

Monitoring with Grafana

Estimated time needed: 30 minutes

Welcome to the **Monitoring with Grafana** lab. In this lab you will learn to use Grafana as a visualization tool and dashboard for Prometheus.

Learning Objectives

In this hands-on lab, you will:

- Deploy Prometheus to OpenShift
- Deploy Grafana to OpenShift
- Connect Prometheus as a datasource for Grafana
- Create a dashboard with Grafana

Prerequisites

There is some setup to do before you can proceed with the lab. In this prerequisite step you will clone the GitHub repository that contains the Kubernetes manifests needed to deploy **Prometheus** and **Grafana** to an **OpenShift** cluster in the lab environment.

Your Task

1. First, if you do not already have a terminal open, open a terminal using the top menu item **Terminal** -> **New Terminal** and **cd** (change directory) into the default `/home/project` folder:

```
1. 1
1. cd /home/project
Copied! Executed!
```

2. Then use the `git clone` command to clone [this](#) repository:

```
1. 1
1. git clone https://github.com/ibm-developer-skills-network/ondaw-prometheus-grafana-lab.git
Copied! Executed!
```

3. Finally, change into the `ondaw-prometheus-grafana-lab` folder:

```
1. 1
1. cd ondaw-prometheus-grafana-lab
Copied! Executed!
```

4. (Optional) Make your command prompt shorter with this bash command:

```
1. 1
1. export PS1="\[\033[01;32m\]\u\[\033[00m\]:\[\033[01;34m\]\W\[\033[00m\]\$ "
Copied! Executed!
```

You are now ready to start the lab.

Step 1: Deploy node exporters

In this step, you will deploy 3 Node Exporters, which will be used as monitoring targets.

Your Task

1. Use the `oc create deployment` command to deploy 3 node exporters, named `node-exporter1`, `node-exporter2`, and `node-exporter3`, all listening on port 9100.

```
1. 1
2. 2
3. 3
1. oc create deployment node-exporter1 --port=9100 --image=bitnami/node-exporter:latest
2. oc create deployment node-exporter2 --port=9100 --image=bitnami/node-exporter:latest
3. oc create deployment node-exporter3 --port=9100 --image=bitnami/node-exporter:latest
Copied! Executed!
```

2. Next, use the `oc expose` command to create services that expose the 3 node exporters so that Prometheus can communicate with them:

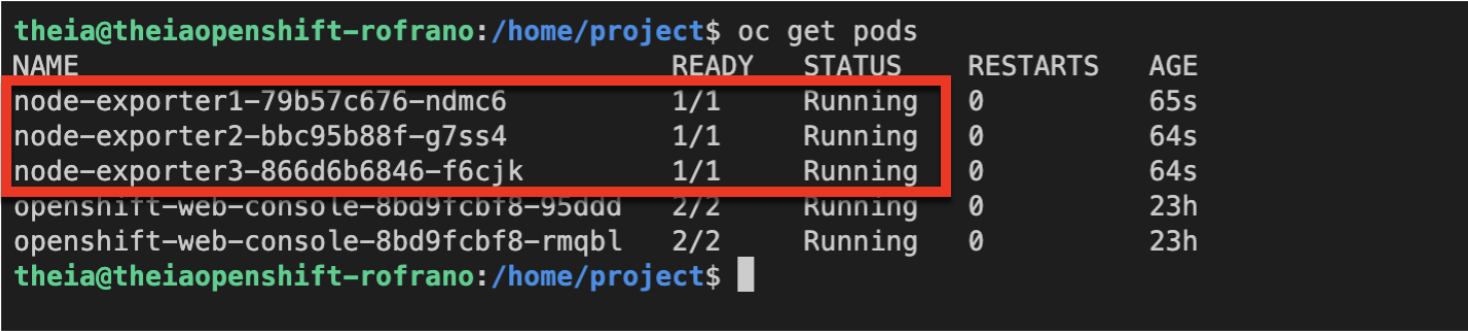
```
1. 1
2. 2
3. 3
1. oc expose deployment node-exporter1 --port=9100 --type=ClusterIP
2. oc expose deployment node-exporter2 --port=9100 --type=ClusterIP
3. oc expose deployment node-exporter3 --port=9100 --type=ClusterIP
Copied! Executed!
```

3. Check that the pods are up and running with the following `oc` command:

```
1. 1
1. oc get pods
Copied! Executed!
```

Results

You should see that all of the pods for the three node-exporters are in a running state like the image below:



You are now ready to deploy Prometheus.

Step 2: Deploy Prometheus

In this step, you will configure and deploy Prometheus. While normally you would modify configuration files to configure Prometheus, this is not the case for Kubernetes. For a Kubernetes environment the proper approach is to use a ConfigMap. This makes it easy to change the configuration later. You will find the configuration files from which to make the ConfigMap in a folder named `./config`.

You will also need Kubernetes manifests to describe the Prometheus deployment and to link the ConfigMap with the Prometheus. These manifests can be found in the `./deploy` folder.

Your Task

Use the following commands to create a ConfigMap and deploy Prometheus:

1. First, create a ConfigMap called `prometheus-config` that is needed by Prometheus from the `prometheus.yml` and the `alerts.yml` file in the `./deploy` folder.

```
1. 1
2. 2
3. 3
1. oc create configmap prometheus-config \
2.   --from-file=prometheus=./config/prometheus.yml \
3.   --from-file=prometheus-alerts=./config/alerts.yml
```

Copied! Executed!

You should see a message that the `prometheus-config` was created.

2. Now deploy Prometheus using the `deploy/prometheus-deployment.yml` deployment manifest.

```
1. 1
1. oc apply -f deploy/prometheus-deployment.yml
```

Copied! Executed!

You should see a message that the `prometheus-volume-claim`, `deployment.apps/prometheus`, and `service/prometheus` were all created.

3. Finally, use the `oc` command to check that the `prometheus` pod is running.

```
1. 1
1. oc get pods -l app=prometheus
```

Copied! Executed!

Results

You should see that the `prometheus` pod is in a running state.

```
theia:ondaw-prometheus-grafana-lab$ oc get pods -l app=prometheus
NAME                                READY   STATUS    RESTARTS   AGE
prometheus-69765db9d5-nnj94        1/1     Running   0           10s
```

You are now ready to deploy Grafana.

Step 3: Deploy Grafana

Now that you have 3 node exporters to emit metrics, and Prometheus to collect them, it is time to add Grafana for dashboarding. You will deploy Grafana into OpenShift and create a route which will allow you to open up the Grafana web UI and work with it.

Your Task

1. First, deploy Grafana using the `deploy/grafana-deployment.yml` deployment manifest.

```
1. 1
1. oc apply -f deploy/grafana-deployment.yml
```

Copied! Executed!

You should see a message that both the `grafana` deployment and service have been created.

2. Next, use the `oc expose` command to expose the `grafana` service with an OpenShift route. Routes are a special feature of OpenShift that makes it easier to use for developers.

```
1. 1
1. oc expose svc grafana
```

Copied! Executed!

You should see the message: "route.route.openshift.io/prometheus exposed"

3. Use the following `oc patch` command to enable TLS and the `https://` protocol for the route.

```
1. 1
1. oc patch route grafana -p '{"spec":{"tls":{"termination":"edge","insecureEdgeTerminationPolicy":"Redirect"}}}'
```

Copied! Executed!

4. Then use the `oc get routes` command to check the URL that the route was assigned. You will be able to interact with Grafana using this URL.

```
1. 1
1. oc get routes
```

Copied! Executed!

5. Finally, use the `oc` command to check that the `grafana` pod is running.

```
1. 1
1. oc get pods -l app=grafana
```

Copied! Executed!

Results

You should see that the `grafana` pod is in a running state.

```
theia:ondaw-prometheus-grafana-lab$ oc get pods -l app=grafana
NAME                                READY    STATUS    RESTARTS   AGE
grafana-6dc6566c6b-9zkpg           1/1      Running   0           4m4s
theia:ondaw-prometheus-grafana-lab$
```

You should also see the URL of the route that you created.

```
theia:ondaw-prometheus-grafana-lab$ oc get routes
NAME                                HOST/PORT
PATH    SERVICES    PORT    TERMINATION    WILDCARD
grafana  grafana-sn-labs-rofrano.labs-prod-openshift-san-a45631dc5778dc6371c67d206ba9ae5
omain.cloud    grafana    3000    None
```

You are now ready to configure Grafana.

Step 4: Log in to Grafana

Now that Prometheus and Grafana are deployed and running, it is time to configure Grafana. In order to do this, you will need the URL from the route that you created in the last step.

Note: You do not need a Grafana account in order to complete this step.

Your Task

- 1. Use the `oc describe` command along with `grep` to extract the URL of the Requested Host for the `grafana` route:

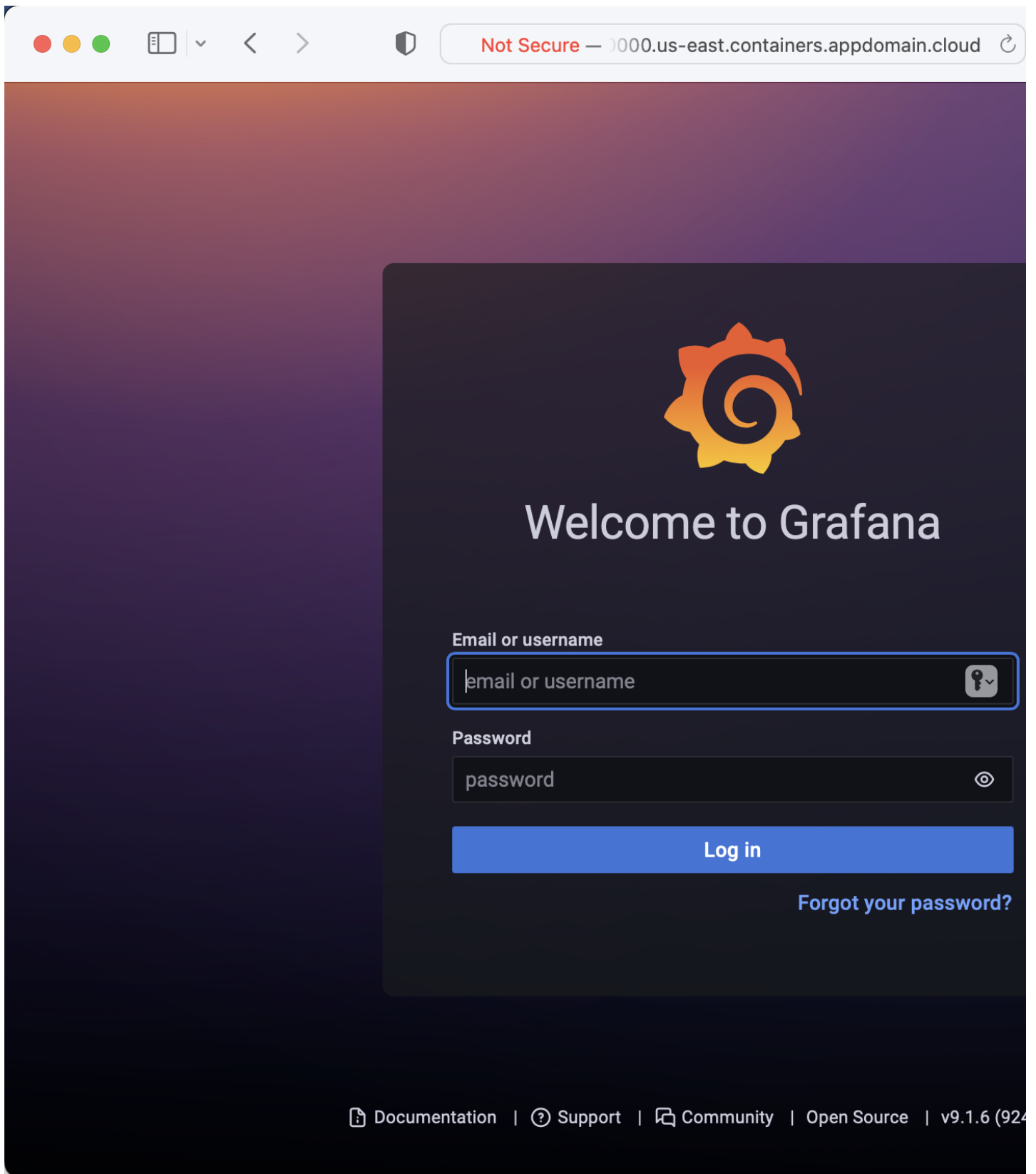
```
1. 1
1. oc describe route grafana | grep "Requested Host:"
```

Copied! Executed!

You will see the words "Requested Host:" in red followed by a URL. It will start with the word "grafana-" followed by the rest of the URL.

- 2. Copy the URL after the words "Requested Host:" to the clipboard and paste it into a new web browser window outside of the lab environment.

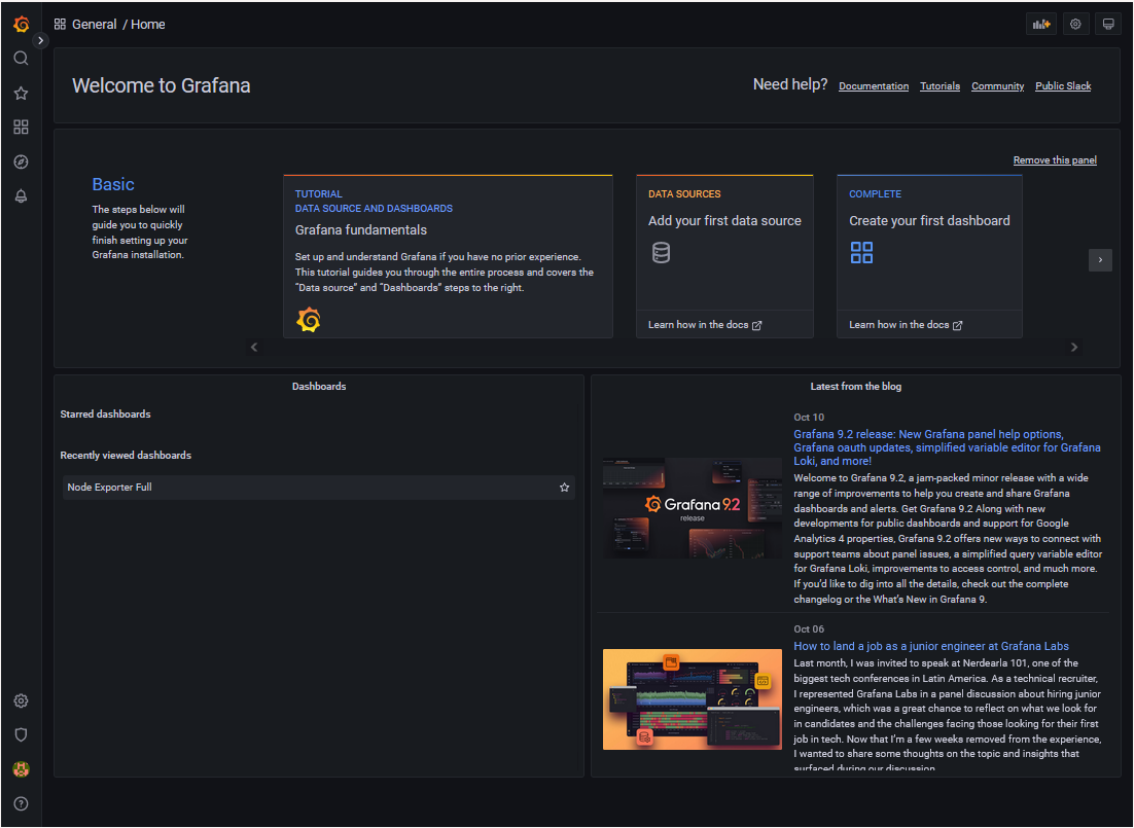
You should see the Grafana log in page:



3. From the Grafana login screen, log in with the default userid `admin` and default password `admin`, and then click the **[Log in]** button. You will be prompted to change your password. You can press the skip link to bypass that for now.

Results

You should see the Grafana home page:

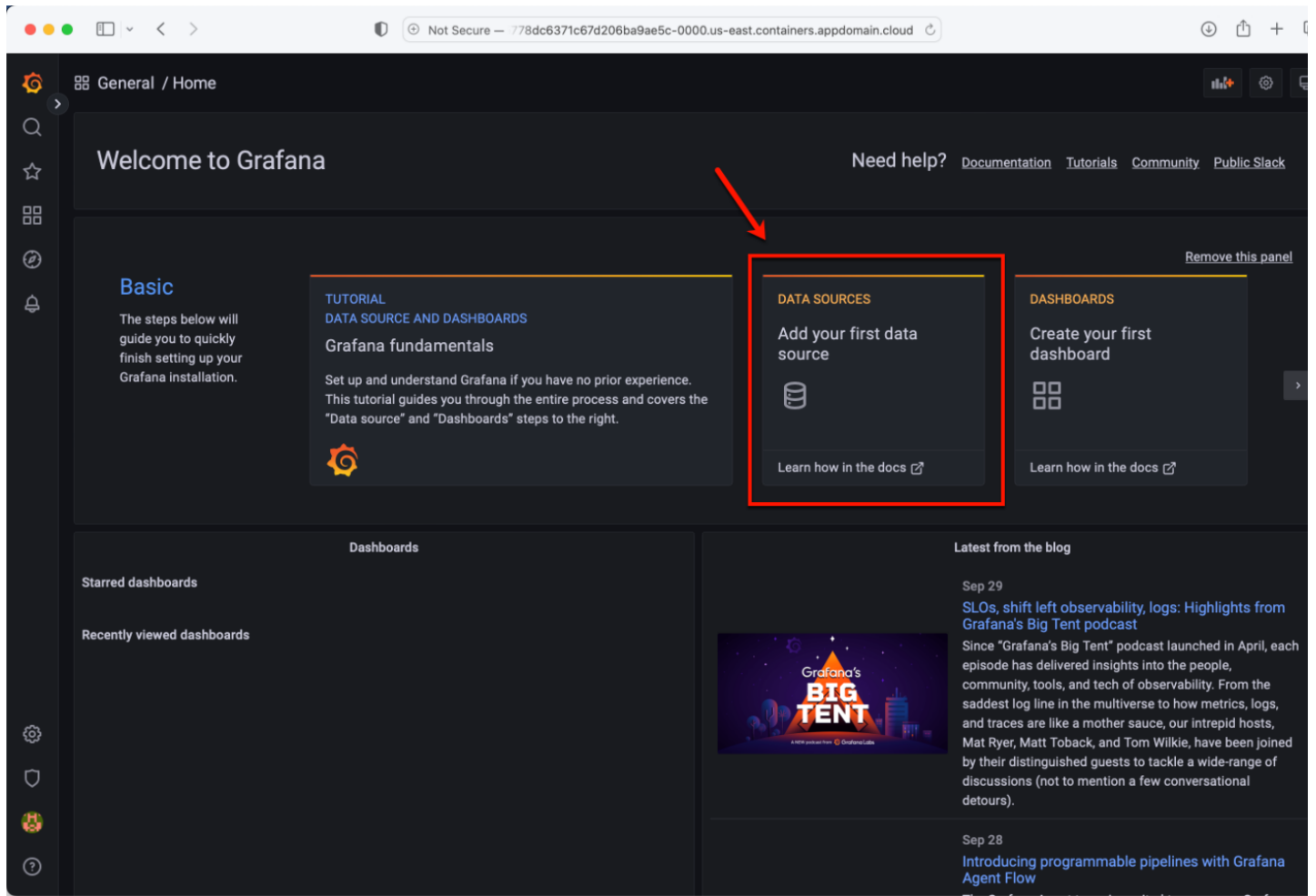


Step 5: Configure Grafana











Once you have logged in to Grafana, you can configure Prometheus as your datasource.


Your Task

- 1. At the home page, select the data sources icon.



2. At the Add data source page select Prometheus.






Add data source

Choose a data source type


Time series databases



Prometheus

Open source time series database & alerting


Core



Graphite

Open source time series database


Core



InfluxDB

Open source time series database

Core




OpenTSDB

Open source time series database

Core


Logging & document databases



Loki

Like Prometheus but for logs. OSS logging solution from Grafana Labs

Core



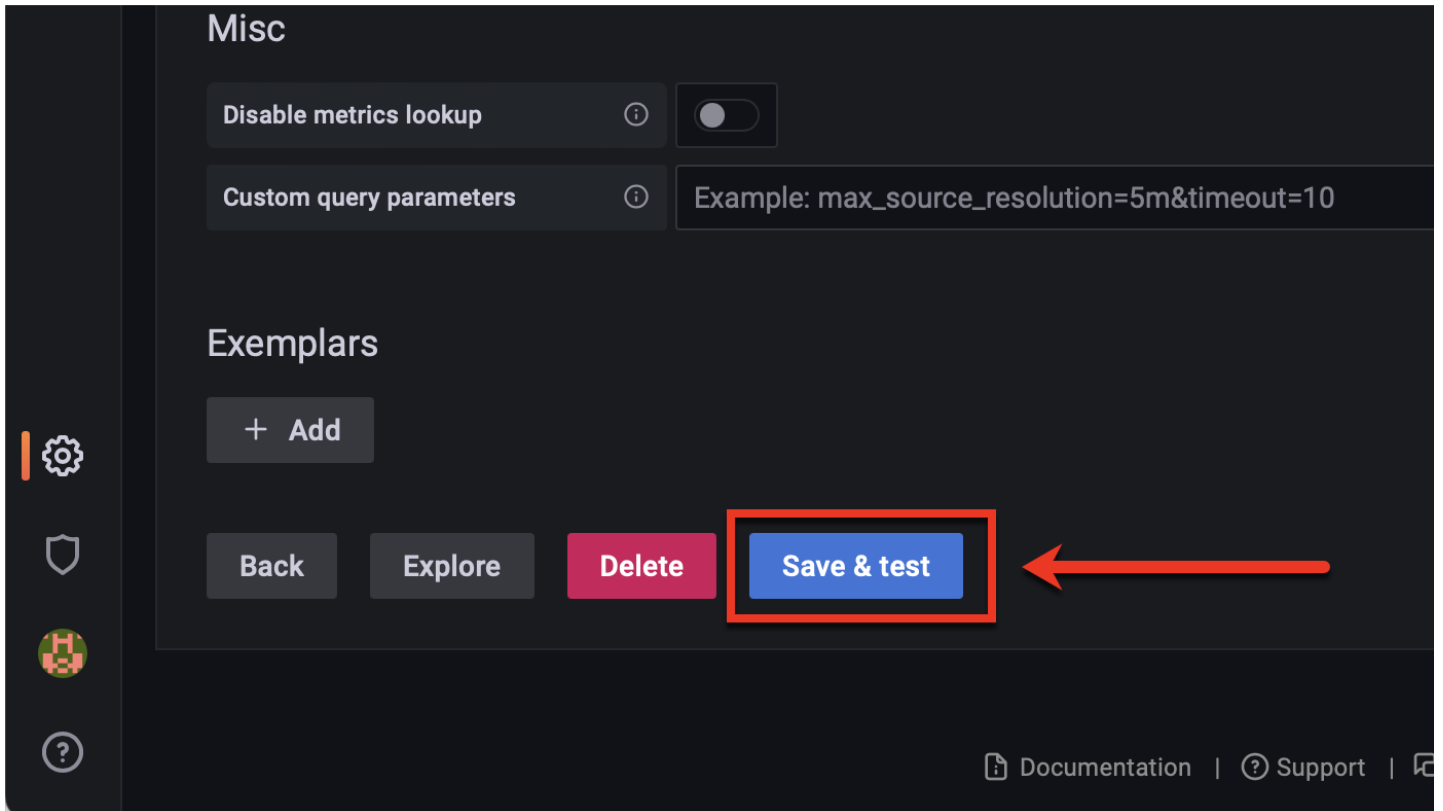
Elasticsearch

Open source logging & analytics database

Core

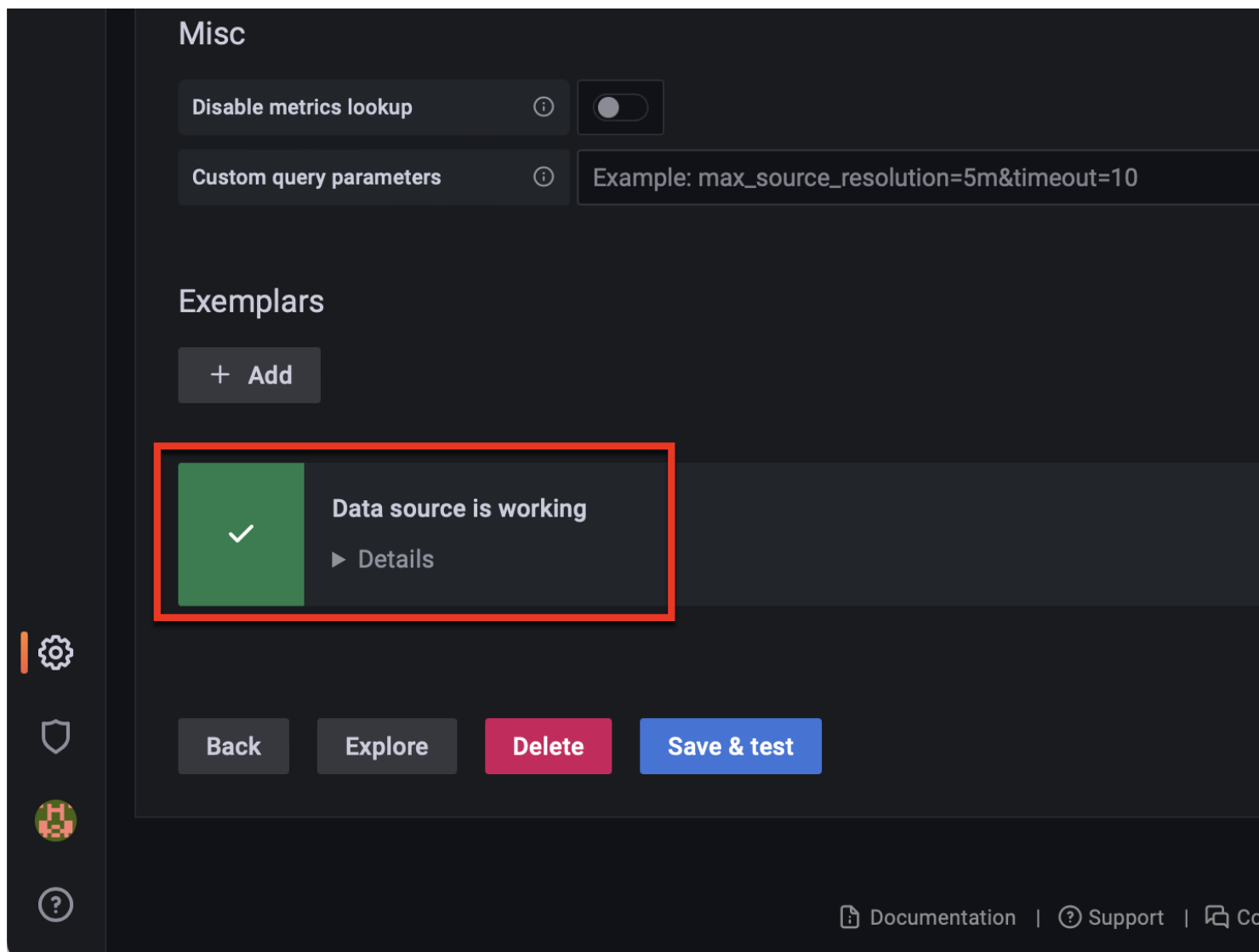
3. On the Prometheus configuration page, set the Prometheus URL to `http://prometheus:9090` which is the name and port of the Prometheus service in OpenShift.

4. Scroll down to the bottom of the screen and click the Save & test button.



Results

You should see a message, confirming that the data source is working:

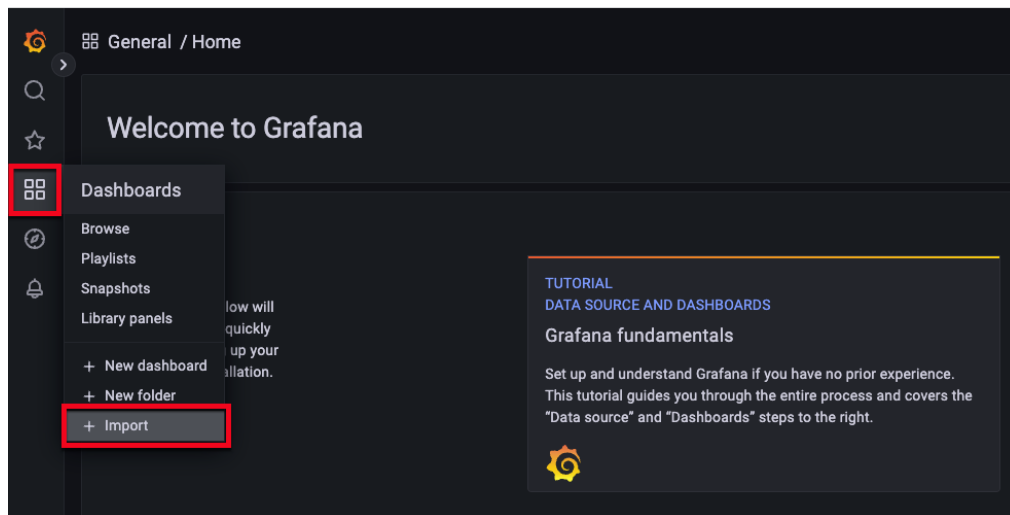


Step 6: Create a Dashboard

Now you can create your first dashboard. For this lab, you will use a precreated template provided by Grafana Dashboard. This template is identified by the id 1860.

Your Task

1. On the Grafana homepage, click the Dashboards icon, and select + Import from the menu to start creating the dashboard.



2. Next, enter the template identifier id 1860 in the space provided and click the [Load] button to import that dashboard from Grafana.

Dashboards

Manage dashboards and folders

Browse

Playlists

Snapshots

Library panels

Upload JSON file

Import via grafana.com

1860

Load

3. You will see the default name for the template, **Node Exporter Full**, displayed. You are allowed to change it if you want to. The change will be local and valid only for your instance.

Dashboards

Manage dashboards and folders

Browse

Playlists

Snapshots

Library panels

Importing dashboard from Grafana.com

Published by

rfraile

Updated on

2022-09-11 20:25:51

Options

Name

Node Exporter Full

Folder

General

Unique identifier (UID)

The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

rYdddIPWk

Change uid

Prometheus

Select a Prometheus data source

Prometheus (default)

4. At the bottom of the page, choose Prometheus (Default) as the data source and click Import.

Prometheus

Prometheus

Import

Cancel

This completes importing a dashboard. Next you will see what the dashboard allows you to monitor.

Step 7: View the Dashboard

You should now see the **General / Node Exporter Full** dashboard which shows **CPU Busy**, **Sys Load**, **RAM used** and other information. This dashboard will allow you to get a broad overview of how the system is performing, as well as drilling down into individual nodes to see how they are performing. Now, try a few things out.

Things to try

- 1. You can hover over the graph to get specific information at a given time.

about:blank

13/16



2. Choose a different host to observe the details about that host.

General / Node Exporter Full ☆ 🔗

datasource

default ▾

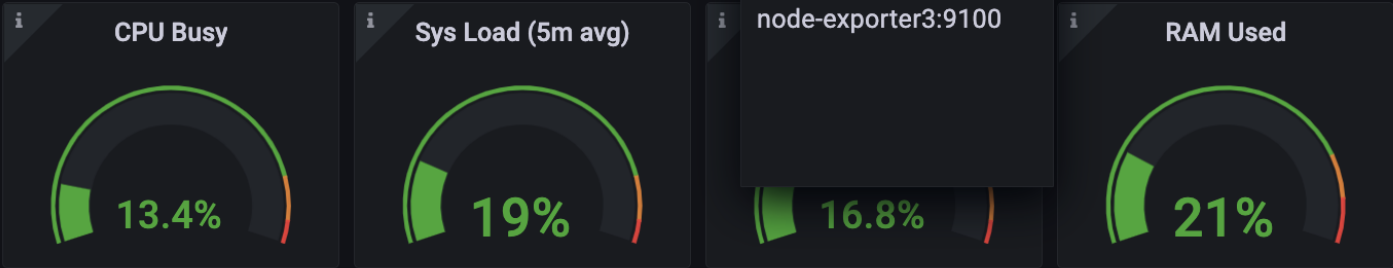
Job

prometheus ▾

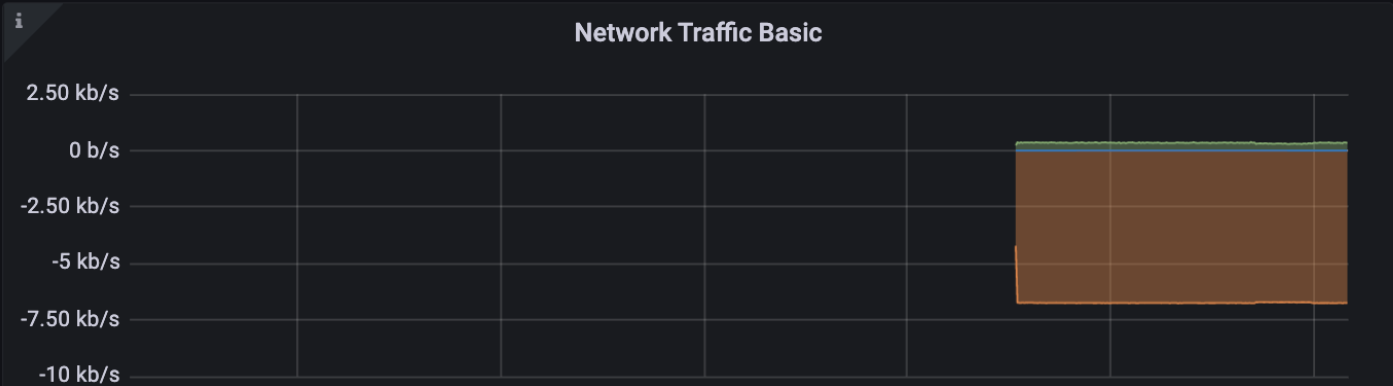
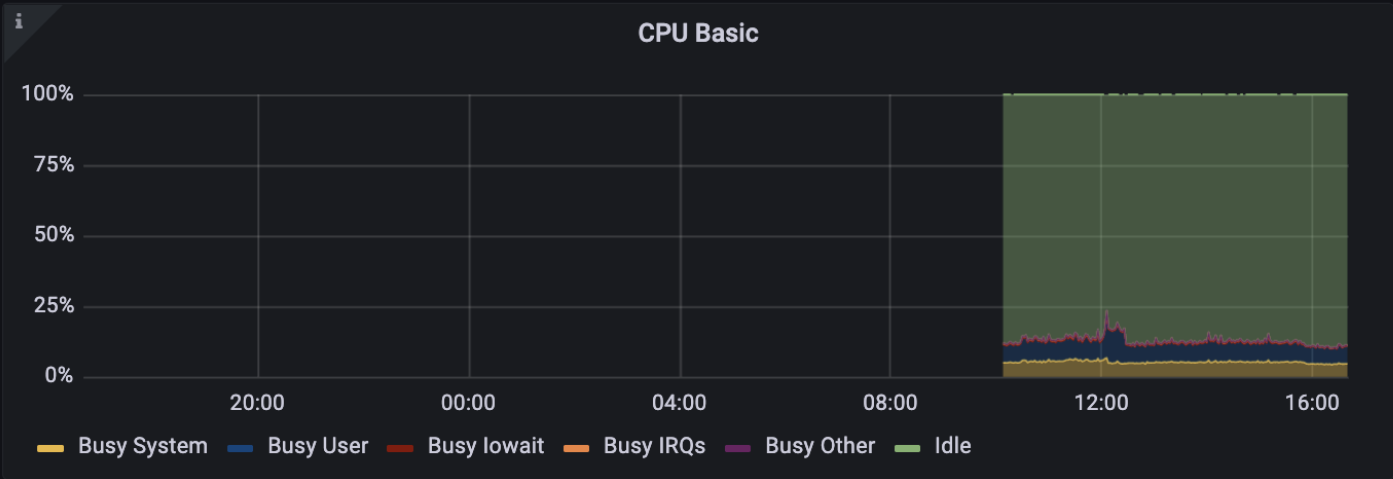
Host:

Enter variable value

Quick CPU / Mem / Disk



Basic CPU / Mem / Net / Disk



3. Choose to visualize the metrics over different time ranges.

General / Node Exporter Full

datasource

default

Job

prometheus

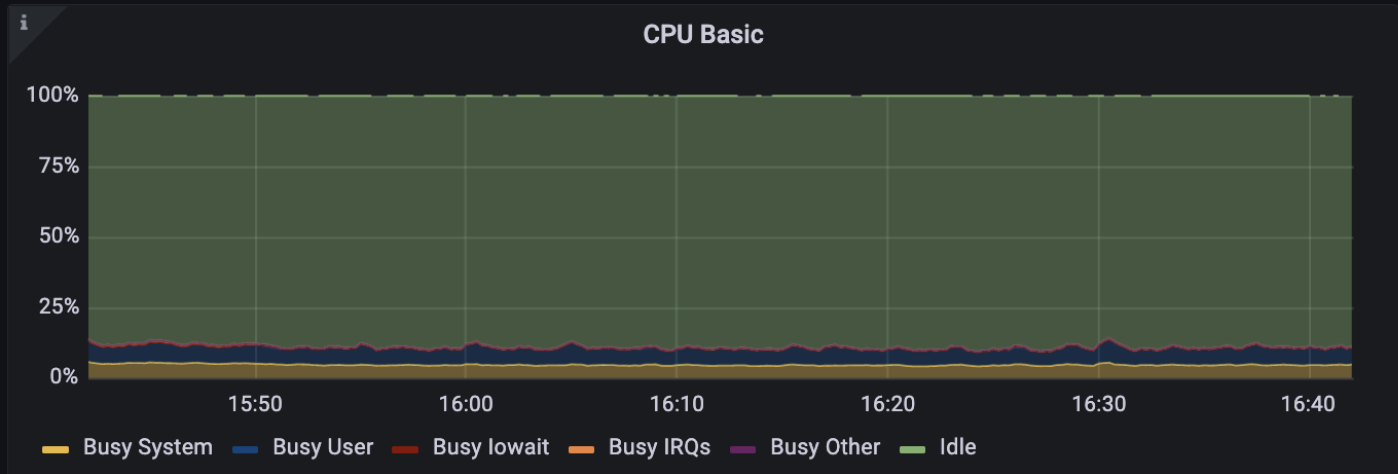
Host:

node-exporter1:9100

Quick CPU / Mem / Disk



Basic CPU / Mem / Net / Disk



As you can imagine, this could be monitoring an application that you have deployed into Kubernetes or OpenShift so that you can see the CPU, Memory, and other metrics to determine the overall health of your application.

Conclusion

Congratulations! You have completed the lab on **Monitoring in Action with Grafana**. You are now well on your way to monitoring your own applications to ensure they are running properly.

In this lab you deployed three node exporters and use Prometheus metrics to monitor them. You also learned how to instrument a Python Flask application, deploy it in Docker, and modify the Prometheus configuration to start monitoring this new application.

Next Steps

Your next challenge is to set up Prometheus in your development environment to monitor your applications. Use the **Prometheus Flask exporter** to instrument one of your Python Flask applications. Then use some of the queries you have learned in this lab to check on the health and performance of your application.

Author(s)

[Lavanya T S](#)
[John J. Rofrano](#)

Other Contributor(s)

Pallavi Rai

Changelog

Date	Version	Changed by	Change Description
2022-09-12	0.1	Lavanaya T S	Initial version created
2022-09-30	0.2	John Rofrano	Added additional content and OpenShift
2022-10-03	0.3	Lavanaya T S	Added Grafana Monitoring
2022-10-12	0.4	Amy Norton	ID Review