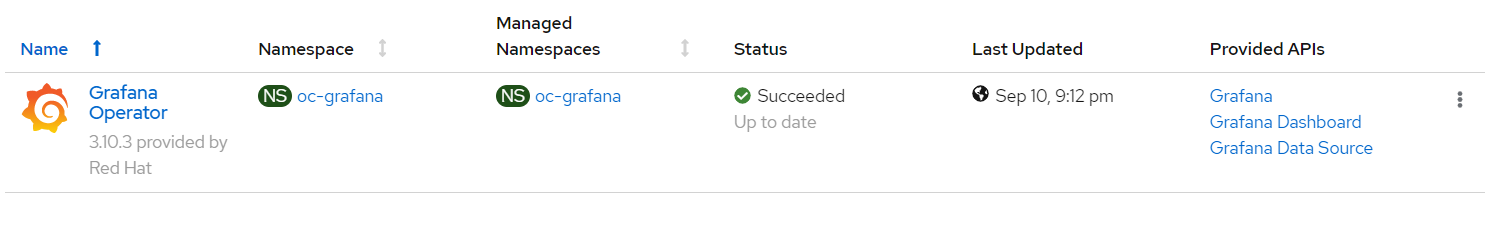
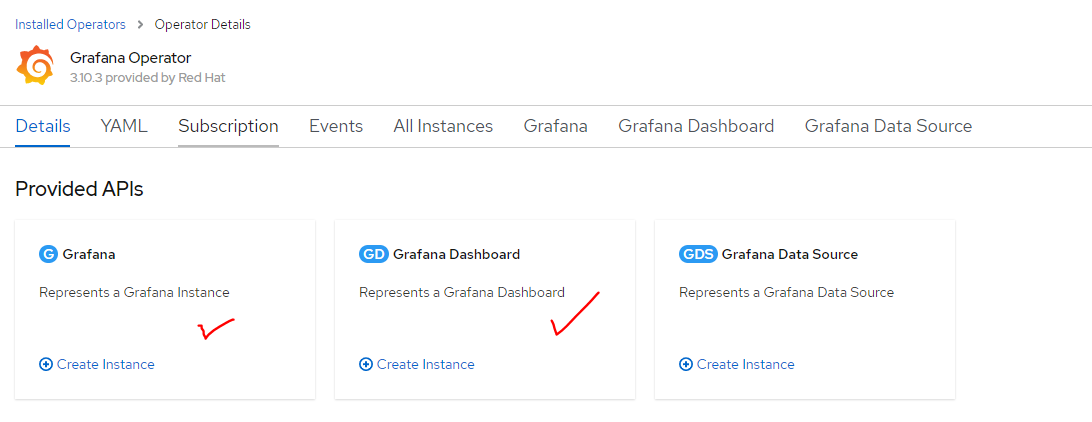
Installing Grafana with ROKS in-built Prometheus

In the beginning, after installing ROKS, we get a default Prometheus and a Grafana. Though Prometheus works well, Grafana is not upto the requirement for a production level system. So, we need to enable a bunch of settings in ROKS and install a Grafana for customizations.

Let’s start with installing a Grafana. The community-powered Grafana cannot be deployed to the existing openshift-monitoring namespace, so we will create a new namespace (e.g., oc-grafana) to deploy into instead. Navigate to OperatorHub and select the community-powered Grafana Operator. Press Continue to accept the disclaimer, press Install, and press Subscribe to accept the default configuration values and deploy to the oc-grafana namespace. Within some time, the Grafana operator will be made available in the oc-grafana namespace.



From Installed Operators, select the Grafana Operator. For the Grafana resource, press Create Instance to create a new Grafana instance.



In the Grafana instance YAML, make a note of the default username (e.g., root) and password (we kept deroksveu123) to log in, and press Create.

**Connecting Grafana with default Prometheus**

But now if we try out Grafana, we will see that as a empty welcome dashboard. It won’t be able to pull and show metrics from ROKS’s default Prometheus instance. We need some efforts for it.

The grafana-serviceaccount service account was created alongside the Grafana instance. We will grant it the cluster-monitoring-view cluster role.

We need to issue two commands for this purpose

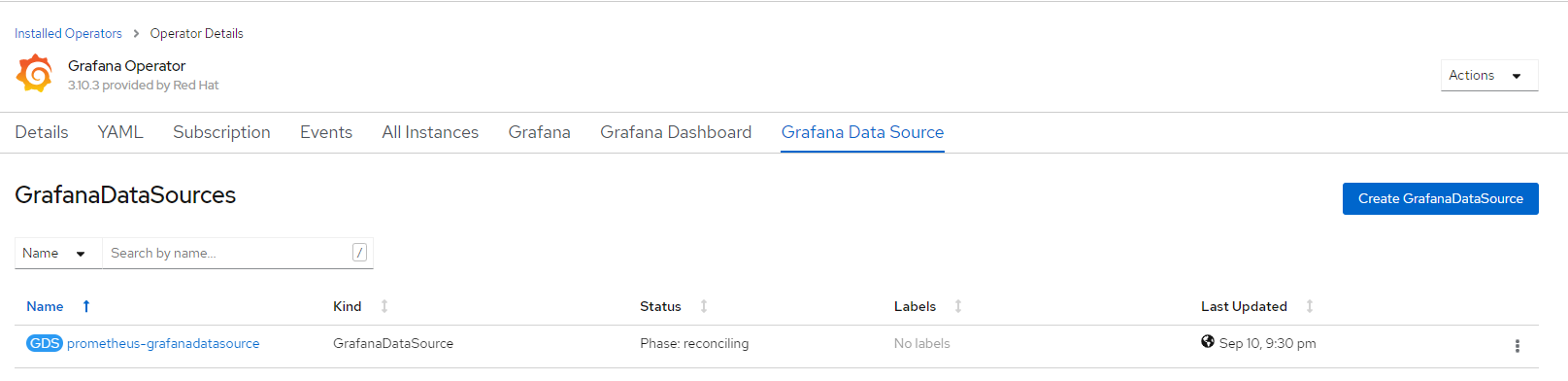
**oc adm policy add-cluster-role-to-user cluster-monitoring-view -z grafana-serviceaccount**

The bearer token for this service account is used to authenticate access to Prometheus in the openshift-monitoring namespace. The following command will display this token.

**oc serviceaccounts get-token grafana-serviceaccount -n oc-grafana**

From the Grafana Data Source resource, press Create Instance, and navigate to the YAML view. In the below YAML, substitute ${BEARER\_TOKEN} with the output of the command above, copy the YAML, and press Create.

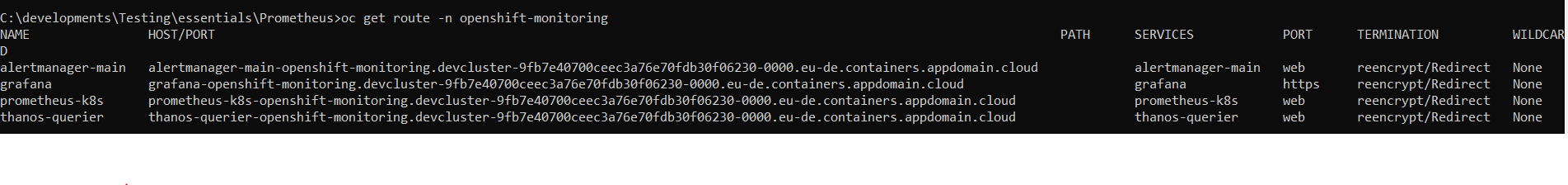
Find the Grafana Operator and move into Grafana Data Source tab



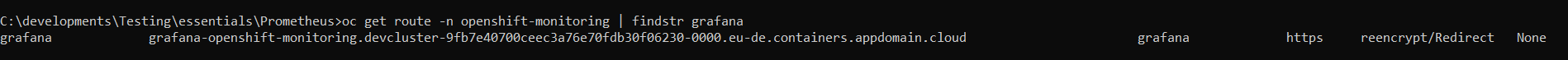
Click on the data source and replace the default value with below script, where bearer token to replace with same retrieved from service account above



Now it’s time to access the route for Grafana. Please search all routes in openshift-monitoring namespace, where we have Grafana, Prometheus and Alert Manager routes.



We will choose the Grafana route



Next, we will import some common dashboards into our Grafana dashboard

Cluster Stats:

https://grafana.com/grafana/dashboards/7685

Grafana All Pods:

https://grafana.com/grafana/dashboards/6588

Grafana Persistent Volumes:

https://grafana.com/grafana/dashboards/13646

We will do some more on demand, like for some common middleware performances of MongoDB / MySQL / Rabbit MQ / Redis / PostgreSQL etc. Since at start of configurations, it is not available in system, we will update it latter.

Grafana Dashboard Document References

**Reference Link**: https://www.redhat.com/en/blog/custom-grafana-dashboards-red-hat-openshift-container-platform-4

**Enhancing in built Prometheus instance**

We will enhance the Prometheus for some important enablement

**Add storage**

**Add Metrics Retention Time**

**Add Email Configurations**

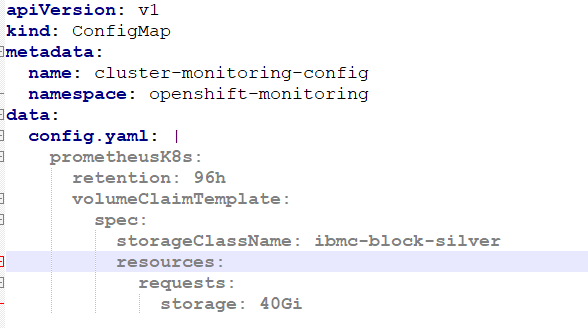
**Add Security**

To start all these, we need to check whether we have cluster-monitoring-config configmap inside openshift-monitoring namespace.

**oc -n openshift-monitoring get configmap cluster-monitoring-config**

We will get empty output. We will create a configmap and deploy the same in cluster

Please find the below for an example configmap file



We will enhance it for some more components, like Alert manager, etc.

**Source Reference**

https://docs.openshift.com/container-platform/4.5/monitoring/cluster\_monitoring/configuring-the-monitoring-stack.html#modifying-retention-time-for-prometheus-metrics-data\_configuring-monitoring

<https://bitnami.com/stack/grafana/helm>

<https://github.com/bitnami/charts/tree/master/bitnami/grafana/#installing-the-chart>

Helm3 Command for Custom Grafana:

helm3 upgrade --install my-grafana bitnami/grafana --set imageRenderer.enabled=true,datasources.secretName=datasource-secret,admin.user=admin,admin.password=admin,service.type=NodePort,persistence.storageClass=ibmc-block-silver --namespace oc-grafana

Grafana Dashboard

root / deroksveu123

Reference Link: https://www.redhat.com/en/blog/custom-grafana-dashboards-red-hat-openshift-container-platform-4

Grafana Route: grafana-route-oc-grafana.devcluster-9fb7e40700ceec3a76e70fdb30f06230-0000.eu-de.containers.appdomain.cloud

Grafana Dashboard Importing Site:

https://grafana.com/grafana/dashboards/7685

Grafana Kubernetes Cluster Dashboard:

https://grafana.com/grafana/dashboards/6417

Grafana All Pods:

https://grafana.com/grafana/dashboards/6588

Grafana Persistent Volumes:

https://grafana.com/grafana/dashboards/13646

Kubernetes Pods (Prometheus):

<https://grafana.com/grafana/dashboards/6336>

To set some variables, we need to create a separate Grafana-config map outside helm chart, which can be used for injecting some parameters, like

[auth]

login\_maximum\_lifetime\_duration = 30m

[smtp]

enabled = true

host = smtp.sendgrid.net:587

user = apikey

# If the password contains # or ; you have to wrap it with trippel quotes. Ex """#password;"""

password = SG.PGddOciKTy2j2ddIun-QVg.aqjeE8dNs2odQE-b2SLbiXgwUkmCX8zko\_mdcO68PUM

;cert\_file =

;key\_file =

;skip\_verify = false

from\_address = radhamde@in.ibm.com

from\_name = SREROKSTeam

[emails]

welcome\_email\_on\_sign\_up = true

There are more settings

https://grafana.com/docs/grafana/latest/administration/configuration/#login\_maximum\_inactive\_lifetime\_duration

After every change in grafa-config config map and deploying the same, pod need to be restarted to let it take effect.