Monitoring using Prometheus and Grafana

What is Prometheus?

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open source project and maintained independently of any company. Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.

What is 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. In plain English, it provides you with tools to turn your time-series database (TSDB) data into beautiful graphs and visualizations.

Is helm installed?

We will use **helm** to install Prometheus & Grafana monitoring tools for this chapter.

```
# add prometheus Helm repo
helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts

# add grafana Helm repo
helm repo add grafana https://grafana.github.io/helm-charts
```

Deploy Prometheus

First we are going to install Prometheus. In this example, we are primarily going to use the standard configuration, but we do override the storage class. We will use gp2 EBS volumes for simplicity and demonstration purpose. When deploying in production, you would use io1 volumes with desired IOPS and increase the default storage size in the manifests to get better performance. Run the following command:

```
kubectl create namespace prometheus

helm install prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set alertmanager.persistentVolume.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
```

Make note of the prometheus endpoint in helm response (you will need this later). It should look similar to below:

```
The Prometheus server can be accessed via port 80 on the following DNS
name from within your cluster:
prometheus-server.prometheus.svc.cluster.local
```

Check if Prometheus components deployed as expected

```
kubectl get all -n prometheus
```

You should see response similar to below. They should all be Ready and Available

```
NAME                                             READY   STATUS
RESTARTS   AGE
pod/prometheus-alertmanager-868f8db8c4-67j2x     2/2     Running   0
78s
pod/prometheus-kube-state-metrics-6df5d44568-c4tkn   1/1     Running   0
78s
pod/prometheus-node-exporter-dh6f4               1/1     Running   0
78s
pod/prometheus-node-exporter-v8rd8               1/1     Running   0
78s
pod/prometheus-node-exporter-vcbjq               1/1     Running   0
78s
pod/prometheus-pushgateway-759689fbc6-hvjjm      1/1     Running   0
78s
pod/prometheus-server-546c64d959-qxbzd           2/2     Running   0
78s

NAME                                TYPE        CLUSTER-IP
EXTERNAL-IP   PORT(S)    AGE
service/prometheus-alertmanager     ClusterIP   10.100.38.47
<none>        80/TCP     78s
```

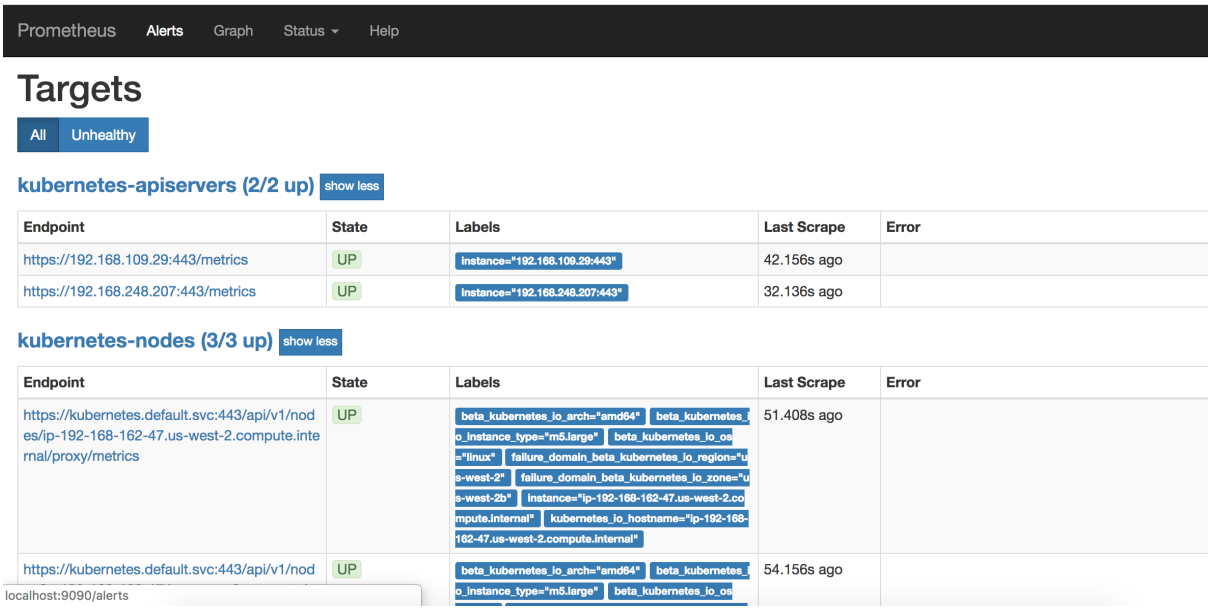
```
service/prometheus-kube-state-metrics   ClusterIP   10.100.165.139
<none>        8080/TCP   78s
service/prometheus-node-exporter        ClusterIP   None
<none>        9100/TCP   78s
service/prometheus-pushgateway          ClusterIP   10.100.150.237
<none>        9091/TCP   78s
service/prometheus-server               ClusterIP   10.100.209.224
<none>        80/TCP     78s

NAME                                      DESIRED   CURRENT   READY
UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-node-exporter   3         3         3       3
3           <none>          78s

NAME                                            READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/prometheus-alertmanager         1/1     1            1
78s
deployment.apps/prometheus-kube-state-metrics   1/1     1            1
78s
deployment.apps/prometheus-pushgateway          1/1     1            1
78s
deployment.apps/prometheus-server               1/1     1            1
78s

NAME                                                       DESIRED
CURRENT   READY   AGE
replicaset.apps/prometheus-alertmanager-868f8db8c4         1         1
1       78s
replicaset.apps/prometheus-kube-state-metrics-6df5d44568   1         1
1       78s
replicaset.apps/prometheus-pushgateway-759689fbc6          1         1
1       78s
replicaset.apps/prometheus-server-546c64d959               1         1
1       78s
```

In order to access the Prometheus server URL, we are going to use the kubectl port-forward command to access the application. In your terminal run:

```
kubectl port-forward -n prometheus deploy/prometheus-server 8080:9090
```

/targets

In the web UI, you can see all the targets and metrics being monitored by Prometheus:



Deploy Grafana

We are now going to install Grafana. For this example, we are primarily using the Grafana defaults, but we are overriding several parameters. As with Prometheus, we are setting the storage class to gp2, admin password, configuring the datasource to point to Prometheus and creating an external load balancer for the service.

Create YAML file called grafana.yaml with following commands:

```
mkdir ${HOME}/environment/grafana

cat << EoF > ${HOME}/environment/grafana/grafana.yaml
datasources:
  datasources.yaml:
    apiVersion: 1
    datasources:
    - name: Prometheus
      type: prometheus
      url: http://prometheus-server.prometheus.svc.cluster.local
      access: proxy
      isDefault: true
EoF
```

```
kubectl create namespace grafana

helm install grafana grafana/grafana \
    --namespace grafana \
    --set persistence.storageClassName="gp2" \
    --set persistence.enabled=true \
    --set adminPassword='EKS!sAWSome' \
    --values ${HOME}/environment/grafana/grafana.yaml \
    --set service.type=LoadBalancer
```

Run the following command to check if Grafana is deployed properly:

```
kubectl get all -n grafana
```

You should see similar results. They should all be Ready and Available

```
NAME                           READY   STATUS    RESTARTS   AGE
pod/grafana-f64dbbcf4-794rk    1/1     Running   0          55s
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)        AGE
service/grafana   LoadBalancer   10.100.60.167
aa0fa7322d86e408786cdd21ebcc461c-1708627185.us-east-2.elb.amazonaws.com
80:31929/TCP   55s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana    1/1     1            1           55s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/grafana-f64dbbcf4     1         1         1       55s
```

It can take several minutes before the ELB is up, DNS is propagated and the nodes are registered.

You can get Grafana ELB URL using this command. Copy & Paste the value into browser to access Grafana web UI.

```
export ELB=$(kubectl get svc -n grafana grafana -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}')

echo "http://$ELB"
```

When logging in, use the username **admin** and get the password hash by running the following:

```
kubectl get secret --namespace grafana grafana -o
jsonpath="{.data.admin-password}" | base64 --decode ; echo
```

Dashboards

Log in to Grafana

Log in to Grafana dashboard using credentials supplied during configuration.

You will notice that **‘Install Grafana’** & **‘create your first data source’** are already completed. We will import community created dashboard for this tutorial.

Cluster Monitoring Dashboard

For creating a dashboard to monitor the cluster:

- Click **'+'** button on left panel and select **‘Import’**.
- Enter **3119** dashboard id under Grafana.com Dashboard.
- Click **‘Load’**.
- Select **‘Prometheus’** as the endpoint under prometheus data sources drop down.
- Click **‘Import’**.

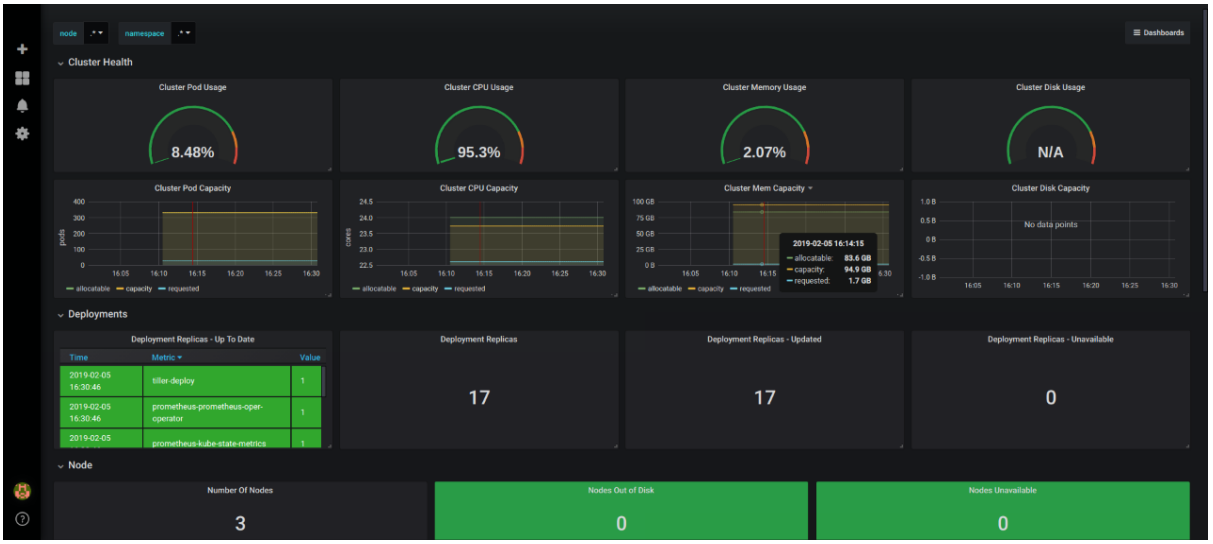
This will show monitoring dashboard for all cluster nodes

Pods Monitoring Dashboard

For creating a dashboard to monitor all the pods:

- Click **'+'** button on left panel and select **‘Import’**.
- Enter **6417** dashboard id under Grafana.com Dashboard.
- Click **‘Load’**.
- Enter **Kubernetes Pods Monitoring** as the Dashboard name.
- Click **change** to set the Unique identifier (uid).
- Select **‘Prometheus’** as the endpoint under prometheus data sources drop down.s

- Click **‘Import’**.



Cleanup

Uninstall Prometheus and Grafana

```
helm uninstall prometheus --namespace prometheus
```

```
kubectl delete ns prometheus
```

```
helm uninstall grafana --namespace grafana
kubectl delete ns grafana

rm -rf ${HOME}/environment/grafana
```