically /opt/dremio/dremio-env.

# Extra Java options - shared between dremio and dremio-admin commands

#

# Add JMX to Prometheus exporter

DREMIO\_JAVA\_EXTRA\_OPTS=-javaagent:/opt/dremio/jars/3rdparty/jmx\_prometheus\_javaagent-0.12.0.jar=8080:/opt/dremio/conf/jmx\_config.yaml

Note that /opt/dremio/conf/jmx\_config.yaml is an empty file, since we don’t have any custom configuration for the exporter.

### Restart the coordinators

Restart both Active and Standby coordinators (in a HA setup). JMX metrics for both of them should be available at port 8080. You may verify it by running curl command like so: curl <HOST\_NAME>:8080

## Prometheus Push Gateway

Pushgateway is the mechanism that we use to ingest metrics from API/JDBC sources into Prometheus.

### Download the Package

Download the latest version of Push Gateway from the [downloads](https://prometheus.io/download/#pushgateway) page. For example, run the following command to download the latest version.

wget <https://github.com/prometheus/pushgateway/releases/download/v1.2.0/pushgateway-1.2.0.linux-amd64.tar.gz>

### Configure Push Gateway

Push Gateway is not a resource intensive application. We recommend running it on the same host as Prometheus.

* Create a user

sudo useradd --no-create-home --shell /bin/false prometheus

* Create needed directories

sudo mkdir -p /var/local/pushgateway

sudo chown prometheus:prometheus /var/local/pushgateway

* Extract the tar and copy required files

tar xvzf pushgateway-1.2.0.linux-amd64.tar.gz

cd pushgateway-1.2.0.linux-amd64

sudo cp pushgateway /usr/local/bin

sudo chown prometheus:prometheus /usr/local/bin/pushgateway

* Create a systemd service file.

$ cat /etc/systemd/system/pushgateway.service

[Unit]

Description=Prometheus Pushgateway

[Service]

Type=Simple

User=prometheus

Group=prometheus

ExecStart=/usr/local/bin/pushgateway --web.listen-address=:9091 --web.telemetry-path=/metrics --persistence.file=/var/local/pushgateway/pushgateway.store --web.enable-admin-api

ExecReload=/bin/kill -HUP $MAINPID

KillMode=process

Restart=on-failure

[Install]

WantedBy=multi-user.target

* Start the service

sudo systemctl daemon-reload

sudo systemctl restart pushgateway

systemctl status pushgateway - Should show that process is running

* Verify that Push gateway is working by issuing curl <host>:9091/metrics on the command line. You should see metrics for the push gateway process.

Note that we are enabling the Admin API for Push Gateway. This needs to be used by Admins to maintain metrics, when the source of metric disappears. This is well documented in Push Gateway [documentation](https://prometheus.io/docs/practices/pushing/). It’s possible to automate the cleanup process though.

Here’s one example of when the cleanup needs to be run. Let’s say we have 2 sub clusters called mycluster1 and mycluster2 running on Dremio. At some point, mycluster2 gets deleted (or renamed). However, push gateway will continue to expose the metric for the sub cluster. We will then need to invoke the Admin API to delete the metric like so:

curl -X DELETE http://<push gateway>:9091/metrics/job/dremioCluster1/cluster/mycluster2

## Prometheus

[Prometheus](https://github.com/prometheus) is an open-source systems monitoring and alerting toolkit. We recommend running Prometheus along with Push Gateway and Grafana on the same host. A 2CPU, 8GB host should be good enough to run all three.

### Download the Package

Download the latest version of Prometheus from the [downloads](https://prometheus.io/download/#prometheus) page. For example, run the following command to download the latest version.

wget <https://github.com/prometheus/prometheus/releases/download/v2.18.0-rc.0/prometheus-2.18.0-rc.0.linux-amd64.tar.gz>

### Configure Prometheus

* Create a user

sudo useradd --no-create-home --shell /bin/false prometheus

* Create needed directories

sudo mkdir -p /etc/prometheus /var/local/prometheus

sudo chown prometheus:prometheus /etc/prometheus /var/local/prometheus

* Extract the tar and copy required files

tar xvzf prometheus-2.18.0-rc.0.linux-amd64.tar.gz

cd prometheus-2.18.0-rc.0.linux-amd64

sudo cp prometheus /usr/local/bin

sudo cp promtool /usr/local/bin

sudo chown prometheus:prometheus /usr/local/bin/prometheus

sudo chown prometheus:prometheus /usr/local/bin/promtool

sudo cp -r consoles /etc/prometheus

sudo cp -r console\_libraries /etc/prometheus

sudo chown -R prometheus:prometheus /etc/prometheus/consoles

sudo chown -R prometheus:prometheus /etc/prometheus/console\_libraries

* Configure scrape settings

sudo touch /etc/prometheus/prometheus.yml

sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml

Here’s a sample configuration. Please note that it’s a best practice to set the job\_name

to the name of the cluster that’s getting monitored. Also note that we’re scraping metrics

from the push gateway.

global:

scrape\_interval: 5s

scrape\_configs:

- job\_name: "prometheus"

static\_configs:

- targets: ["localhost:9090"]

- job\_name: "mycluster1"

static\_configs:

- targets: ["<DREMIO COORDINATOR HOST1>:8080"]

- targets: ["<DREMIO COORDINATOR HOST2>:8080"]

- job\_name: "push"

honor\_labels: true

static\_configs:

- targets: ["<PUSH GATEWAY HOST>:9091"]

* Create a systemd service file. Location where data is stored should have enough space (with faster disks, ideally).

$ cat /etc/systemd/system/prometheus.service

[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/local/prometheus/ \

--web.console.templates=/etc/prometheus/consoles \

--web.console.libraries=/etc/prometheus/console\_libraries

[Install]

WantedBy=multi-user.target

* Start the service

sudo systemctl daemon-reload

sudo systemctl restart prometheus

systemctl status prometheus - Should show that process is running

* Verify that Prometheus Web UI is up and running, by visiting <host>:9090 on the browser.

## Grafana

Grafana is open source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored. We highly recommend to run this on the same host as Prometheus.

### Configure the YUM repo

Grafana is available in a YUM repository. Create a repository file /etc/yum.repos.d/grafana.repo with following contents.

[grafana]

name=grafana

baseurl=https://packages.grafana.com/oss/rpm

repo\_gpgcheck=1

enabled=1

gpgcheck=1

gpgkey=https://packages.grafana.com/gpg.key

sslverify=1

sslcacert=/etc/pki/tls/certs/ca-bundle.crt

### Install Grafana

Run sudo yum install grafana to install grafana.

### Start the service

There’s no need to configure grafana. The package installs systemd script. Run following commands to start the service and verify that it’s running.

sudo systemctl daemon-reload

sudo systemctl start grafana-server

sudo systemctl status grafana-server

Open <HOST>:3000 on the browser and you should see grafana Web UI.

## Prometheus Python Client

We use this to push metrics to Prometheus via Push Gateway. This is needed only on the host that runs the metric collection scripts.

### Install

This is installed via pip. Collection scripts make use of python3, so install it using the following command.

pip3 install prometheus\_client

## Python JDBC Module

The JayDeBeApi module allows you to connect from Python code to databases using Java JDBC. We need this to query Dremio.

### Install

This is installed via pip. Collection scripts make use of python3, so install it using the following command.

pip3 install jaydebeapi

This installs the latest version of the JPype module that provides full access to Java from within Python. Unfortunately, the latest version has [issues](https://github.com/baztian/jaydebeapi/issues/99) with the JDBC module. To fix the issue, downgrade JPype to 0.6.3 version. Note other dependencies that need to be installed as a privileged user.

sudo yum install python3-devel

sudo yum install gcc-c++

pip3 install JPype1==0.6.3 --force-reinstall