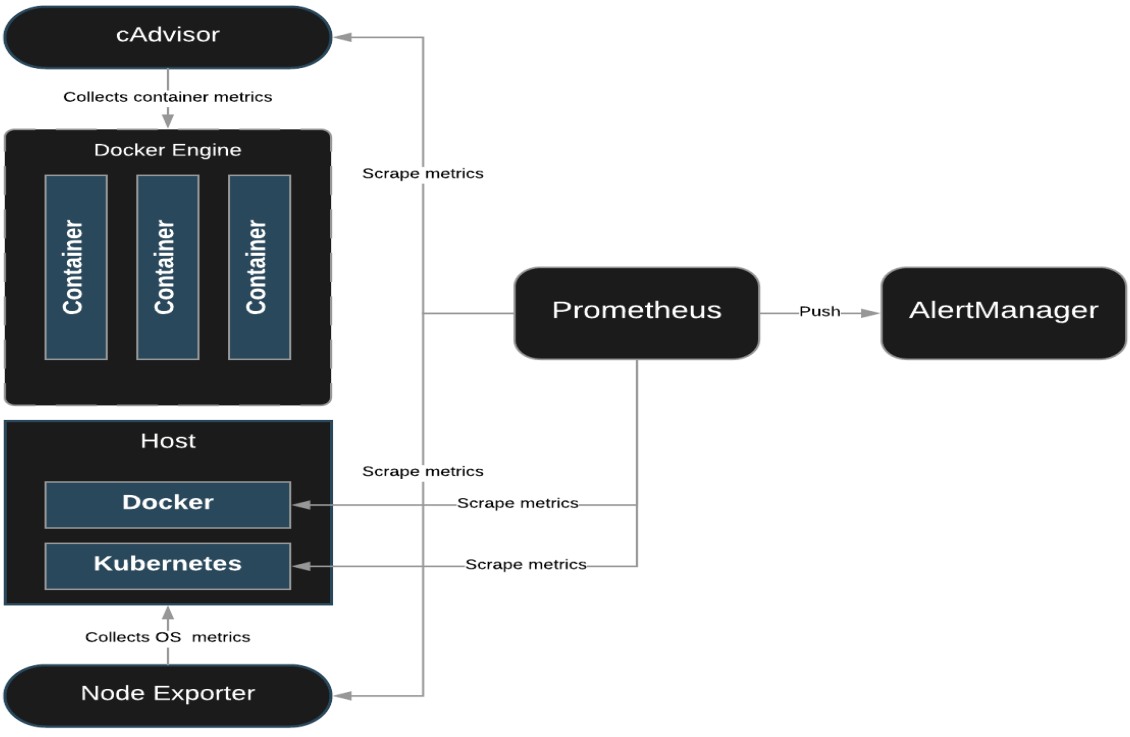
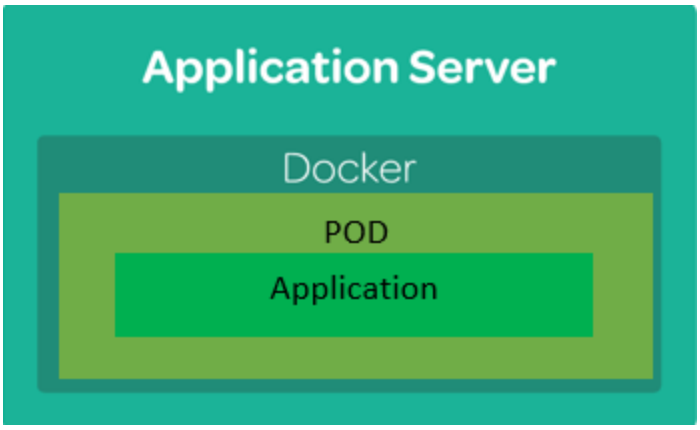
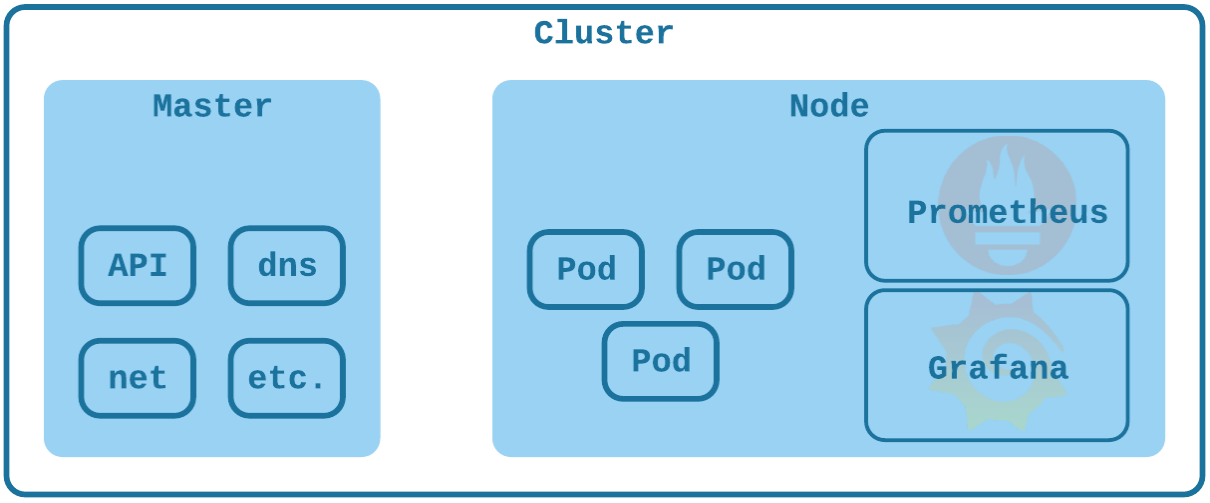
Prometheus + Grafana with K8s Short Notes

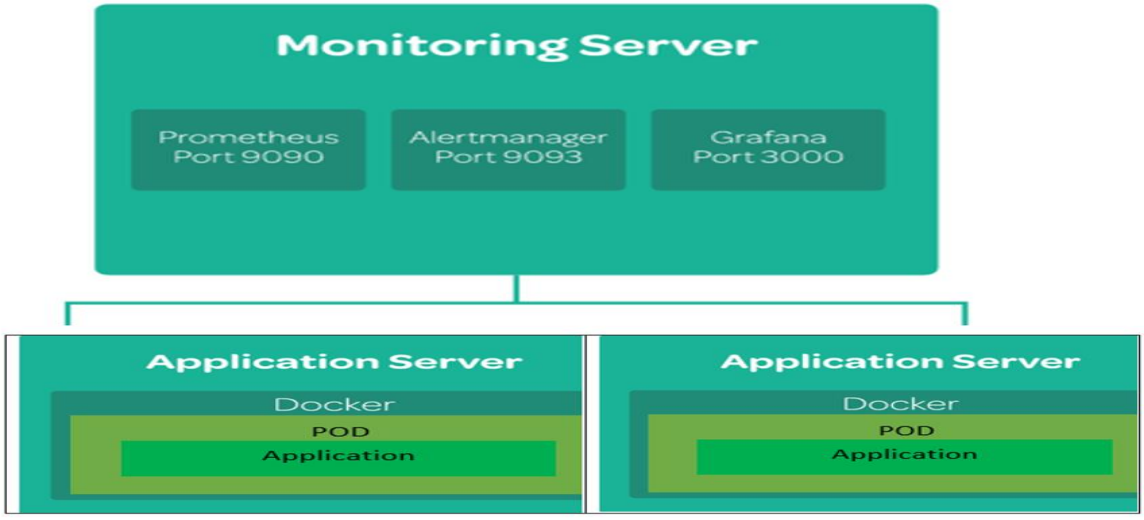
- **Prometheus** - it contains the time series database and the logic of scraping stats from exporters as well as alerts. Prometheus can efficiently manage many vital parameters such as a retention policy or a frequency of metrics collection. It stores data in its own time-series database. Responsible for collecting and storing statistics data
- **Grafana** is to build dashboard to visualize the application which shows key metrics from Prometheus in real-time
- **Alert manager** sends the Prometheus alert to various channels like email, pagers, and slack - and so on.
- **Exporters** is node-exporter, it collect system metrics like cpu/memory/storage usage and then it exports it for Prometheus to scrape.
- **Application metrics** - custom metrics you record which are valuable to see in the dashboard, like number of logged-in users or number of checked-out baskets.
- **Runtime metrics** - data already collected by the operating system or runtime host, like the requests per second handled by a web server, or the memory usage.
- **Docker metrics** - metrics from the container platform, including containers running in each state, node availability and health checks.



Monitoring in Kubernetes with Prometheus and Grafana


Application Server
Docker
POD
Application


Cluster
Master
API
dns
net
etc.
Node
Pod
Pod
Pod
Prometheus
Grafana


Monitoring Server
Prometheus Port 9090
Alertmanager Port 9093
Grafana Port 3000
Application Server
Docker
POD
Application
Application Server
Docker
POD
Application

Access Prometheus from your browser

1. Access Prometheus from your browser:

```
http://[Master Node Public IP address]:9090
```

   Status > **Targets** will show you more information on the targets in the cluster.

2. You may also access the cAdvisor dashboard at the following address:

```
http://[Worker Node Public IP Address]:8080
```

Use the provided PromQL queries to interrogate your cluster

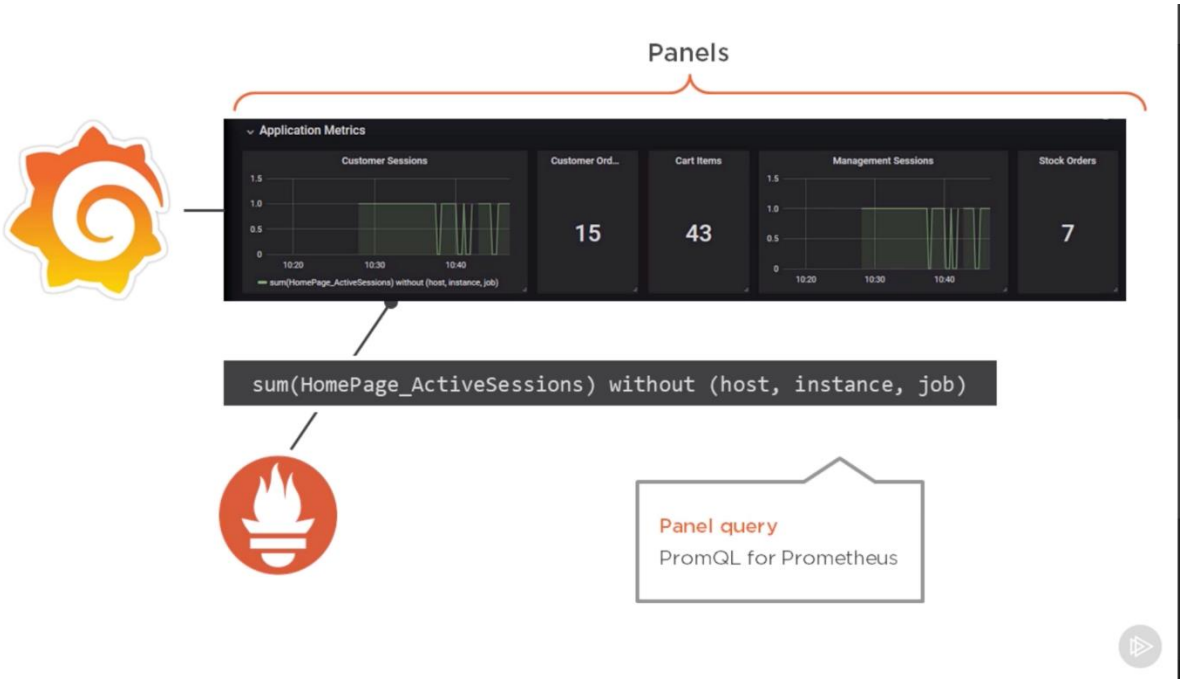
The following are the suggested PromQL queries you may perform.

1. To measure CPU utilization:

```
node_cpu_seconds_total

irate(node_cpu_seconds_total{job="node"}[5m])

avg(irate(node_cpu_seconds_total{job="node"}[5m])) by (instance)

avg(irate(node_cpu_seconds_total{job="node",mode="idle"}[5m])) by (instance) * 100

100 - avg(irate(node_cpu_seconds_total{job="node",mode="idle"}[5m])) by (instance) * 100
```

2. To measure memory, use:

```
(node_memory_MemTotal_bytes - (node_memory_MemFree_bytes + node_memory_Cached_bytes + node_memory_Buffers_bytes)) / node_memory_MemTotal_bytes * 100
```



Initialize Helm

```
helm init --wait
```

Install Prometheus in the Kubernetes Cluster

1. To do this, make sure you have cloned the Kubernetes charts repo:

```
2. cd ~/
3. git clone https://github.com/kubernetes/charts
4. cd charts
5. git checkout efdcffe0b6973111ec6e5e83136ea74cdbe6527d
```

```
cd ../
```

6. Create a *prometheus-values.yml* file:

```
vi prometheus-values.yml
```

7. And paste in this content:

```
alertmanager:
```

```
 persistentVolume:
  enabled: false
 server:

persistentVolume:

    enabled: false
```

8. Save and close the file:

```
 :wq
```

Use helm to install Prometheus with *prometheus-values.yml*:

```
 helm install -f ~/prometheus-values.yml ~/charts/stable/prometheus --name prometheus --namespace prometheus
```

We can see which pods are running in this new namespace with the command:

```
 kubectl get pods -n prometheus
```

Install Grafana in the Kubernetes Cluster

1. Create a *grafana-values.yml*:

```
 vi grafana-values.yml
```

2. And paste in this content (you will use this password to log in to Grafana):

```
 adminPassword: password
```

3. Save and close the file:

```
 :wq
```

4. Use helm to install Grafana with *grafana-values.yml*:

```
 helm install -f ~/grafana-values.yml ~/charts/stable/grafana --name grafana --namespace grafana
```

5. We can see which pods are running in this new namespace with the command:

```
 kubectl get pods -n grafana
```

Deploy a NodePort Service to Provide External Access to Grafana

1. Make a file called *grafana-ext.yml*:

```
vi grafana-ext.yml
```

2. Paste in this content:

```
kind: Service
apiVersion: v1
metadata:
  namespace: grafana
  name: grafana-ext
spec:
  type: NodePort
  selector:
    app: grafana
  ports:
  - protocol: TCP
    port: 3000
```

```
    nodePort: 8081
```

15. Save and close the file:

```
:wq
```

16. Deploy the service:

```
kubectl apply -f ~/grafana-ext.yml
```

17. Log in to Grafana using the **Kubernetes Node Public IP** provided on the hands-on lab page:

```
<KUBERNETES_NODE_PUBLIC_IP>;:8081
```

18. Log in using the following credentials that were set earlier:

    - **Username**: admin
    - **Password**: password

Create the Monitoring Dashboards

Add a Datasource for Prometheus

1. Click on **Add data source**
    - **Name**: Kubernetes
    - **Type**: Prometheus
    - **URL**: http://prometheus-server.prometheus.svc.cluster.local
2. Click **Save & Test**

Add the Kubernetes All Nodes Community Dashboard

1. Hover your mouse over the **+** in the left sidebar and click on **Import**.
2. In the *Grafana.com Dashboard* field, provide the ID 3131. Click outside of the field to load information about the dashboard.
3. In the *Options* section:
    - **prometheus**: Kubernetes
4. Click **Import**.
5. Hover your mouse over the **+** in the left sidebar and then click **Dashboard**. Select the **Graph**panel.
6. Hover over the *Panel Title* and click **Edit**.
7. In the **General** tab at the bottom of the screen, set the *Title* to **Requests Per Minute**.
8. In the **Metrics** tab, paste in the following query:

```
sum(rate(http_request_duration_ms_count[2m])) by (service, route, method, code)  * 60
```

9. With that in place, let's load our train-schedule app to give our graph some data:

```
<KUBERNETES_NODE_PUBLIC_IP>:8080
```

10. Refresh the page a few times.
11. Now, navigate back to the Grafana dashboard tab in your browser. In the top-right of the page, click **Last 6 hours** and change the time selection to **Last 5 minutes**.
12. Click **Back to dashboard** in the top-right of the page. Next, click **Save dashboard**, also in the top-right of the page.
13. Name the dashboard "Train Schedule Performance" and click **Save**.