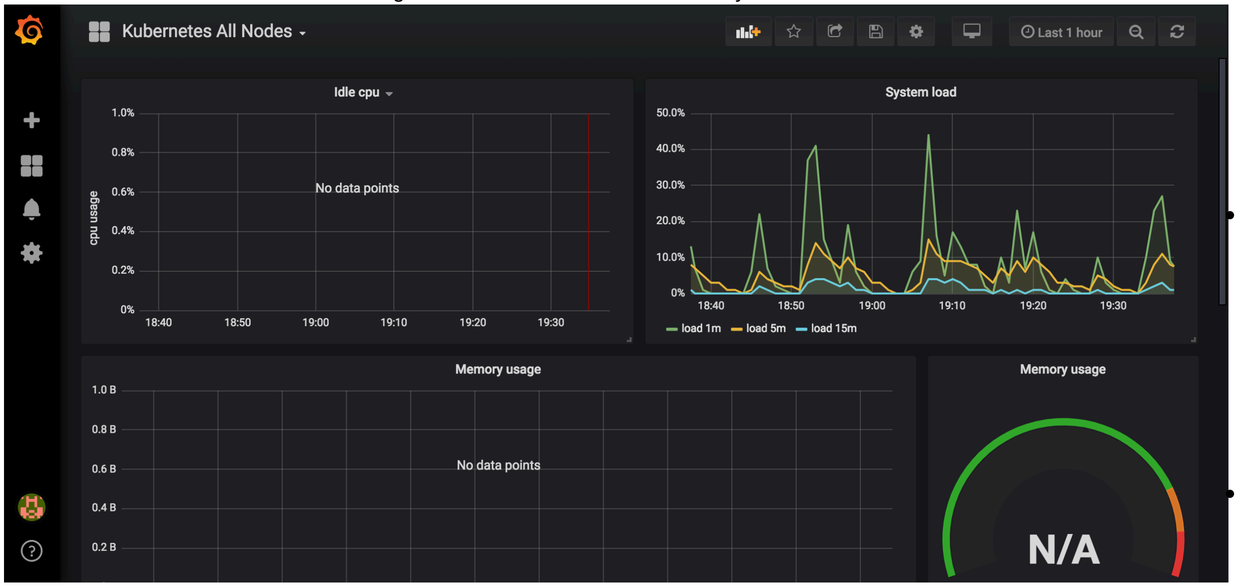
Deploying Prometheus and Grafana in a Gospel EKS cluster

We will be using Prometheus and Grafana to monitor Gospel Kubernetes clusters, the audience for these dashboards being system/database administrators that are interested in having an overview of the health of their system.



1. Pre-requisites

- Helm - you can set it up with these steps

  https://eksworkshop.com/beginner/060_helm/helm_intro/install/

- An EKS cluster running Gospel in its own separate namespace

2. Deploying Prometheus

Prometheus is used in order to gather metrics relevant for the Kubernetes cluster which will ultimately be used by Grafana to graph data.

```
kubectl create namespace prometheus

helm install stable/prometheus \
    --name 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-77cfdf85db-s9p48            2/2       Running
0          1m
pod/prometheus-kube-state-metrics-74d5c694c7-vqtjd      1/1       Running
0          1m
pod/prometheus-node-exporter-6dhpw                      1/1       Running
0          1m
pod/prometheus-node-exporter-nrfkn                      1/1       Running
0          1m
pod/prometheus-node-exporter-rtrm8                      1/1       Running
0          1m
pod/prometheus-pushgateway-d5fdc4f5b-dbmrg              1/1       Running
0          1m
pod/prometheus-server-6d665b876-dsmh9                   2/2       Running
0          1m

NAME                                        TYPE        CLUSTER-IP
EXTERNAL-IP   PORT(S)    AGE
service/prometheus-alertmanager             ClusterIP   10.100.89.154
<none>        80/TCP     1m
service/prometheus-kube-state-metrics       ClusterIP   None
<none>        80/TCP     1m
service/prometheus-node-exporter            ClusterIP   None
<none>        9100/TCP   1m
service/prometheus-pushgateway              ClusterIP   10.100.136.143
<none>        9091/TCP   1m
service/prometheus-server                   ClusterIP   10.100.151.245
<none>        80/TCP     1m

NAME                                          DESIRED   CURRENT   READY
UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-node-exporter   3         3         3
3            3           <none>          1m

NAME                                                DESIRED   CURRENT
UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-alertmanager             1         1         1
1            1m
deployment.apps/prometheus-kube-state-metrics       1         1         1
1            1m
deployment.apps/prometheus-pushgateway              1         1         1
1            1m
```

```
deployment.apps/prometheus-server               1         1         1
1           1m

NAME                                                       DESIRED
CURRENT   READY     AGE
replicaset.apps/prometheus-alertmanager-77cfdf85db         1         1
1         1m
replicaset.apps/prometheus-kube-state-metrics-74d5c694c7   1         1
1         1m
replicaset.apps/prometheus-pushgateway-d5fdc4f5b           1         1
1         1m
```

```
replicaset.apps/prometheus-server-6d665b876                1         1
1         1m
```

3. Deploying Grafana

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.

```
kubectl create namespace grafana
helm install stable/grafana \
    --name grafana \
    --namespace grafana \
    --set persistence.storageClassName="gp2" \
    --set adminPassword='EKS!sAWSome' \
    --set image.tag=6.6.1-ubuntu \
    --set datasources."datasources\.yaml".apiVersion=1 \
    --set datasources."datasources\.yaml".datasources[0].name=Prometheus
\
    --set datasources."datasources\.yaml".datasources[0].type=prometheus
\
    --set
datasources."datasources\.yaml".datasources[0].url=http://prometheus-ser
ver.prometheus.svc.cluster.local \
    --set datasources."datasources\.yaml".datasources[0].access=proxy \
    --set datasources."datasources\.yaml".datasources[0].isDefault=true
\
    --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-b9697f8b5-t9w4j   1/1       Running   0          2m

NAME              TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
service/grafana   LoadBalancer   10.100.49.172
abe57f85de73111e899cf0289f6dc4a4-1343235144.us-west-2.elb.amazonaws.com
80:31570/TCP   3m


NAME                      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE
AGE
deployment.apps/grafana   1         1         1            1
2m

NAME                                DESIRED   CURRENT   READY     AGE
replicaset.apps/grafana-b9697f8b5   1         1         1         2m
```

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
```

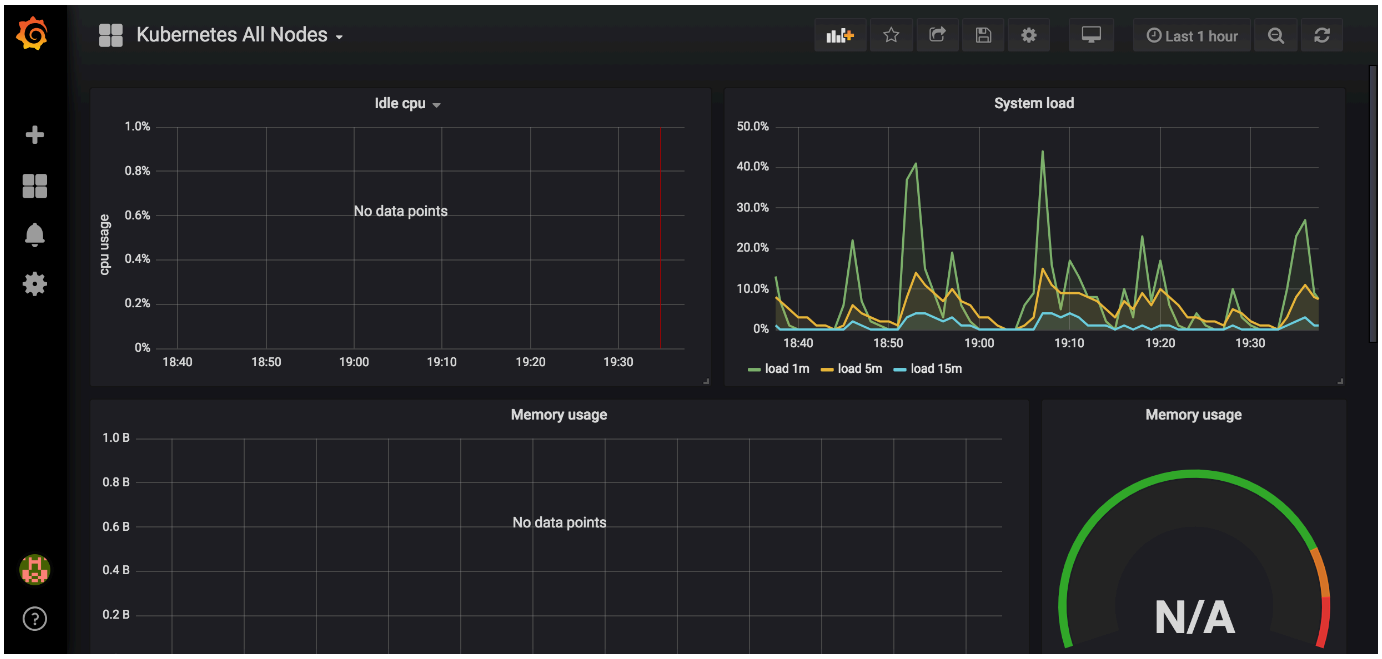
4. Creating Grafana dashboards

Log into the Grafana dashboard using credentials supplied during configuration (available above).

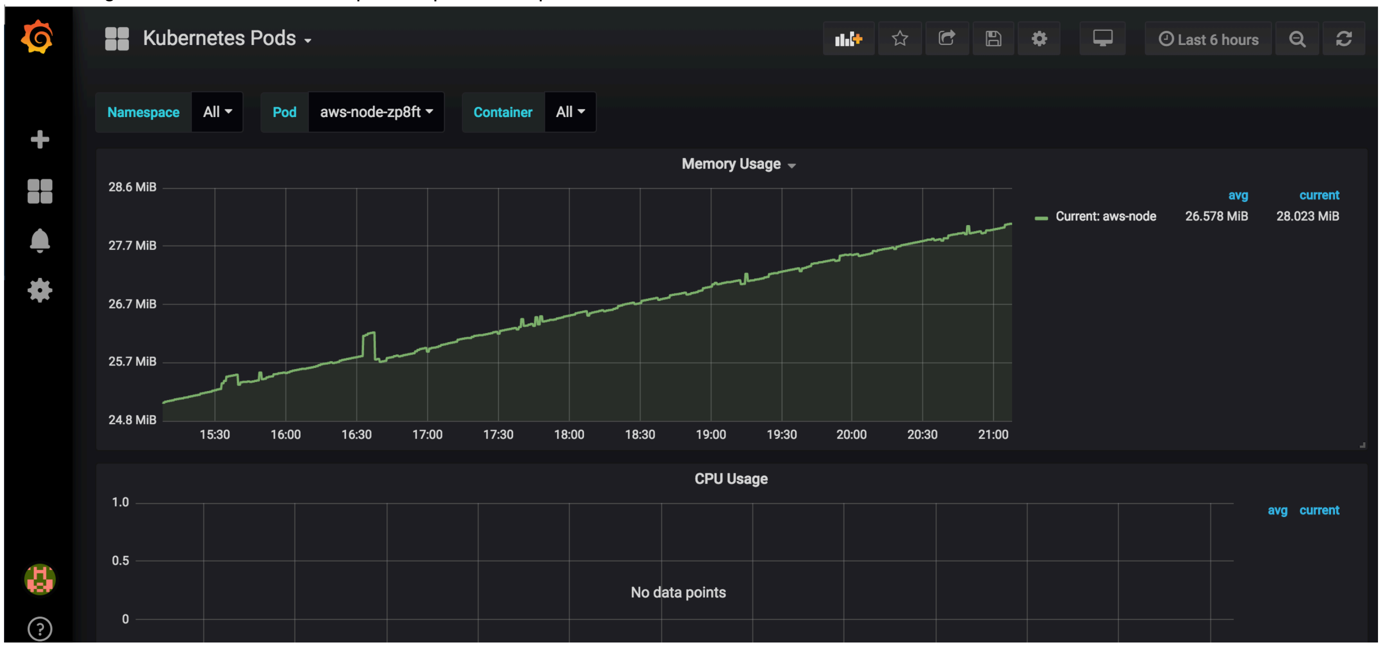
You will notice that **'Install Grafana'** & **'create your first data source'** are already completed. We will import community created dashboard:

- Click '+' button on left panel and select '**Import**'
- Enter **3131** dashboard id under Grafana.com Dashboard & click **'Load'**.
- Leave the defaults, select **'Prometheus'** as the endpoint under prometheus data sources drop down, click **'Import'**.

This will show monitoring dashboard for all cluster nodes:



For creating dashboard to monitor all pods, repeat same process as above and enter **3146** for dashboard id:



Source for these instructions - AWS EKS workshop

https://eksworkshop.com/intermediate/240_monitoring/