Prometheus Integration Guide

Couchbase
NoEQUAL

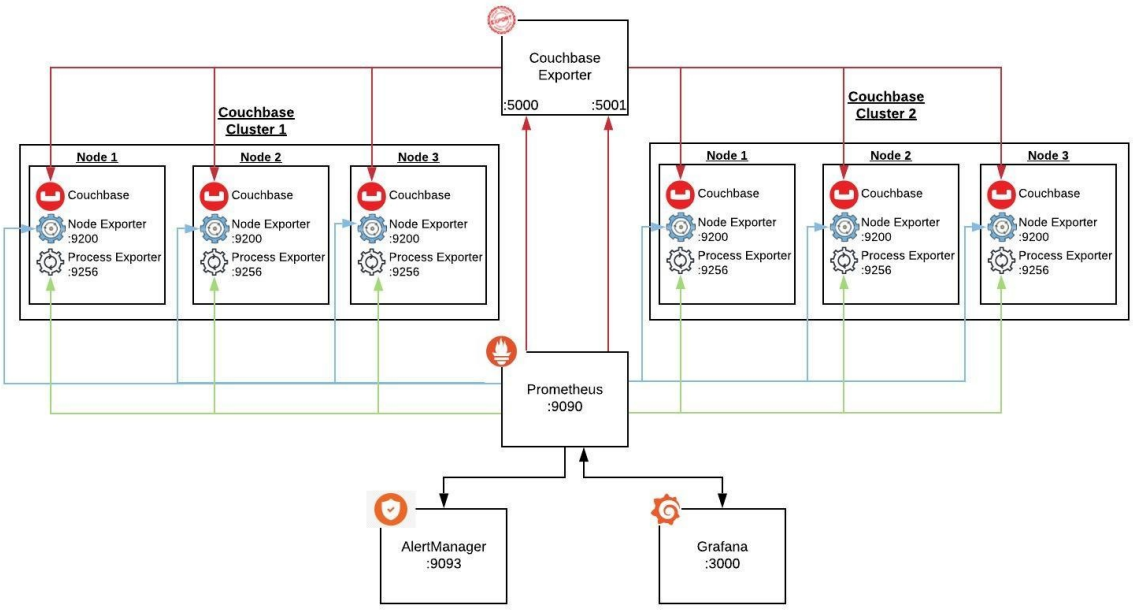
Prometheus Integration Guide

This document details how to setup and configure an external monitoring and alerting system for Couchbase using Prometheus and Grafana.

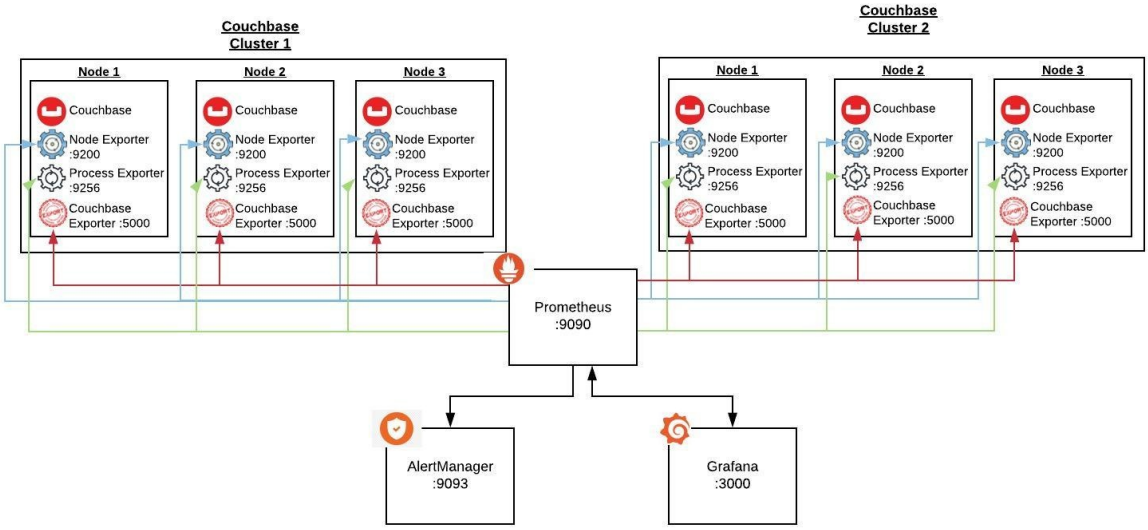
There are many different configurations that the monitoring solution can be deployed in, there is not a right or wrong way. You can colocate Prometheus, Grafana and the Couchbase Exporter or they can all be deployed to separate servers, both are perfectly acceptable. Ultimately, the configuration should be based on your use-case, storage and availability requirements.

The Couchbase Exporter can be deployed in a standalone cluster configuration, where there is a single Couchbase Exporter for each Couchbase Cluster that is being monitored. The Couchbase Exporter can also be deployed in a local mode, where each Couchbase Node has the Exporter installed. After completing the integration guide, your configuration will be very similar to one of the following:

Standalone / Cluster Mode



Local Mode



1. Install Prometheus

The following will walk you through how to install and configure Prometheus. This should NOT be installed on each Couchbase node or any node in the cluster. This should be a stand alone instance.

Step 1.1: Download

Download the Prometheus binary to the server that you will use for Prometheus.

```
wget \
  https://github.com/prometheus/prometheus/releases/download/v2.24.0/prometh
eus-2.24.0.linux-amd64.tar.gz
```

Visit the Prometheus downloads page for the latest version.

Step 1.2: Create User

Create a Prometheus user, required directories, and make prometheus user as the owner of those directories.

```
sudo groupadd -f prometheus
sudo useradd -g prometheus --no-create-home --shell /bin/false prometheus
sudo mkdir /etc/prometheus
sudo mkdir /var/lib/prometheus
sudo chown prometheus:prometheus /etc/prometheus
sudo chown prometheus:prometheus /var/lib/prometheus
```

Step 1.3: Unpack

Untar and move the downloaded Prometheus binary

```
tar -xvf prometheus-2.24.0.linux-amd64.tar.gz
mv prometheus-2.24.0.linux-amd64 prometheus-files
```

Step 1.4: Install Prometheus

Copy `prometheus` and `promtool` binary from `prometheus-files` folder to `/usr/bin` and change the ownership to prometheus user.

```
sudo cp prometheus-files/prometheus /usr/bin/
sudo cp prometheus-files/promtool /usr/bin/
sudo chown prometheus:prometheus /usr/bin/prometheus
sudo chown prometheus:prometheus /usr/bin/promtool
```

Step 1.5: Install Prometheus Libraries

Move the `prometheus.yml` , `consoles` and `console_libraries` directories from `prometheus-files` to `/etc/prometheus` folder and change the ownership to prometheus user.

```
sudo cp -r prometheus-files/consoles /etc/prometheus
```

```
sudo cp -r prometheus-files/console_libraries /etc/prometheus
sudo cp prometheus-files/prometheus.yml /etc/prometheus/prometheus.yml
sudo chown -R prometheus:prometheus /etc/prometheus/consoles
sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml
```

Step 1.6: Setup Service

Create a prometheus service file.

```
sudo vi /usr/lib/systemd/system/prometheus.service
```

Add the following configuration and save the file

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/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
```

```
sudo chmod 664 /usr/lib/systemd/system/prometheus.service
```

Note: Prometheus is configured to use `/var/lib/prometheus` as it's tsdb storage location, ensure there is enough space available.

Step 1.7: Reload systemd

Reload the `systemd` service to register the prometheus service and start the prometheus service.

```
sudo systemctl daemon-reload
sudo systemctl start prometheus
```

Check the prometheus service status using the following command.

```
sudo systemctl status prometheus
```

Configure Prometheus to start at boot

```
sudo systemctl enable prometheus.service
```

```
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2018-08-29 19:55:49 UTC; 2min 12s ago
 Main PID: 14789 (prometheus)
   CGroup: /system.slice/prometheus.service
           └─14789 /usr/local/bin/prometheus --config.file /etc/prometheus/prometheus.yml --storage.
```

If `firewalld` is enabled and running, add a rule for port `9090`

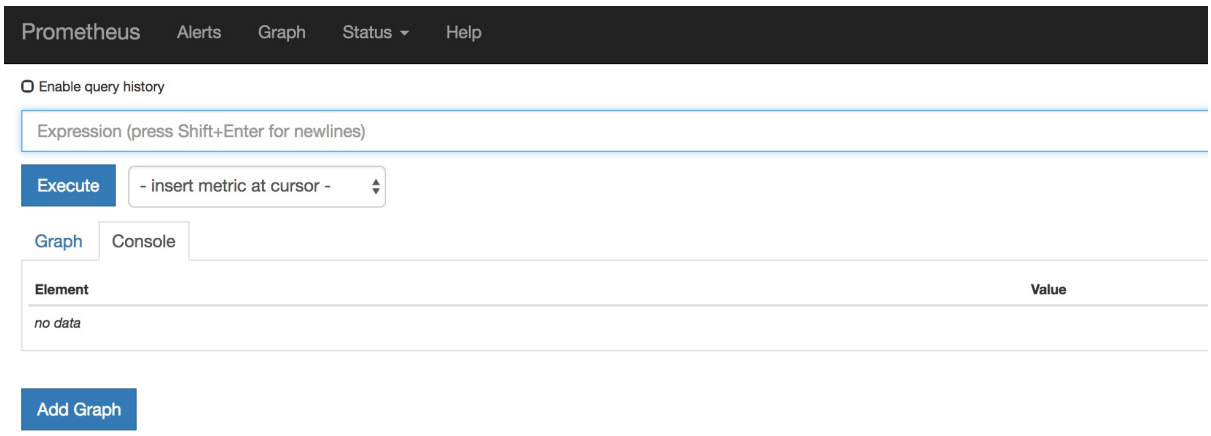
```
sudo firewall-cmd --permanent --zone=public --add-port=9090/tcp
sudo firewall-cmd --reload
```

Step 1.8: Access UI

Now you will be able to access the prometheus UI on `9090` port of the prometheus server.

```
http://<prometheus-ip>:9090/graph
```

You should be able to see the following UI as shown below.



Step 1.9: Clean Up

Remove the download and temporary files

```
rm -rf prometheus-2.24.0.linux-amd64.tar.gz prometheus-files
```

---

2. Node Exporter Setup

The Node Exporter is an agent that gathers system metrics and exposes them in a format which can be ingested by Prometheus. The Node Exporter is a project that is maintained through the Prometheus project. This is a completely optional step and can be skipped if you do not wish to gather system metrics. The following will need to be performed on each server that you wish to monitor system metrics for.

Step 2.1: Download

Download the Node Exporter binary to each Couchbase Server that you want to monitor. The Node Exporter will export system related stats.

```
wget \
  https://github.com/prometheus/node_exporter/releases/download/v1.0.1/node_
exporter-1.0.1.linux-amd64.tar.gz
```

Visit the Prometheus downloads page for the latest version.

Step 2.2: Create User

Create a Node Exporter user, required directories, and make prometheus user as the owner of those directories.

```
sudo groupadd -f node_exporter
sudo useradd -g node_exporter --no-create-home --shell /bin/false node_expor
ter
sudo mkdir /etc/node_exporter
sudo chown node_exporter:node_exporter /etc/node_exporter
```

Step 2.3: Unpack

Untar and move the downloaded Node Exporter binary

```
tar -xvf node_exporter-1.0.1.linux-amd64.tar.gz
```

```
mv node_exporter-1.0.1.linux-amd64 node_exporter-files
```

Step 2.4: Install Node Exporter

Copy `node_exporter` binary from `node_exporter-files` folder to `/usr/bin` and change the ownership to prometheus user.

```
sudo cp node_exporter-files/node_exporter /usr/bin/
sudo chown node_exporter:node_exporter /usr/bin/node_exporter
```

Step 2.5: Setup Service

Create a node_exporter service file.

```
sudo vi /usr/lib/systemd/system/node_exporter.service
```

Add the following configuration

```
[Unit]
Description=Node Exporter
Documentation=https://prometheus.io/docs/guides/node-exporter/
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
Restart=on-failure
ExecStart=/usr/bin/node_exporter \
  --web.listen-address=:9200

[Install]
WantedBy=multi-user.target
```

```
sudo chmod 664 /usr/lib/systemd/system/node_exporter.service
```

** Note: The default port for the `node_exporter` is actually `:9100` but that is the same port as the Couchbase Index Admin Port and cannot be used.

Step 2.7: Reload systemd and start Node Exporter

Reload the `systemd` service to register the prometheus service and start the prometheus service.

```
sudo systemctl daemon-reload
sudo systemctl start node_exporter
```

Check the node exporter service status using the following command.

```
sudo systemctl status node_exporter
```

```
● node_exporter.service - Node Exporter
   Loaded: loaded (/usr/lib/systemd/system/node_exporter.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2020-04-28 21:08:14 UTC; 17h ago
     Docs: https://prometheus.io/docs/guides/node-exporter/
 Main PID: 27646 (node_exporter)
   CGroup: /system.slice/node_exporter.service
           └─27646 /usr/local/bin/node_exporter --web.listen-address=:9200
```

Configure node_exporter to start at boot

```
sudo systemctl enable node_exporter.service
```

If `firewalld` is enabled and running, add a rule for port `9200`

```
sudo firewall-cmd --permanent --zone=public --add-port=9200/tcp
sudo firewall-cmd --reload
```

Step 2.8: Verify the Exporter is Running

Verify the exporter is running by visiting the `/metrics` endpoint on the node on port `9200`

```
http://<node_exporter-ip>:9200/metrics
```

You should be able to see something similar to the following:

```
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
```

```
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 7
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.12.5"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 919280
...
```

Step 2.9: Clean Up

Remove the download and temporary files

```
rm -rf node_exporter-1.0.1.linux-amd64.tar.gz node_exporter-files
```

---

3. Process Exporter Setup

The Process Exporter is an agent that gathers process specific metrics and exposes them in a format which can be ingested by Prometheus. This is a completely optional step and can be skipped if you do not wish to gather process metrics. The following will need to be performed on each server that you wish to monitor process metrics for.

Step 3.1: Download

Download the Process Exporter binary to each Couchbase Server that you want to monitor.

```
wget \
  https://github.com/ncabatoff/process-exporter/releases/download/v0.6.0/pro
cess-exporter-0.6.0.linux-amd64.tar.gz
```

Step 3.2: Create User

Create a Process Exporter user, required directories, and make prometheus user as the owner of those directories.

```
sudo groupadd -f process_exporter
sudo useradd -g process_exporter --no-create-home --shell /bin/false process
_exporter
sudo mkdir /etc/process_exporter
```

```
sudo chown process_exporter:process_exporter /etc/process_exporter
```

Step 3.3: Unpack

Untar and move the downloaded Process Exporter binary

```
tar -xvf process-exporter-0.6.0.linux-amd64.tar.gz
mv process-exporter-0.6.0.linux-amd64 process_exporter-files
```

Step 3.4: Install Process Exporter

Copy `process_exporter` binary from `process_exporter-files` folder to `/usr/bin` and change the ownership to prometheus user.

```
sudo cp process_exporter-files/process-exporter /usr/bin/
sudo chown process_exporter:process_exporter /usr/bin/process-exporter
```

Step 3.5: Create Process Exporter Configuration File

Create a `etc/process_exporter/process-exporter.yaml` with the following configuration.

```
sudo vi /etc/process_exporter/process-exporter.yaml
```

```
process_names:
  - comm:
    # Data service responsible for storing user data
    - memcached
    # Couchbase cluster manager run as Erlang virtual machines -
    # babysitter, ns_server, and ns_couchdb
    - beam.smp
    # Index service
    - indexer
    # Full-Text Search Service
    - cbft
    # Analytics Service
    - cbas
    # Couchbase Query service
    - cbq-engine
    # Extracts secondary key from documents
    - projector
    # Cross Data Center Replication (XDCR) - replicates data from one cluste
r to another
```

```
    - goxdcr
    # Utility in Go to get disk usage stats
    - godu
    # Process that acts as a bridge between ns_server (Erlang) and the other
    # server components (cbq- engine, cbft, etc.)
    - goport
    # Service that is used to encrypt the cluster configuration stored on di
sk
    - gosecrets
    # Erlang port process (wrapper) used to talk to the saslauthd daemon for
 authentication purposes
    - saslauthd-port
    # Erlang-specific process which acts as a name server for Erlang distrib
ution
    - epmd
    # Erlang-specific process used to collect CPU: 1 for ns_server VM and 1
for ns_couchdb VM
    - cpu_sup
    # Erlang-specific process used to collect memory usage: 1 for ns_server
VM
    # and 1 for ns_couchdb VM
    - memsup
    # Built-in Erlang port process that is used to perform name service look
up
    - inet_gethost
    # Open source tool sigar that is used to collect system information
    - portsigar
    # Eventing Service
    - eventing-produc
    # Eventing Service
    - eventing-consum
    - uwsgi
    - prometheus
    - alertmanager
    - grafana
```

Change the ownership to `process_exporter` user.

```
sudo chown process_exporter:process_exporter /etc/process_exporter/process-e
xporter.yaml
```

Step 3.6: Setup Service

Create a process_exporter service file.

```
sudo vi /usr/lib/systemd/system/process_exporter.service
```

Add the following configuration

```
[Unit]
Description=Process Exporter for Prometheus
Documentation=https://github.com/ncabatoff/process-exporter
Wants=network-online.target
After=network-online.target

[Service]
User=process_exporter
Group=process_exporter
Type=simple
Restart=on-failure
ExecStart=/usr/bin/process-exporter \
  --config.path /etc/process_exporter/process-exporter.yaml \
  --web.listen-address=:9256

[Install]
WantedBy=multi-user.target
```

```
sudo chmod 664 /usr/lib/systemd/system/process_exporter.service
```

Step 3.7: Reload systemd and start Process Exporter

Reload the `systemd` service to register the prometheus service and start the prometheus service.

```
sudo systemctl daemon-reload
sudo systemctl start process_exporter
```

Check the Process Exporter service status using the following command.

```
sudo systemctl status process_exporter
```

```
● process_exporter.service - Process Exporter for Prometheus
   Loaded: loaded (/usr/lib/systemd/system/process_exporter.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2020-04-29 14:20:25 UTC; 6s ago
     Docs: https://github.com/ncabatoff/process-exporter
 Main PID: 20813 (process-exporte)
   CGroup: /system.slice/process_exporter.service
           └─20813 /usr/bin/process-exporter --config.path /etc/process_exporter/process-exporter.yaml --web.listen-address=:9256
```

Configure process_exporter to start at boot

```
sudo systemctl enable process_exporter.service
```

If `firewalld` is enabled and running, add a rule for port `9256`

```
sudo firewall-cmd --permanent --zone=public --add-port=9256/tcp
sudo firewall-cmd --reload
```

Step 3.8: Verify the Exporter is Running

Verify the exporter is running by visiting the `/metrics` endpoint on the node on port `9256`

```
http://<process_exporter-ip>:9256/metrics
```

You should be able to see something similar to the following:

```
...
# HELP namedprocess_namegroup_context_switches_total Context switches
# TYPE namedprocess_namegroup_context_switches_total counter
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="beam.smp"} 6657
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="cbft"} 441
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="cbq-engine"} 3
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="eventing-consumer"} 0
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="eventing-producer"} 1225
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="godu"} 0
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="goport"} 3
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="goxdcr"} 52
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="indexer"} 1932
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="memcached"} 3105
namedprocess_namegroup_context_switches_total{ctxswitchtype="nonvoluntary",g
roupname="projector"} 11
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="beam.smp"} 35152
```

```
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="cbft"} 3229
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="cbq-engine"} 9164
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="eventing-consumer"} 680
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="eventing-producer"} 4690
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="godu"} 1723
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="goport"} 3865
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="goxdcr"} 3315
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="indexer"} 121989
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="memcached"} 7141
namedprocess_namegroup_context_switches_total{ctxswitchtype="voluntary",grou
pname="projector"} 9823
...
```

Step 3.9: Clean Up

Remove the download and temporary files

```
rm -rf process-exporter-0.6.0.linux-amd64.tar.gz process_exporter-files
```

---

4. Couchbase Exporter Setup

The following will walk you through how to install and configure the Couchbase Exporter and its dependencies. Depending on which mode you want to run the Couchbase Exporter in, there will either be a single instance for each cluster, or an instance for each node in the cluster.

Step 4.1: Download

Download the Couchbase Exporter python code.

```
curl -L \
  https://github.com/couchbaselabs/cbprometheus_python/tarball/master > \
  couchbase_exporter.tar.gz
```

Step 4.2: Create User

Create a Couchbase Exporter user, required directories, and make prometheus user as the owner of those directories.

```
sudo groupadd -f couchbase_exporter
sudo useradd -g couchbase_exporter --no-create-home --shell /bin/false couch
base_exporter
sudo mkdir /etc/couchbase_exporter
sudo chown couchbase_exporter:couchbase_exporter /etc/couchbase_exporter
```

Step 4.3: Unpack Couchbase Exporter

Untar and move the downloaded Couchbase Exporter code

```
mkdir -p couchbase_exporter
tar -xzf couchbase_exporter.tar.gz \
  -C couchbase_exporter --strip-components=1
```

Step 4.4: Install Python Dependencies

Python is required for the exporter to run, along with the `uwsgi` package.

`pip` is not packaged in official software repositories of CentOS/RHEL. The EPEL repository needs to be enabled.

CentOS

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

If you're running in AWS you'll need to run:

```
sudo amazon-linux-extras install epel
```

Install `pip`

CentOS

```
sudo yum install python-pip python-devel gcc -y
```

Install `uwsgi` and `flask` using `pip`

```
sudo pip install -r ./couchbase_exporter/requirements
```

Step 4.5: Install Couchbase Exporter

Copy `couchbase_exporter` directory from `couchbase_exporter` folder to `/opt/couchbase_exporter` and change the ownership to the couchbase_exporter user.

```
sudo mv couchbase_exporter /opt
sudo chown -R couchbase_exporter:couchbase_exporter /opt/couchbase_exporter
```

Step 4.6: Setup ssh keys for cbstats

This step only needs to be performed if you are running the exporter in a cluster/standalone mode and wish to retrieve `cbstats` metrics. If you are running the exporter in local mode, this step is not required as the local version of `cbstats` is used.

This can be done a few ways. This example we will be creating a user for the exporter to use on the Couchbase nodes. You will need to have ssh sudo access to complete this step.

From the exporter:

```
ssh-keygen -t rsa -b 4096 -C "enter.user@domain.com"
```

Enter file in which to save the key (/home/vagrant/.ssh/id_rsa): `exporter` \ Enter passphrase (empty for no passphrase):\ Enter same passphrase again:\ Your identification has been saved in exporter.\ Your public key has been saved in exporter.pub.

```
mv exporter* ~/.ssh
cat ~/.ssh/exporter.pub
```

Copy the key to your clipboard

You can setup keys on each of the individual Couchbase nodes and the exporter will connect to each node and run cbstats against that node. Or you can setup the key on a single host in the cluster and use that node to access the other nodes in the cluster. If you do the latter you have to set the `CB_SSH_HOST` environment variable.

On each host the exporter will need to connect to:

```
sudo useradd -m -d /home/exporter -s /bin/bash -G couchbase exporter
sudo su
mkdir /home/exporter/.ssh
```

```
chown exporter:exporter /home/exporter/.ssh/
vi /home/exporter/.ssh/authorized_keys
```

Paste the copied key into the authorized keys file

```
chmod 600 /home/exporter/.ssh/authorized_keys
chown exporter:exporter /home/exporter/.ssh/authorized_keys
exit
```

Step 4.7 Configure uwsgi Emperor

Emperor will maintain and execute multiple instances of `uwsgi`.

Create a directory for the `uwsgi` configuration files.

```
sudo mkdir -p /etc/uwsgi/vassals
```

Create a new file for the `emperor.ini`

```
sudo vi /etc/uwsgi/emperor.ini
```

Add the following contents to the `emperor.ini` file

```
[uwsgi]
emperor = /etc/uwsgi/vassals
```

Set the appropriate permissions

```
sudo chown -R couchbase_exporter:couchbase_exporter /etc/uwsgi
```

Step 4.8 Configure Vassals (Cluster Monitoring Instances)

Create an `ini` file for each cluster that you wish to monitor.

```
sudo vi /etc/uwsgi/vassals/{{CLUSTER}}.ini
```

Replace `{{CLUSTER}}` with a friendly name that contains no spaces i.e. (`cluster1.ini`).

Add the following contents to the file. Replace `{{CLUSTER_HOSTNAME}}` with the hostname of one of the Couchbase nodes in the cluster that you wish to monitor. Each exporter will need to run on a different port, it is recommended that you start with `5000` for `{{PORT}}` and increment by 1 (i.e. 5000, 5001, 5002, etc.)

Variables

- `CB_EXPORTER_MODE` This can be `"standalone"` or `"local"` .
- `CLUSTER` - Friendly cluster name (no spaces). If `CB_EXPORTER_MODE` is set to `local` this value is changed to `"localhost"`
- `CLUSTER_HOSTNAME` - A comma-delimited list of one or more nodes (from the same cluster).
- `CLUSTER_USERNAME` - An RBAC user with Read-Only Admin as well as System Catalog Query Permissions
- `CLUSTER_PASSWORD` - The Password for the RBAC user
- `PORT` - The port for the exporter to listen on
- `CB_RESULTSET` - Optional, used to limit the result size. Default is 60. For larger clusters or clusters with a high number of buckets/indexes, consider lowering this value.
- `CB_CBSTAT_PATH` - Optional, Used to state path to cbstats for non-default installations of Couchbase
- `CB_KEY` - Required if intending to use cbstats in standalone mode from the exporter, path to private key
- `CB_SSH_USER` - Required if intending to use cbstats from the exporter in standalone mode, username for private key
- `CB_SSH_HOST` - Required if using cbstats in standalone mode and only connecting to a single host, ip address of that host.
- `CB_NODE_EXPORTER_PORT` - Optional, The port that node exporter is running on. The Couchbase Exporter can act as a proxy to Node Exporter, retrieving Node Exporter and adding labels with Couchbase Server information to the Node Exporter Metrics. Defaults to 9200.
- `CB_PROCESS_EXPORTER_PORT` - Optional, The port that process exporter is running on. The Couchbase Exporter can act as a proxy to Process Exporter, retrieving Process Exporter and adding labels with Couchbase Server information to the Node Exporter Metrics. Defaults to 9256.

```
[uwsgi]
http = :{{PORT}}
pidfile = /tmp/{{CLUSTER}}.pid
env = CB_DATABASE={{CLUSTER_HOSTNAME}}
env = CB_USERNAME={{CLUSTER_USERNAME}}
env = CB_PASSWORD={{CLUSTER_PASSWORD}}
env = CB_RESULTSET={{CB_RESULTSET}}
```

```
env = CB_CBSTAT_PATH={{CB_CBSTAT_PATH}}
env = CB_KEY={{CB_KEY}}
env = CB_SSH_USER={{CB_SSH_USER}}
env = CB_SSH_HOST={{CB_SSH_HOST}}
processes = 1
master =
chdir = /opt/couchbase_exporter/src
wsgi-file = /opt/couchbase_exporter/src/wsgi.py
enable-threads =
```

Set the appropriate permissions on the file

```
sudo chown couchbase_exporter:couchbase_exporter /etc/uwsgi/vassals/{{CLUSTE
R}}.ini
```

Step 4.9: Setup Service

Configure emperor to run as a service by creating the following file:

```
sudo vi /usr/lib/systemd/system/emperor.uwsgi.service
```

Add the following configuration

```
[Unit]
Description=uWSGI Emperor
After=syslog.target

[Service]
User=couchbase_exporter
Group=couchbase_exporter
ExecStart=/usr/bin/uwsgi --ini /etc/uwsgi/emperor.ini
RuntimeDirectory=/opt/couchbase_exporter
Restart=always
KillSignal=SIGQUIT
Type=notify
StandardError=syslog
NotifyAccess=all

[Install]
WantedBy=multi-user.target
```

Set the appropriate permissions

```
sudo chmod 664 /usr/lib/systemd/system/emperor.uwsgi.service
```

Step 4.10: Reload systemd and start Emperor

Reload the `systemd` service to register the prometheus service and start the prometheus service.

```
sudo systemctl daemon-reload
sudo systemctl start emperor.uwsgi.service
```

Check the Emperor service status using the following command.

```
sudo systemctl status emperor.uwsgi.service
```

```
[ec2-user@ip-172-31-1-114 ~]$ sudo systemctl status emperor.uwsgi.service
● emperor.uwsgi.service - uWSGI Emperor
   Loaded: loaded (/usr/lib/systemd/system/emperor.uwsgi.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2020-02-13 19:00:05 UTC; 10s ago
 Main PID: 32668 (uwsgi)
   Status: "The Emperor is governing 2 vassals"
   CGroup: /system.slice/emperor.uwsgi.service
           ├─32668 /usr/bin/uwsgi --ini /etc/uwsgi/emperor.ini
           ├─32669 /usr/bin/uwsgi --ini cluster2.ini
           ├─32670 /usr/bin/uwsgi --ini cluster1.ini
           ├─32673 /usr/bin/uwsgi --ini cluster2.ini
           ├─32674 /usr/bin/uwsgi --ini cluster2.ini
           ├─32675 /usr/bin/uwsgi --ini cluster1.ini
           └─32676 /usr/bin/uwsgi --ini cluster1.ini
```

Configure Emperor to start at boot

```
sudo systemctl enable emperor.uwsgi.service
```

If `firewalld` is enabled and running, add a rule for port each exporter configured i.e `5000`, `5001`, etc.

```
sudo firewall-cmd --permanent --zone=public --add-port=5000/tcp
sudo firewall-cmd --reload
```

Step 4.11: Verify the Exporter is Running

Verify the exporter is running by visiting the `/metrics` endpoint on the node on port `5000`

```
http://<couchbase_exporter/emperor-ip>:5000/metrics/buckets
```

You should be able to see something similar to the following:

```
...
ep_dcp_other_producer_count {cluster="Demo-6.0.3", bucket="demo", node="10.1
.2.100", type="bucket"} 0 1581621651527
ep_dcp_other_producer_count {cluster="Demo-6.0.3", bucket="demo", node="10.1
.2.100", type="bucket"} 0 1581621652528
ep_dcp_other_producer_count {cluster="Demo-6.0.3", bucket="demo", node="10.1
.2.100", type="bucket"} 0 1581621653528
ep_dcp_other_producer_count {cluster="Demo-6.0.3", bucket="demo", node="10.1
.2.100", type="bucket"} 0 1581621654527
ep_dcp_other_producer_count {cluster="Demo-6.0.3", bucket="demo", node="10.1
.2.100", type="bucket"} 0 1581621655527
ep_dcp_other_producer_count {cluster="Demo-6.0.3", bucket="demo", node="10.1
.2.100", type="bucket"} 0 1581621656528
ep_dcp_other_producer_count {cluster="Demo-6.0.3", bucket="demo", node="10.1
.2.100", type="bucket"} 0 1581621657528
...
```

Step 4.12: Clean Up

Remove the download and temporary files

```
rm -rf couchbase_exporter*
```

If you wish to add another cluster in the future, repeat Step 4.7 and restart the emperor service.

---

5. Configure Prometheus

Prometheus is configured through a single YAML file called `prometheus.yml`. When we configured Prometheus to run as a service, we specified the path of `/etc/prometheus/prometheus.yml`.

After changing the file, the prometheus service will need to be restarted to pickup the changes.

```
sudo systemctl restart prometheus
```

When we installed prometheus we configured an additional tool called `promtool`. It can be used to interact with prometheus, execute queries as well as validate configuration files.

```
promtool check config /etc/prometheus/prometheus.yml
```

Prometheus ingests stats via scrape jobs, we will configure 10 different scrape jobs for the following:

1. Prometheus
2. Node Exporter
3. Process Exporter
4. Buckets
5. Indexes
6. Query
7. XDCR
8. System
9. Eventing
10. Analytics
11. FTS

You do not have to restart Prometheus after adding each job listed below. The restarts and validation are simply added for verification purposes.

A full `prometheus.yml` file can be found at https://github.com/couchbaselabs/cbprometheus_python/blob/master/prometheus/prometheus.yml

Step 5.1 Configure Prometheus to Monitor Itself

Edit the prometheus configuration file

```
sudo vi /etc/prometheus/prometheus.yml
```

Add the following contents

```
global:
  scrape_interval: 60s # How frequently to scrape targets by default.
  scrape_timeout: 10s # How long until a scrape request times out.
  evaluation_interval: 60s # How frequently to evaluate rules.

# A scrape configuration
scrape_configs:
  - job_name: prometheus
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics
```

```
    static_configs:
    - targets: ['localhost:9090']
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Validate the target has been added and is being monitoring

Open the Prometheus UI

```
http://<prometheus-ip>:9090/targets
```

The new job `prometheus` should be listed with a status of "Up". If the status shows as "Unknown" give it a few seconds and refresh.

5.2 Setup File Service Discovery

When configuring Prometheus to monitor itself, a static config was used. For the remaining jobs, a file_sd_config will be used.

Create the following directory and file with the appropriate permissions.

```
sudo mkdir /etc/prometheus/file_sd
sudo touch /etc/prometheus/file_sd/couchbase.yml
sudo chown prometheus:prometheus /etc/prometheus/file_sd
sudo chown prometheus:prometheus /etc/prometheus/file_sd/couchbase.yml
```

Edit the `/etc/prometheus/file_sd/couchbase.yml` with each of the Couchbase Exporter instances that have been configured.

```
sudo vi /etc/prometheus/file_sd/couchbase.yml
```

```
- targets:
  - node1.cluster1.example.org
  - node2.cluster1.example.org
  - node3.cluster1.example.org
  - node1.cluster2.example.org
  - node2.cluster2.example.org
  - node3.cluster2.example.org
```

Once Prometheus has been fully configured, anytime new nodes are added edit the `/etc/prometheus/file_sd/couchbase.yml` file with the new nodes. The nodes will automatically be picked up by Prometheus and monitored without a restart of Prometheus being required.

5.3 Configure Node Exporter Job

If you have not configured the Node Exporter from Step 2, skip this step and proceed to the next step. Here we will configure the Node Exporter job to gather system metrics from each of our nodes in each of the clusters.

The Couchbase Exporter will be used as a proxy to Node Exporter, the Couchbase Exporter will add the labels of `cluster="...", node="..."` to the metrics returned by Node Exporter, this way the Node Exporter metrics can be associated with Couchbase metrics.

Edit the `prometheus.yml` file and add the following job.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: node_exporter
    honor_labels: true
    scheme: http
    scrape_interval: 10s
    scrape_timeout: 9s
    metrics_path: /metrics/node_exporter
    file_sd_configs:
      - files:
        - /etc/prometheus/file_sd/couchbase.yml
```

Spacing is very important in YAML and we want to validate our changes before they take effect. You can validate your Prometheus config by issuing the following command:

```
promtool check config /etc/prometheus/prometheus.yml
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.4 Configure Process Exporter Job

If you have not configured the Process Exporter from Step 2, skip this step and proceed to the next step. Here we will configure the Process Exporter to gather system metrics from each of our nodes in each of the clusters.

The Couchbase Exporter will be used as a proxy to Process Exporter, the Couchbase Exporter will add the labels of `cluster="...", node="..."` to the metrics returned by Process Exporter, this way the Process Exporter metrics can be associated with Couchbase metrics.

Edit the `prometheus.yml` file and add the following job.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: process_exporter
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 10s
    scrape_timeout: 9s
    metrics_path: /metrics/process_exporter
    file_sd_configs:
      - files:
        - /etc/prometheus/file_sd/couchbase.yml
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.5 Configure Couchbase Buckets Job

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
- job_name: couchbase-buckets
  honor_labels: true
  honor_timestamps: true
  scheme: http
  scrape_interval: 60s
  scrape_timeout: 55s
  metrics_path: /metrics/buckets
  file_sd_configs:
    - files:
      - /etc/prometheus/file_sd/couchbase.yml
  metric_relabel_configs:
    # if the stat name starts with data_* strip off index_
    - source_labels: [__name__]
      regex: 'data_(.*)'
      replacement: '$1'
      target_label: __name__
    # add data_ to the start of every stat
    - source_labels: [__name__]
      regex: '(.*)'
      replacement: 'data_$1'
      target_label: __name__
```

Notice in this instance there are 2 targets configured, one on port `5000` and one on `5001` this is to illustrate monitoring multiple clusters. In Step 3 we demonstrated how to configure multiple clusters and configure the appropriate ini files.

Additionally, for this job we are leveraging a metric*renamer. This takes an existing Couchbase stat as exposed by the exporter and renames it. This ensures for this job, every stat that is ingested is prefixed with `data*`. This is useful when using issuing PromQL statements or querying in Grafana which we will cover later.

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.6 Configure Couchbase Indexes Job

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: couchbase-indexes
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics/indexes
    file_sd_configs:
      - files:
        - /etc/prometheus/file_sd/couchbase.yml
    metric_relabel_configs:
      # if the stat name starts with index_* strip off index_
      - source_labels: [__name__]
        regex: 'index_(.*)'
        replacement: '$1'
        target_label: __name__
      # add index_ to the start of every stat
      - source_labels: [__name__]
        regex: '(.*)'
        replacement: 'index_$1'
        target_label: __name__
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.7 Configure Couchbase Queries Job

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: couchbase-queries
    honor_labels: true
```

```
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics/query
    file_sd_configs:
      - files:
        - /etc/prometheus/file_sd/couchbase.yml
    metric_relabel_configs:
      # if the stat name starts with query_* strip off index_
      - source_labels: [__name__]
        regex: 'query_(.*)'
        replacement: '$1'
        target_label: __name__
      # add query_ to the start of every stat
      - source_labels: [__name__]
        regex: '(.*)'
        replacement: 'query_$1'
        target_label: __name__
```

If you do not want slow queries to be returned from the `/metrics/queries` endpoint, you can add the following block to the job:

```
    params:
      slow_queries: false
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.8 Configure Couchbase XDCR Job

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: couchbase-xdcr
```

```
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics/xdcr
    file_sd_configs:
      - files:
        - /etc/prometheus/file_sd/couchbase.yml
    metric_relabel_configs:
      # if the stat name starts with xdcr_* strip off index_
      - source_labels: [__name__]
        regex: 'xdcr_(.*)'
        replacement: '$1'
        target_label: __name__
      # add xdcr_ to the start of every stat
      - source_labels: [__name__]
        regex: '(.*)'
        replacement: 'xdcr_$1'
        target_label: __name__
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.9 Configure Couchbase System Job

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: couchbase-system
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics/system
```

```
    file_sd_configs:
      - files:
        - /etc/prometheus/file_sd/couchbase.yml
    metric_relabel_configs:
      # if the stat name starts with system_* strip off index_
      - source_labels: [__name__]
        regex: 'system_(.*)'
        replacement: '$1'
        target_label: __name__
      # add system_ to the start of every stat
      - source_labels: [__name__]
        regex: '(.*)'
        replacement: 'system_$1'
        target_label: __name__
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.10 Configure Couchbase Eventing Job

Even if you are not currently using Eventing in your deployment, it is suggested to monitor it via Prometheus and the Couchbase Exporter. This way if it is ever enabled, you will begin to immediately start to monitor it.

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: couchbase-eventing
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics/eventing
    file_sd_configs:
      - files:
```

```
        - /etc/prometheus/file_sd/couchbase.yml
    metric_relabel_configs:
      # if the stat name starts with eventing_* strip off index_
      - source_labels: [__name__]
        regex: 'eventing_(.*)'
        replacement: '$1'
        target_label: __name__
      # add eventing_ to the start of every stat
      - source_labels: [__name__]
        regex: '(.*)'
        replacement: 'eventing_$1'
        target_label: __name__
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.11 Configure Couchbase Analytics Job

Even if you are not currently using Analytics in your deployment, it is suggested to monitor it via Prometheus and the Couchbase Exporter. This way if it is ever enabled, you will begin to immediately start to monitor it.

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: couchbase-analytics
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics/analytics
    file_sd_configs:
      - files:
        - /etc/prometheus/file_sd/couchbase.yml
    metric_relabel_configs:
```

```
      # if the stat name starts with analytics_* strip off index_
      - source_labels: [__name__]
        regex: 'analytics_(.*)'
        replacement: '$1'
        target_label: __name__
      # add analytics_ to the start of every stat
      - source_labels: [__name__]
        regex: '(.*)'
        replacement: 'analytics_$1'
        target_label: __name__
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.12 Configure Couchbase FTS Job

Even if you are not currently using FTS in your deployment, it is suggested to monitor it via Prometheus and the Couchbase Exporter. This way if it is ever enabled, you will begin to immediately start to monitor it.

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: couchbase-fts
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics/fts
    file_sd_configs:
      - files:
        - /etc/prometheus/file_sd/couchbase.yml
    metric_relabel_configs:
      # if the stat name starts with fts_* strip off index_
      - source_labels: [__name__]
```

```
        regex: 'fts_(.*)'
        replacement: '$1'
        target_label: __name__
      # add fts_ to the start of every stat
      - source_labels: [__name__]
        regex: '(.*)'
        replacement: 'fts_$1'
        target_label: __name__
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

5.13 Configure Couchbase cbstats Job

This job will execute cbstats against each node in the cluster. These metrics are different than the ones that have been previously gathered and should be queried at an independent interval as the values returned are point in time without a history. This requires that every node in the cluster is configured with a public/private key pair and that the system variables are set in the exporter configuration.

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: couchbase-cbstats
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics/cbstats
    file_sd_configs:
      - files:
        - /etc/prometheus/file_sd/couchbase.yml
    metric_relabel_configs:
      # if the stat name starts with fts_* strip off index_
```

```
      - source_labels: [__name__]
        regex: 'cbstats_(.*)'
        replacement: '$1'
        target_label: __name__
      # add fts_ to the start of every stat
      - source_labels: [__name__]
        regex: '(.*)'
        replacement: 'cbstats_$1'
        target_label: __name__
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

Open the Prometheus UI and validate the new job is listed and is "Up"

```
http://<prometheus-ip>:9090/targets
```

---

6. Configure AlertManager

It is important to know that Prometheus can support multiple AlertManagers for High-Availability by specifying more than 1 `alertmanager` target. For the purpose of this example, we will install a single AlertManager on the same server as Prometheus and Grafana.

Step 6.1: Download

Download the AlertManager binary to the server that you will use for alerting.

```
wget \
  https://github.com/prometheus/alertmanager/releases/download/v0.21.0/alert
manager-0.21.0.linux-amd64.tar.gz
```

Visit the Prometheus downloads page for the latest version.

Step 6.2: Create User

Create a Prometheus user, required directories, and make prometheus user as the owner of those directories.

```
sudo groupadd -f alertmanager
sudo useradd -g alertmanager --no-create-home --shell /bin/false alertmanage
r
sudo mkdir -p /etc/alertmanager/templates
sudo mkdir /var/lib/alertmanager
sudo chown alertmanager:alertmanager /etc/alertmanager
sudo chown alertmanager:alertmanager /var/lib/alertmanager
```

Step 6.3: Unpack

Untar and move the downloaded Prometheus binary

```
tar -xvf alertmanager-0.21.0.linux-amd64.tar.gz
mv alertmanager-0.21.0.linux-amd64 alertmanager-files
```

Step 6.4: Install AlertManager

Copy `alertmanager` and `amtool` binary from prometheus-files folder to `/usr/bin` and change the ownership to prometheus user.

```
sudo cp alertmanager-files/alertmanager /usr/bin/
sudo cp alertmanager-files/amtool /usr/bin/
sudo chown alertmanager:alertmanager /usr/bin/alertmanager
sudo chown alertmanager:alertmanager /usr/bin/amtool
```

Step 6.5: Install AlertManager Configuration File

Move the `alertmanager.yml` file from `alertmanager-files` to the `/etc/alertmanager` folder and change the ownership to alertmanager user.

```
sudo cp alertmanager-files/alertmanager.yml /etc/alertmanager/alertmanager.y
ml
sudo chown alertmanager:alertmanager /etc/alertmanager/alertmanager.yml
```

Step 6.6: Setup Service

Create a alertmanager service file.

```
sudo vi /usr/lib/systemd/system/alertmanager.service
```

Add the following configuration and save the file

```
[Unit]
Description=AlertManager
Wants=network-online.target
After=network-online.target

[Service]
User=alertmanager
Group=alertmanager
Type=simple
ExecStart=/usr/bin/alertmanager \
    --config.file /etc/alertmanager/alertmanager.yml \
    --storage.path /var/lib/alertmanager/

[Install]
WantedBy=multi-user.target
```

```
sudo chmod 664 /usr/lib/systemd/system/alertmanager.service
```

Step 6.7: Reload systemd

Reload the `systemd` service to register the prometheus service and start the prometheus service.

```
sudo systemctl daemon-reload
sudo systemctl start alertmanager
```

Check the alertmanager service status using the following command.

```
sudo systemctl status alertmanager
```

```
[ec2-user@ip-172-31-2-75 ~]$ sudo systemctl status alertmanager
● alertmanager.service - AlertManager
   Loaded: loaded (/usr/lib/systemd/system/alertmanager.service; disabled; vendor preset: disabled)
   Active: active (running) since Fri 2020-02-14 14:50:06 UTC; 6s ago
 Main PID: 11582 (alertmanager)
   CGroup: /system.slice/alertmanager.service
           └─11582 /usr/local/bin/alertmanager --config.file /etc/alertmanager/alertmanager.yml --storage.path /var/lib/alertmanager/
```

Configure AlertManager to start at boot

```
sudo systemctl enable alertmanager.service
```

If `firewalld` is enabled and running, add a rule for port `9093`

```
sudo firewall-cmd --permanent --zone=public --add-port=9093/tcp
```

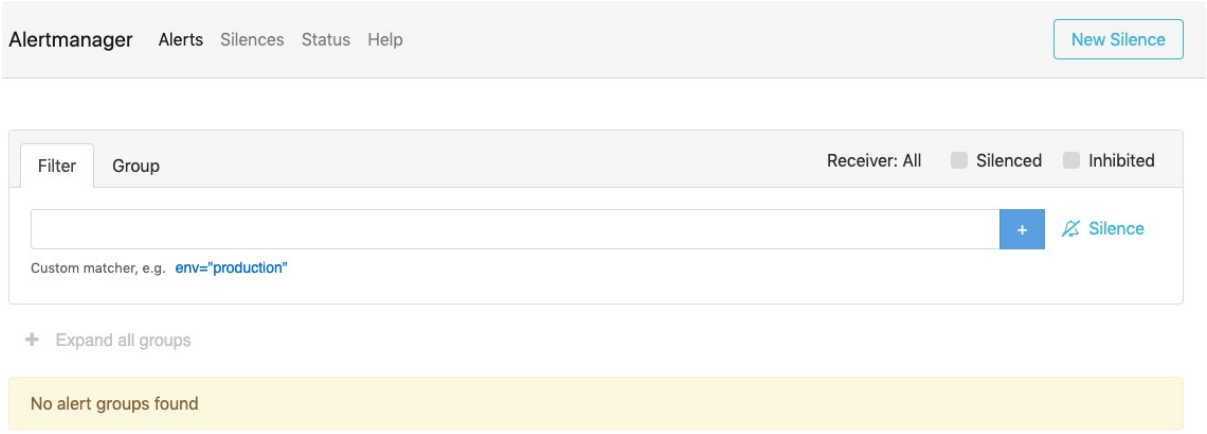
```
sudo firewall-cmd --reload
```

Step 6.8: Access UI

Now you will be able to access the AlertManager UI on `9093` port of the alertmanager server.

```
http://<alertmanager-ip>:9093
```

You should be able to see the following UI as shown below.



Step 6.9 Create a Test Alert

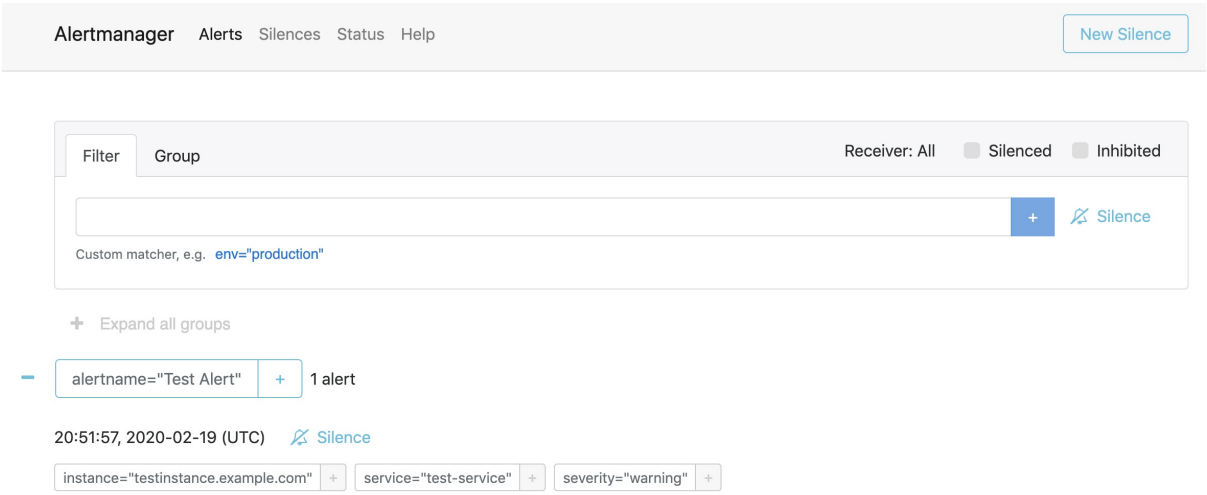
Execute the following statement, be sure to replace `<alertmanager-ip>` with the IP address / hostname of your Alertmanager instance. If you are ssh'd into the Alertmanager server already you can use `localhost` .

```
curl -XPOST "http://<alertmanager-ip>:9093/api/v1/alerts" \
  -d \
  "[{
    \"status\": \"firing\",
    \"labels\": {
      \"alertname\": \"Test Alert\",
      \"service\": \"test-service\",
      \"severity\":\"warning\",
      \"instance\": \"testinstance.example.com\"
    },
    \"annotations\": {
      \"summary\": \"High latency is high!\"
    }
  }]"
```

Open the Alertmanager UI in a web browser

```
http://<alertmanager-ip>:9093
```

You should be able to see your test alert in the UI as shown below.



Prometheus automatically takes care of sending alerts generated by its configured alerting rules, it is not recommended to generate alerts by calling the AlertManager APIs directly.

Step 6.10 Configure AlertManager

Edit the `alertmanager.yml` file and view the current configuration.

```
sudo vi /etc/alertmanager/alertmanager.yml
```

```
global:
  resolve_timeout: 5m

route:
  group_by: ['alertname']
  group_wait: 10s
  group_interval: 10s
  repeat_interval: 1h
  receiver: 'web.hook'
receivers:
- name: 'web.hook'
  webhook_configs:
  - url: 'http://127.0.0.1:5001/'
inhibit_rules:
  - source_match:
```

```
      severity: 'critical'
    target_match:
      severity: 'warning'
    equal: ['alertname', 'dev', 'instance']
```

Restart AlertManager

```
sudo systemctl restart alertmanager
```

Currently the only receiver that is configured is a webhook on the local machine. There are an endless number of possibilities for configuring AlertManager including sending emails, webhooks, as well as 3rd party integrations such as Slack or PagerDuty. This guide does not cover these integrations, these can be implemented separately by referencing the documentation at https://prometheus.io/docs/alerting/configuration/.

Step 6.11: Clean Up

Remove the download and temporary files

```
rm -rf alertmanager-0.21.0.linux-amd64.tar.gz alertmanager-files
```

---

7. Configure Prometheus Alerts

Step 7.1 Configure Prometheus to use AlertManager

Edit the `prometheus.yml` file from the server that Prometheus is installed on and add the following YAML below the `global:` block and before the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - <alertmanager-ip>:9093
    scheme: http
    timeout: 10s
```

Step 7.2 Configure Prometheus to Monitor AlertManager

Edit the `prometheus.yml` file and add the following under the `scrape_configs:` block.

```
sudo vi /etc/prometheus/prometheus.yml
```

```
  - job_name: alertmanager
    honor_labels: true
    honor_timestamps: true
    scheme: http
    scrape_interval: 60s
    scrape_timeout: 55s
    metrics_path: /metrics
    static_configs:
    - targets: ['localhost:9093']
```

Restart Prometheus

```
sudo systemctl restart prometheus
```

6.3 Create Rules

Create a rules directory for prometheus to reference.

```
sudo mkdir -p /etc/prometheus/rules
```

Copy all of the example rules into the directory:

```
sudo cp /opt/couchbase_exporter/prometheus/rules/*.yml /etc/prometheus/rules
```

Set the permissions so that the prometheus user is the owner.

```
sudo chown -R prometheus:prometheus /etc/prometheus/rules
```

Verify that all of the rules are valid by using `promtool`

```
promtool check rules /etc/prometheus/rules/*.yml
```

The output should show SUCCESS for all rules files, similar to the following:

```
Checking /etc/prometheus/rules/couchbase.analytics.rules.yml
  SUCCESS: 2 rules found

Checking /etc/prometheus/rules/couchbase.bucket.rules.yml
  SUCCESS: 10 rules found

Checking /etc/prometheus/rules/couchbase.eventing.rules.yml
  SUCCESS: 2 rules found

Checking /etc/prometheus/rules/couchbase.fts.rules.yml
  SUCCESS: 2 rules found

Checking /etc/prometheus/rules/couchbase.index.rules.yml
  SUCCESS: 2 rules found

Checking /etc/prometheus/rules/couchbase.query.rules.yml
  SUCCESS: 4 rules found

Checking /etc/prometheus/rules/couchbase.system.rules.yml
  SUCCESS: 4 rules found

Checking /etc/prometheus/rules/couchbase.xdcr.rules.yml
  SUCCESS: 4 rules found
```

6.4 Configure Prometheus Rules

The rules files exist, now prometheus needs to be configured to use them. Add the following YAML after the `alerting:` block and before the `scrape_configs:` block.

```
# Load rules once and periodically evaluate them according
# to the global evaluation_interval.
rule_files:
  - "rules/couchbase.*.rules.yml"
```

```
sudo vi /etc/prometheus/prometheus.yml
```

Validate the configuration changes using `promtool`

```
promtool check config /etc/prometheus/prometheus.yml
```

Restart Prometheus so the configuration change is picked up.

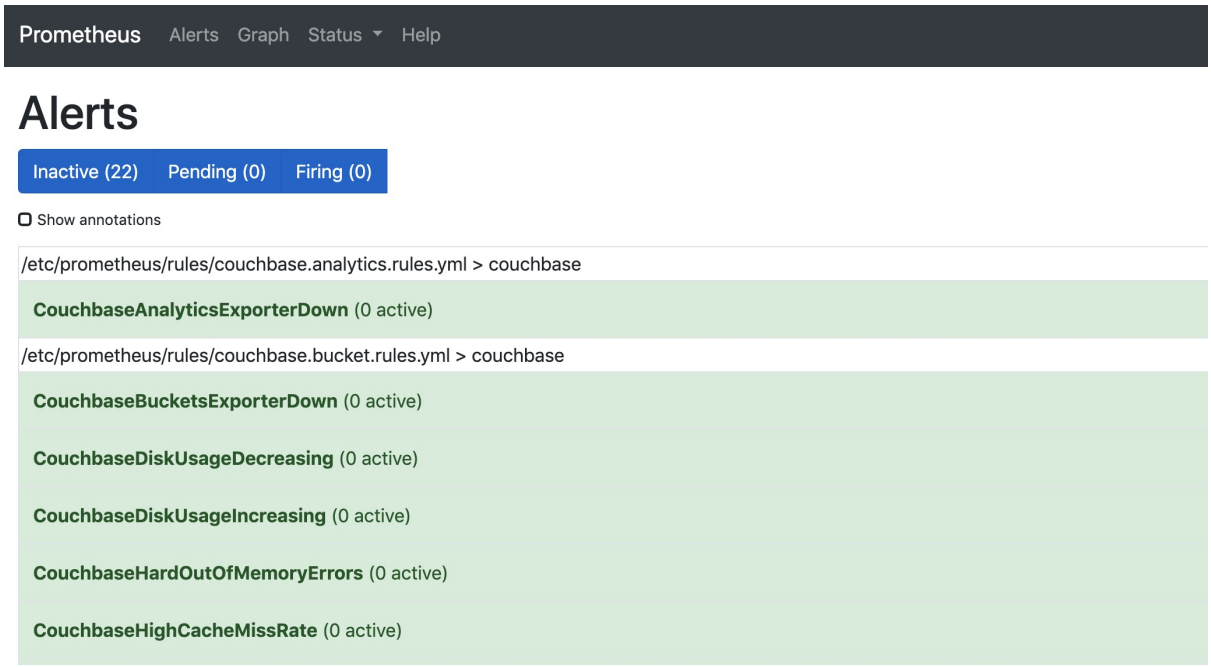
```
sudo systemctl restart prometheus
```

Step 7.5: Access UI

Open the Prometheus UI and go to the "Alerts" tab.

```
http://<prometheus-ip>:9090/alerts
```

You should be able to see all of the configured alerts in the UI.



Disclaimer: The rules that have been provided are for example purposes only. Alerts should be configured and tailored specific to your use-case and environments. Please review the documentation for adding your own custom alerts https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/

---

8. Install Grafana

The following will walk you through how to install and configure Grafana. This should NOT be installed on each Couchbase node or any node in the cluster. This should be a stand alone instance.

Step 8.1: Configure Yum

Add a new file to your YUM repository using the method of your choice.

```
sudo vi /etc/yum.repos.d/grafana.repo
```

Add the following to the file and save it.

```
[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
```

If you would like to install using `rpm` visit
https://grafana.com/docs/grafana/latest/installation/rpm/

Step 8.2: Install Grafana

```
sudo yum install grafana -y
```

Package details

- Installs binary to `/usr/sbin/grafana-server`
- Installs default file (environment vars) to `/etc/sysconfig/grafana-server`
- Copies configuration file to `/etc/grafana/grafana.ini`
- Installs systemd service (if systemd is available) name `/usr/lib/systemd/system/grafana-server.service`
- The default configuration uses a log file at `/var/log/grafana/grafana.log`
- The default configuration specifies an sqlite3 database at `/var/lib/grafana/grafana.db`

Step 8.3: Reload systemd and start Grafana

Reload the `systemd` service to register the grafana service and start the grafana service.

```
sudo systemctl daemon-reload
sudo systemctl start grafana-server
```

Check the grafana service status using the following command.

```
sudo systemctl status grafana-server
```

Configure grafana to start at boot

```
sudo systemctl enable grafana-server.service
```

Step 8.4: Access UI

Now you will be able to access the Grafana UI on port `3000` of the server.

```
http://<grafana-ip>:3000
```

You should be able to see the following UI as shown below.

The default user and password is `admin`, you will be prompted to change this but you are not required to.

---

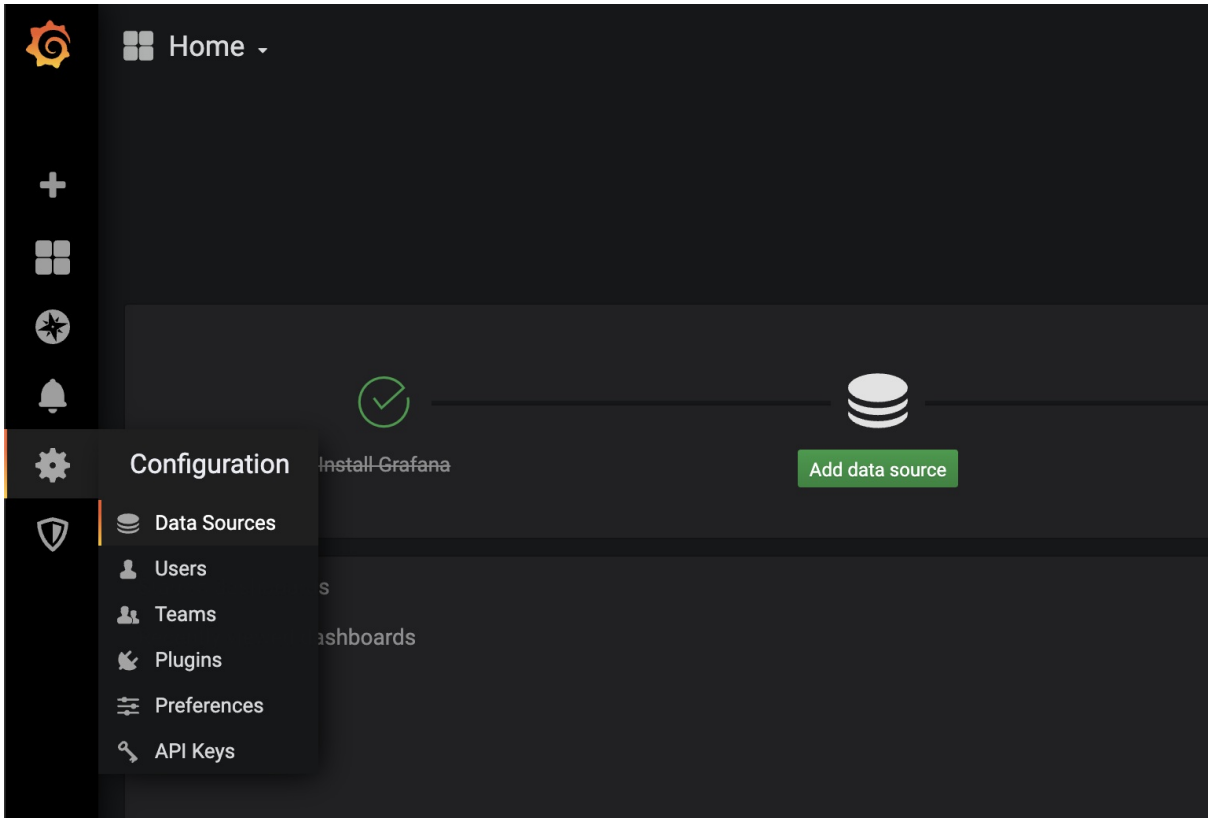
9. Configure Grafana

Step 9.1: Add Data Source

Open the Grafana UI

```
http://<grafana-ip>:3000
```

Add a datasource by going to Configuration -> Data Sources.



Click the "Add data source" button

Select "Prometheus"

Set the name to `Prometheus` (note this is case-sensitive).

In the URL, enter `http://<prometheus-ip>:9090`, leave the rest of the defaults and click "Save & Test".

Step 9.2: Import Grafana Dashboards

Dashboards provide different ways of visualizing your data. Sample dashboards can be found at https://github.com/couchbaselabs/cbprometheus_python/tree/master/grafana. For more information on creating your own dashboards and panels review the documentation provided at https://grafana.com/docs/grafana/latest/features/panels/panels/.

Click the + sign on the left-hand side of the UI, and choose "Import".

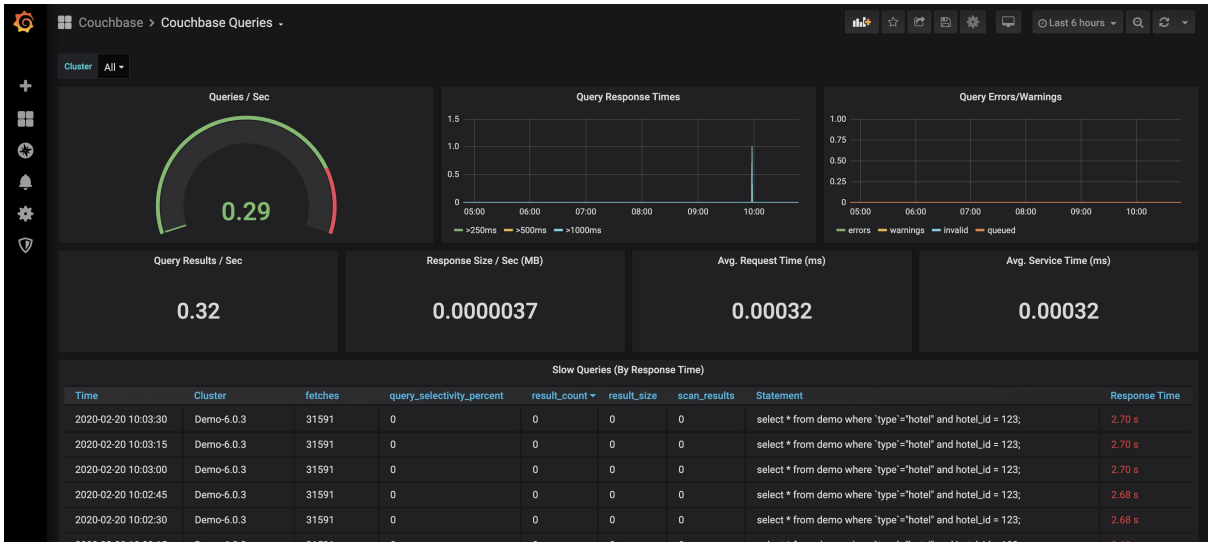
Copy the contents from Couchbase Queries-1581449916780.json and paste them into the textbox.

Click the "Load" button

This will populate the name of the Dashboard automatically. Since this is the first Dashboard being created, click on the Folder dropdown, and choose "--New Folder--". Type "Couchbase" for the folder name and click "Create".

Click the "Import" button

You should see a dashboard similar to the following:



Repeat this process for any other existing dashboards that you want to visualize and start creating your own custom dashboards!

10. Upgrading Services

1. Upgrade Prometheus
2. Upgrade Node Exporter
3. Upgrade Process Exporter
4. Upgrade Couchbase Exporter
5. Upgrade AlertManager
6. Upgrade Grafana

Upgrade Prometheus

Step 1: Download Latest Binary

Visit the Prometheus downloads page for the latest version. Copy the correct link and download the Prometheus binary to the server that you will upgrade Prometheus on.

```
wget \
  https://github.com/prometheus/prometheus/releases/download/v2.18.0/prometh
```

```
eus-2.18.0.linux-amd64.tar.gz
```

Step 2: Unpack

Untar and move the downloaded Prometheus binary

```
tar -xvf prometheus-2.18.0.linux-amd64.tar.gz
mv prometheus-2.18.0.linux-amd64 prometheus-files
```

Step 3: Stop Prometheus Service

The executable that we're replacing is in use, so we need to stop the service so it can be replaced.

```
sudo systemctl stop prometheus.service
```

Step 4: Install Prometheus

Copy `prometheus` and `promtool` binary from prometheus-files folder to `/usr/bin` and change the ownership to prometheus user.

```
sudo cp prometheus-files/prometheus /usr/bin/
sudo cp prometheus-files/promtool /usr/bin/
sudo chown prometheus:prometheus /usr/bin/prometheus
sudo chown prometheus:prometheus /usr/bin/promtool
```

Step 5: Install Prometheus Libraries

Move the `consoles` and `console_libraries` directories from `prometheus-files` to `/etc/prometheus` folder and change the ownership to prometheus user.

```
sudo cp -r prometheus-files/consoles /etc/prometheus
sudo cp -r prometheus-files/console_libraries /etc/prometheus
sudo chown -R prometheus:prometheus /etc/prometheus/consoles
sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
```

Step 6: Start Prometheus

Restart the Prometheus service

```
sudo systemctl start prometheus.service
```

Step 7: Clean Up

Remove the download and temporary files

```
rm -rf prometheus-2.18.0.linux-amd64.tar.gz prometheus-files
```

Upgrade Node Exporter

Step 1: Download Latest Binary

Visit the Prometheus downloads page for the latest version. Copy the correct link and download the Prometheus binary to the server that you will upgrade Prometheus on.

```
wget \
  https://github.com/prometheus/node_exporter/releases/download/v0.18.1/node
_exporter-0.18.1.darwin-amd64.tar.gz
```

Step 2: Unpack

Untar and move the downloaded Node Exporter binary

```
tar -xvf node_exporter-0.18.1.linux-amd64.tar.gz
mv node_exporter-0.18.1.linux-amd64 node_exporter-files
```

Step 3: Stop Node Exporter Service

The executable that we're replacing is in use, so we need to stop the service so it can be replaced.

```
sudo systemctl stop node_exporter.service
```

Step 4: Install Node Exporter

Copy `node_exporter` binary from `node_exporter-files` folder to `/usr/bin` and change the ownership to prometheus user.

```
sudo cp node_exporter-files/node_exporter /usr/bin/
sudo chown node_exporter:node_exporter /usr/bin/node_exporter
```

Step 5: Start Node Exporter Service

Restart the Node Exporter service

```
sudo systemctl start node_exporter.service
```

Step 6: Clean Up

Remove the download and temporary files

```
rm -rf node_exporter-0.18.1.linux-amd64.tar.gz node_exporter-files
```

Upgrade Process Exporter

Step 1: Download Latest Binary

Visit the Prometheus downloads page for the latest version. Copy the correct link and download the Process Exporter binary to the server that you will upgrade.

```
wget \
  https://github.com/ncabatoff/process-exporter/releases/download/v0.6.0/pro
cess-exporter-0.6.0.linux-amd64.tar.gz
```

Step 2: Unpack

Untar and move the downloaded Process Exporter binary

```
tar -xvf process-exporter-0.6.0.linux-amd64.tar.gz
mv process-exporter-0.6.0.linux-amd64 process_exporter-files
```

Step 3: Stop Process Exporter Service

The executable that we're replacing is in use, so we need to stop the service so it can be replaced.

```
sudo systemctl stop process_exporter.service
```

Step 4: Install Node Exporter

Copy `process_exporter` binary from `process_exporter-files` folder to `/usr/bin` and change the ownership to prometheus user.

```
sudo cp process_exporter-files/process-exporter /usr/bin/
sudo chown process_exporter:process_exporter /usr/bin/process-exporter
```

Step 5: Start Process Exporter Service

Restart the Node Exporter service

```
sudo systemctl start process_exporter.service
```

Step 6: Clean Up

Remove the download and temporary files

```
rm -rf process-exporter-0.6.0.linux-amd64.tar.gz process_exporter-files
```

Upgrade Couchbase Exporter

Step 1: Download Latest Binary

Download the Couchbase Exporter python code.

```
curl -L \
  https://github.com/couchbaselabs/cbprometheus_python/tarball/master > \
  couchbase_exporter.tar.gz
```

Step 2: Unpack

Untar and move the downloaded Couchbase Exporter code

```
mkdir -p couchbase_exporter
tar -xzf couchbase_exporter.tar.gz \
  -C couchbase_exporter --strip-components=1
```

Step 3: Install Python Dependencies

It is unlikely that the Python dependcies have changed, incase they have changed they'll need to be installed

```
sudo pip install -r ./couchbase_exporter/requirements
```

Step 4: Install Couchbase Exporter

Copy `couchbase_exporter` directory from `couchbase_exporter` folder to `/opt/couchbase_exporter` and change the ownership to the couchbase_exporter user.

```
sudo cp -R couchbase_exporter/* /opt/couchbase_exporter
sudo chown -R couchbase_exporter:couchbase_exporter /opt/couchbase_exporter
```

Step 5: Restart Emperor Service

Restart the Node Exporter service

```
sudo systemctl restart emperor.uwsgi.service
```

Step 6: Clean Up

Remove the download and temporary files

```
rm -rf couchbase_exporter*
```

Upgrade AlertManager

Step 1: Download Latest Binary

Visit the Prometheus downloads page for the latest version. Copy the correct link and download the AlertManager binary to the server that you will upgrade.

```
wget \
  https://github.com/prometheus/alertmanager/releases/download/v0.20.0/alert
manager-0.20.0.linux-amd64.tar.gz
```

Step 2: Unpack

Untar and move the downloaded Prometheus binary

```
tar -xvf alertmanager-0.20.0.linux-amd64.tar.gz
mv alertmanager-0.20.0.linux-amd64 alertmanager-files
```

Step 3: Stop AlertManager Service

The executable that we're replacing is in use, so we need to stop the service so it can be replaced.

```
sudo systemctl stop alertmanager.service
```

Step 4: Install AlertManager

Copy `prometheus` and `promtool` binary from prometheus-files folder to `/usr/bin` and change the ownership to prometheus user.

```
sudo cp prometheus-files/prometheus /usr/bin/
sudo cp prometheus-files/promtool /usr/bin/
sudo chown prometheus:prometheus /usr/bin/prometheus
sudo chown prometheus:prometheus /usr/bin/promtool
```

Step 5: Install Prometheus Libraries

Copy `alertmanager` and `amtool` binary from `alertmanager-files` folder to `` `/usr/bin `` and change the ownership to alertmanager user.

```
sudo cp alertmanager-files/alertmanager /usr/bin/
sudo cp alertmanager-files/amtool /usr/bin/
sudo chown alertmanager:alertmanager /usr/bin/alertmanager
sudo chown alertmanager:alertmanager /usr/bin/amtool
```

Step 6: Start Prometheus

Restart the AlertManager service

```
sudo systemctl start alertmanager.service
```

Step 7: Clean Up

Remove the download and temporary files

```
rm -rf alertmanager-files
```

Upgrade Grafana

Step 1: Yum or Rpm

```
sudo yum update grafana
```