Amazon EKS - Observability with Prometheus and Grafana

Description

When it comes to running and operating production-grade EKS managed Kubernetes clusters, monitoring and alerting are considered essential components of an enterprise Kubernetes observability stack.

In this hands-on lab, you'll learn how to set up and use the following essential monitoring applications:

- Prometheus
- Grafana

You'll learn how to integrate these monitoring applications together into an effective and cohesive monitoring solution.

Learning Objectives

Upon completion of this Lab, you will be able to:

- Deploy and instrument a sample Python Flask web-based API into Kubernetes, instrumented to provide metrics which will be collected by Prometheus and displayed within Grafana
- Install and configure Prometheus into Kubernetes using Helm
- Setup Prometheus for service discovery
- Install and configure Grafana into Kubernetes using Helm
- Import pre-built Grafana dashboards for real-time visualisations

Lab Environment

This Lab will start with the following AWS resources provisioned automatically for you:

- 1 x EKS cluster - **Cluster-1** - provides a fully functional Kubernetes cluster
  - 1 x NodeGroup
    - 2 x EC2 Worker Nodes
- 2 x EC2 instances
  - **eks.launch.instance** - used to launch the EKS cluster
  - **cloudacademylabs** - used to provide an SSH based terminal

Lab steps

- Logging In to the Amazon Web Services Console

- Connecting to the Virtual Machine using EC2 Instance Connect
- Reviewing Amazon EKS Resources Automatically Created
- Installing Kubernetes Management Tools and Utilities
- Observability using Prometheus and Grafana

Connecting to the Virtual Machine using EC2 Instance Connect

1. The instances list page will open, and you will see an instance named **cloudacademylabs**
2. Right-click the **cloudacademylabs** instance, and click **Connect**
3. In the **Username** textbox, if it is empty, enter *ec2-user*

Reviewing Amazon EKS Resources Automatically Created

Introduction

This lab has automatically created a number of AWS resources for you - including an EKS cluster. Some resources may still be being provisioned in the background. The EKS cluster named **Cluster-1** typically takes 15-25 minutes to reach **Active** status, and only then will the EC2 Worker Node instance named similar to **Cluster-1-___-Node** be provisioned and automatically joined to the cluster. In this lab step, you will review the lab starting state and ensure that all Amazon EKS-related resources have been created and are in the required operational status.

Note: **Before** moving on to the next lab step - review and confirm that all AWS resources itemized below have achieved their expected operational status.

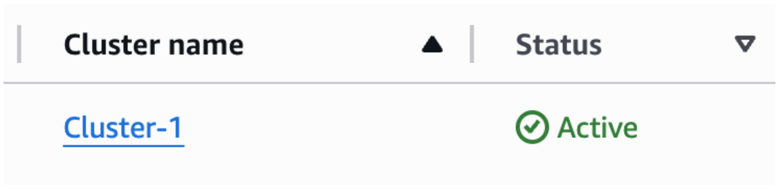
FYI: The EKS cluster named **Cluster-1** and the EC2 Worker Node instance named similar to **Cluster-1-___-Node** are automatically provisioned and joined using the EKSCTL, opens in a new tab tool. This happens in the background and is performed on the EC2 instance named **eks.launch.instance** (access to this instance is not required to complete this lab).

Instructions

1. In the AWS Management Console search bar, enter *EKS*, and click the **EKS** result under **Services**

Click on the **Clusters** on the left and confirm that the EKS cluster (managed control plane) named **Cluster-1** has been successfully created and is in an **Active** state.

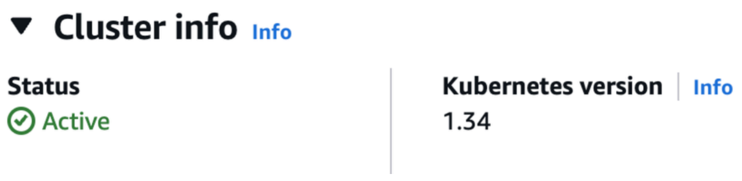
Note: Please be patient, the EKS cluster takes approximately 15-25 minutes to be provisioned and reach the **Active** state:



Cluster name ▲	Status ▽
Cluster-1	✓ Active

2. Within the AWS EKS console click on **Cluster-1** and review the **Cluster info**

Note: The Kubernetes **version** is set during lab launch time, and maybe a version newer than that displayed below (>=1.34).



▼ Cluster info Info

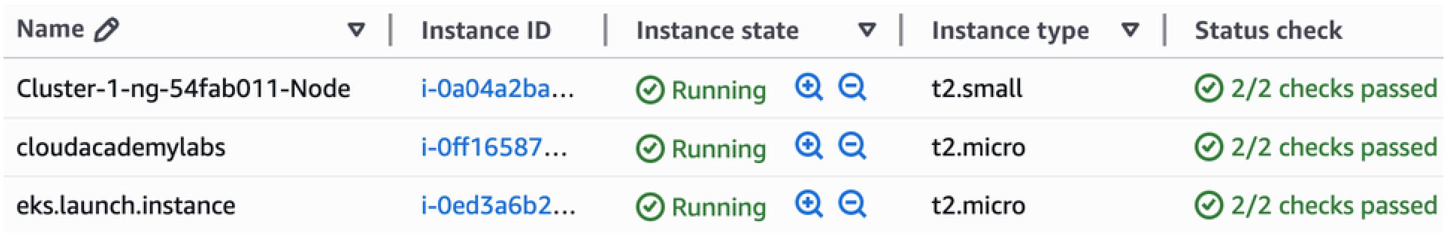
Status	Kubernetes version Info
✓ Active	1.34

3. Within the AWS web console navigate to the **AWS EC2 console**

4. Click **Instances** > **Instances** on the left-hand side menu

Confirm that the following 3 x EC2 instances are in **running** status, can take up to 15-25 minutes:

- **Cluster-1-___-Node** (EKS cluster Worker Node) - running within 15-25 mins only after the EKS cluster has acquired Active status, SSH access to this instance is not required, but is granted for this lab exercise
- **cloudacademylabs** (used to manage the EKS cluster using the kubectl command) - running within 1-2 mins, SSH access to this instance is required and granted for this lab exercise
- **eks.launch.instance** (used to launch EKS cluster and Worker Node) - running within 1-2 mins, SSH access to this instance is not required nor granted for this lab exercise



Name	Instance ID	Instance state	Instance type	Status check
Cluster-1-ng-54fab011-Node	i-0a04a2ba...	✓ Running	t2.small	✓ 2/2 checks passed
cloudacademylabs	i-0ff16587...	✓ Running	t2.micro	✓ 2/2 checks passed
eks.launch.instance	i-0ed3a6b2...	✓ Running	t2.micro	✓ 2/2 checks passed

Note: it typically takes 15-20 minutes for the **Cluster-1-___-Node** (EKS cluster Worker Node) instance to appear - as it only gets provisioned after the EKS cluster has reached the **Active** state.

Summary

In this lab step, you reviewed and confirmed that all Amazon EKS-related resources have achieved their expected operational status.

Installing Kubernetes Management Tools and Utilities

Introduction

In preparation to manage your EKS Kubernetes cluster, you will need to install several Kubernetes management-related tools and utilities. In this lab step, you will install:

- **kubectl:** the Kubernetes command-line utility which is used for communicating with the Kubernetes Cluster API server
- **awscli**: used to query and retrieve your Amazon EKS cluster connection details, written into the ~/.kube/config file

Instructions

1. Download the kubectl utility, give it executable permissions, and copy it into a directory that is part of the PATH environment variable:

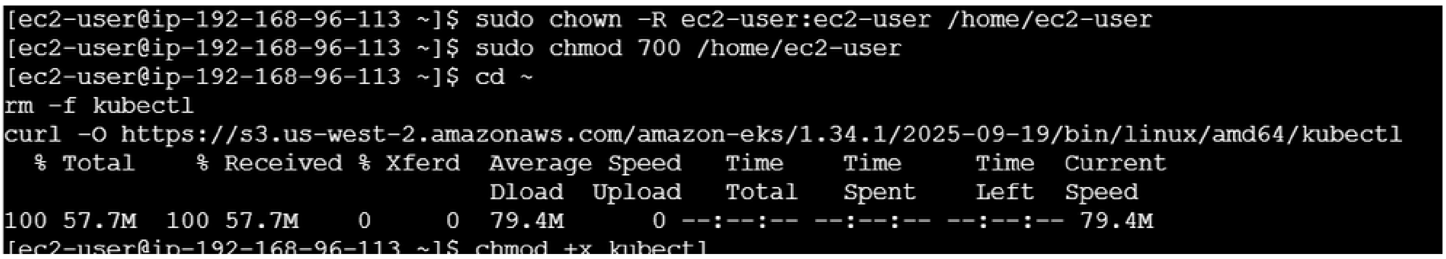
```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.1/2025-09-19/bin/linux/amd64/kubectl
```

```
chmod +x ./kubectl
```

```
sudo cp ./kubectl /usr/local/bin
```

```
export PATH=/usr/local/bin:$PATH
```

2. Test the kubectl utility, ensuring that it can be called like so:



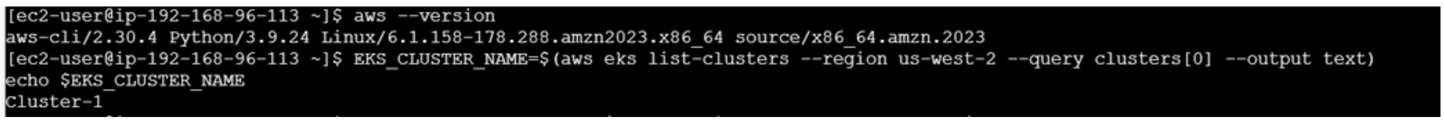
```
[ec2-user@ip-192-168-96-113 ~]$ sudo chown -R ec2-user:ec2-user /home/ec2-user
[ec2-user@ip-192-168-96-113 ~]$ sudo chmod 700 /home/ec2-user
[ec2-user@ip-192-168-96-113 ~]$ cd ~
rm -f kubectl
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.1/2025-09-19/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 57.7M  100 57.7M    0     0  79.4M      0 --:--:-- --:--:-- --:--:-- 79.4M
[ec2-user@ip-192-168-96-113 ~]$ chmod +x kubectl
```

```
kubectl version --client=true
```

You will see the version of the kubectl utility displayed.

3. Verify that the AWS CLI version 2 is installed:

```
aws --version
```



```
[ec2-user@ip-192-168-96-113 ~]$ aws --version
aws-cli/2.30.4 Python/3.9.24 Linux/6.1.158-178.288.amzn2023.x86_64 source/x86_64.amzn.2023
[ec2-user@ip-192-168-96-113 ~]$ EKS_CLUSTER_NAME=$(aws eks list-clusters --region us-west-2 --query clusters[0] --output text)
echo $EKS_CLUSTER_NAME
Cluster-1
```

In most recent Amazon Linux distributions, the AWS CLI v2 comes pre-installed. If you are using another operating system (for example, Ubuntu), you may need to install it manually by following the instructions provided in the official AWS documentation.

4. Use the aws utility, to retrieve EKS Cluster name:

Copy code

```
EKS_CLUSTER_NAME=$(aws eks list-clusters --region us-west-2 --query clusters[0] --output text)
```

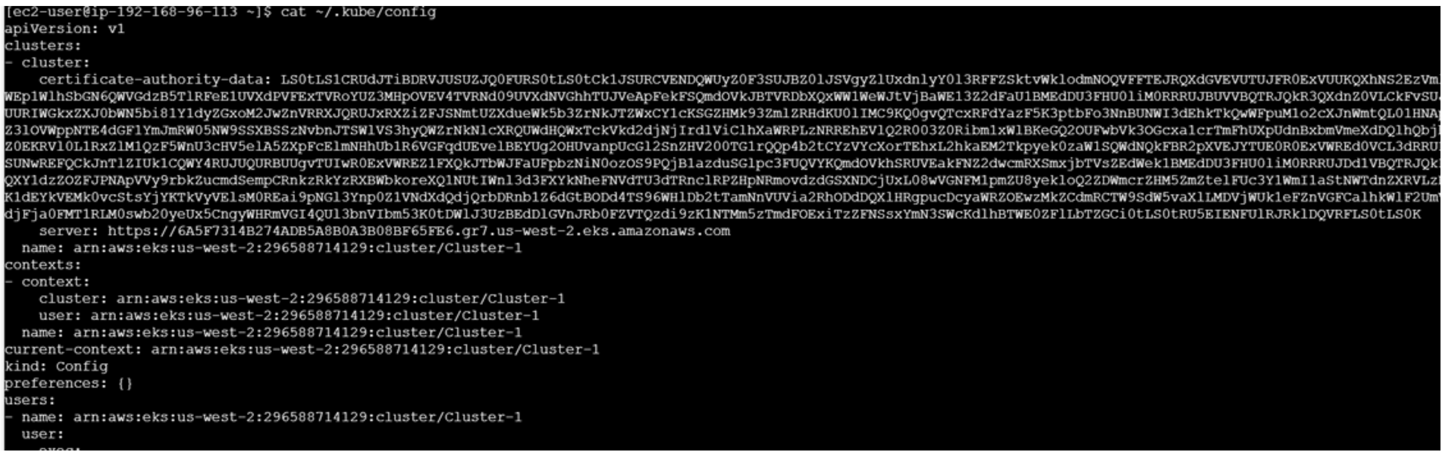
```
echo $EKS_CLUSTER_NAME
```

5. Use the aws utility to query and retrieve your Amazon EKS cluster connection details, saving them into the **~/.kube/config** file. Enter the following command in the terminal:

```
aws eks update-kubeconfig --name $EKS_CLUSTER_NAME --region us-west-2
```

6. View the EKS Cluster connection details. This confirms that the EKS authentication credentials required to authenticate have been correctly copied into the **~/.kube/config** file. Enter the following command in the terminal:

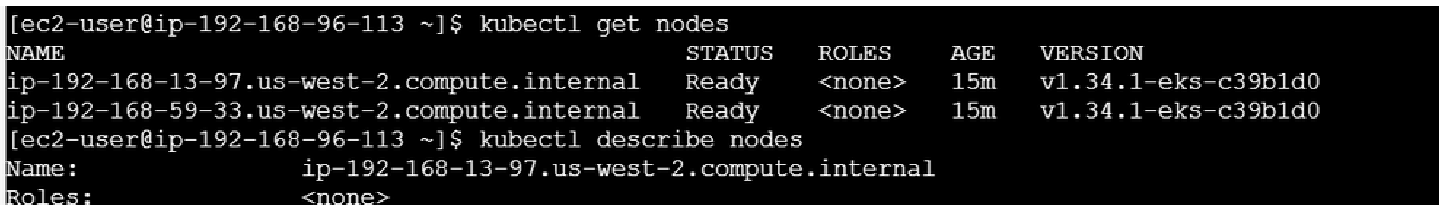
```
cat ~/.kube/config
```



```
[ec2-user@ip-192-168-96-113 ~]$ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJSVgyZlUxdnlyY0l3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVm
WEp1WlhSbGN6QWVGdzB5TlRFeE1UVXdPVFExTVRoYUZ3MHpOVEV4TVRNd09UVXdNVGhhTUJVeApFekFSQmdOVkJBTVRDbXQxWW1WeWJtVjBaWE13Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTRJQkR3QXdnZ0VLCkFvSU
UURIWGkxZXJ0bWN5bi81Y1dyZGxoM2JwZnVRRXJQRUJxRXZiZFJSNmtUZXdueWk5b3ZrNkJTZWxCY1cKSGZHMk93ZmlZRHdKU01IMC9KQ0gvQTcxRFdYazF5K3ptbFo3NnBUNWI3dEhkTkQwWFpuM1o2cXJnWmtQL01HNAp
Z3lOVWppNTE4dGF1YmJmRW05NW9SSXBSSzNvbnJTSW1VS3hyQWZrNkNlcXRQUWdHQWxTckVkd2djNjIrdlViClhXaWRPLzNRREhEV1Q2R003Z0Ribm1xW1BKeGQ2OUFwbVk3OGcxa1crTmFhUXpUdnBxbmVmeXdDQ1hQbj
Z0EKRVl0L1RxZlM1QzF5WnU3cHV5elA5ZXpFcElmNHhUb1R6VGFqdUEvelBEYUg2OHUvanpUcGl2SnZHV200TG1rQQp4b2tCYzVYcXorTEhxL2hkaEM2Tkpyek0zaW1SQWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRU
SUNwREFQCkJnTlZIUk1CQWY4RUJUQURBUUgvTUIwR0ExVWREZ1FXQkJTbWJFaUFpbzNiN0ozOS9PQjBlazduSGlpc3FUQVYKQmdOVkhSRUVEakFNZ2dwcmRXSmxjbTVsZEdWek1BMEdDU3FHU0liM0RRRUJDd1VBQTRJQk
QXYldzZ0ZFJPNApVVy9rbkZucmdSempCRnkzRkYzRXBWbkoreXQ1NUtIWnl3d3FXYkNheFNVdTU3dTRnclRPZHpNRmovdzdGSXNDCjUxL08wVGNFM1pmZU8yeekloQ2ZDWmcrZHM5ZmZtelFUc3Y1WmIlaStNWTdnZXRVLz
K1dEYkVEMk0vcStsYjYKTkVyVElsM0REai9pNGl3Ynp0Z1VNdXdQdjQrbDRnb1Z6dGtBODd4TS96WHlDb2tTamNnVUVia2RhODdDQXlHRgpucDcyaWRZOEwzMkZCdmRCTW9SdW5vaXlLMDVjWUk1eFZnVGFCalhkWlF2Um
djFja0FMT1RLM0swb20yeUx5CngyWHRmVGI4QUl3bnVIbm53K0tDWlJ3UzBEdDlGVnJRb0ZZVTQzdi9zK1NTMm5zTmdFOExiTzZFNSsxYmN3SWcKdlhBTWE0ZFlLbTZGCi0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K
    server: https://6A5F7314B274ADB5A8B0A3B08BF65FE6.gr7.us-west-2.eks.amazonaws.com
  name: arn:aws:eks:us-west-2:296588714129:cluster/Cluster-1
contexts:
- context:
    cluster: arn:aws:eks:us-west-2:296588714129:cluster/Cluster-1
    user: arn:aws:eks:us-west-2:296588714129:cluster/Cluster-1
  name: arn:aws:eks:us-west-2:296588714129:cluster/Cluster-1
current-context: arn:aws:eks:us-west-2:296588714129:cluster/Cluster-1
kind: Config
preferences: {}
users:
- name: arn:aws:eks:us-west-2:296588714129:cluster/Cluster-1
  user:
```

7. Use the kubectl utility to list the EKS Cluster Worker Nodes:

```
kubectl get nodes
```



```
[ec2-user@ip-192-168-96-113 ~]$ kubectl get nodes
NAME                                        STATUS   ROLES    AGE   VERSION
ip-192-168-13-97.us-west-2.compute.internal   Ready    <none>   15m   v1.34.1-eks-c39b1d0
ip-192-168-59-33.us-west-2.compute.internal   Ready    <none>   15m   v1.34.1-eks-c39b1d0
[ec2-user@ip-192-168-96-113 ~]$ kubectl describe nodes
Name:               ip-192-168-13-97.us-west-2.compute.internal
Roles:              <none>
```

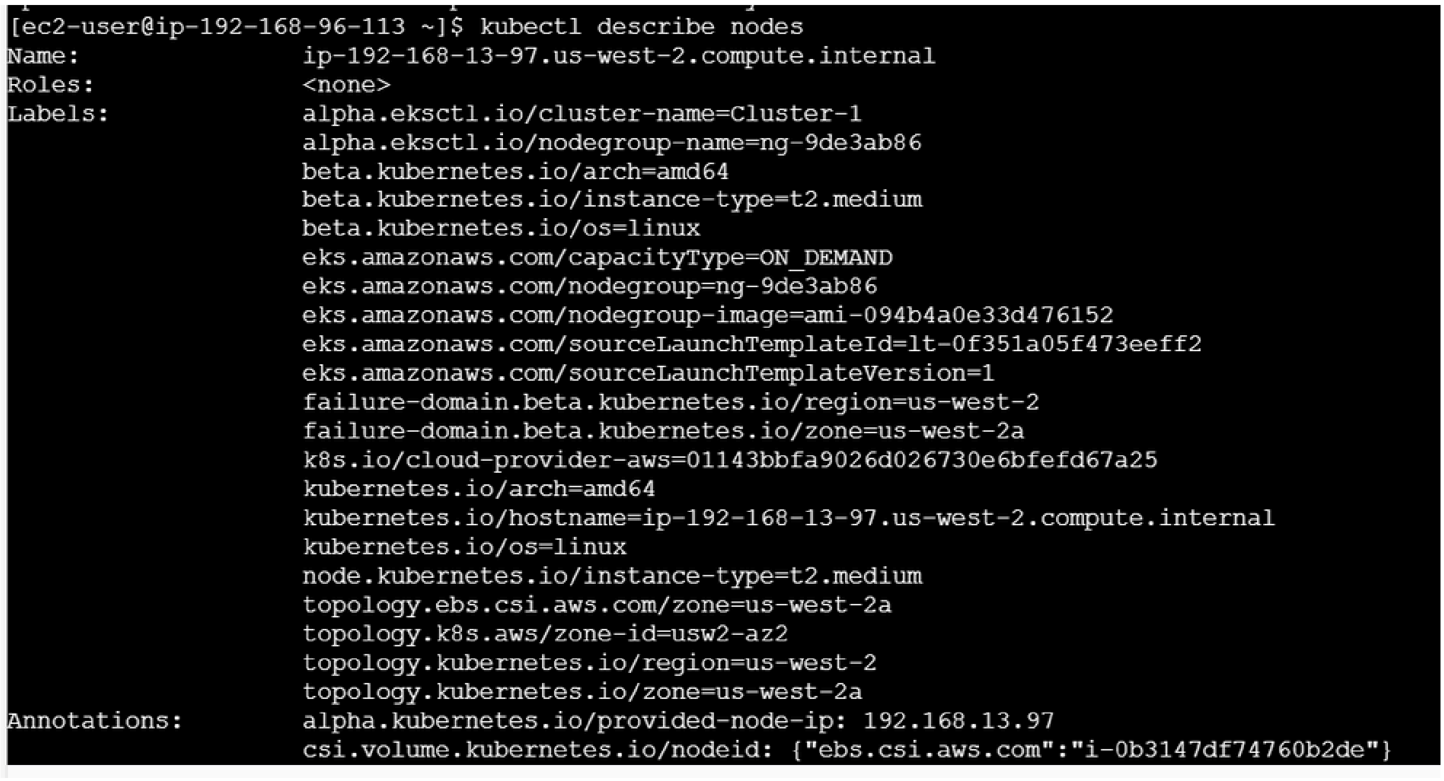
Note: If you see the following error, authorization to the cluster is still propagating to your instance role

Ensure the cluster's node group is ready and reissue the command periodically.

8. Use the kubectl utility to describe in more detail the EKS Cluster Worker Nodes:

Copy code

`kubectl describe nodes`



```
[ec2-user@ip-192-168-96-113 ~]$ kubectl describe nodes
Name:               ip-192-168-13-97.us-west-2.compute.internal
Roles:              <none>
Labels:             alpha.eksctl.io/cluster-name=Cluster-1
                    alpha.eksctl.io/nodegroup-name=ng-9de3ab86
                    beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/instance-type=t2.medium
                    beta.kubernetes.io/os=linux
                    eks.amazonaws.com/capacityType=ON_DEMAND
                    eks.amazonaws.com/nodegroup=ng-9de3ab86
                    eks.amazonaws.com/nodegroup-image=ami-094b4a0e33d476152
                    eks.amazonaws.com/sourceLaunchTemplateId=lt-0f351a05f473eeff2
                    eks.amazonaws.com/sourceLaunchTemplateVersion=1
                    failure-domain.beta.kubernetes.io/region=us-west-2
                    failure-domain.beta.kubernetes.io/zone=us-west-2a
                    k8s.io/cloud-provider-aws=01143bbfa9026d026730e6bfefd67a25
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=ip-192-168-13-97.us-west-2.compute.internal
                    kubernetes.io/os=linux
                    node.kubernetes.io/instance-type=t2.medium
                    topology.ebs.csi.aws.com/zone=us-west-2a
                    topology.k8s.aws/zone-id=usw2-az2
                    topology.kubernetes.io/region=us-west-2
                    topology.kubernetes.io/zone=us-west-2a
Annotations:        alpha.kubernetes.io/provided-node-ip: 192.168.13.97
                    csi.volume.kubernetes.io/nodeid: {"ebs.csi.aws.com":"i-0b3147df74760b2de"}
```

Summary

In this lab step, you installed and configured several Kubernetes management-related tools and utilities. You then performed a connection and authentication against your EKS Cluster and were able to determine the status of the two Worker Nodes provisioned within the cluster.

Observability using Prometheus and Grafana

Introduction

In this lab step, you'll deploy a sample API and API client to generate data points and metrics that will be collected by Prometheus and displayed within Grafana. Using Helm, you'll be required to deploy both Prometheus and Grafana. Prometheus is an open-source systems monitoring and alerting service. Grafana is an open source analytics and interactive visualization web application, providing charts, graphs, and alerts for monitoring and observability requirements. You'll configure Grafana to connect to Prometheus as a data source. Once connected, you'll import and deploy 2 prebuilt Grafana dashboards. You'll configure Prometheus to perform automatic service discovery of both the sample API, and the EKS cluster itself. Prometheus once configured, will then automatically start collecting various metrics from both the sample API and the EKS cluster.

Notes:

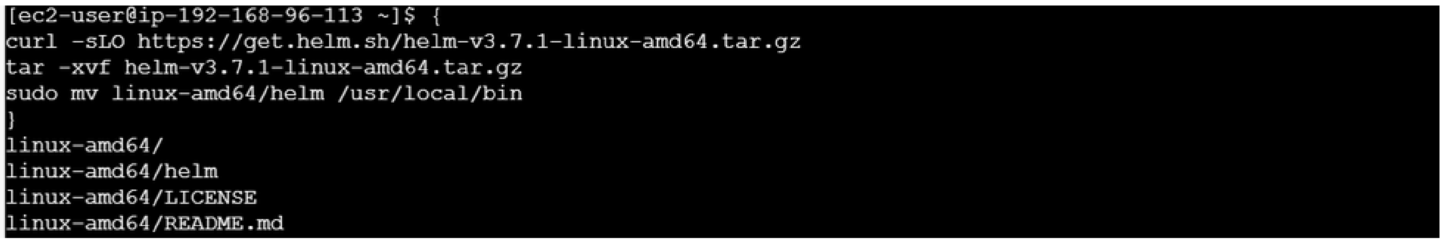
- The Prometheus web admin interface will be exposed over the Internet on port **80**, allowing you to access it from your own workstation.
- The Grafana web admin interface will be exposed over the Internet on port **80**, allowing you to access it from your own workstation.

Instructions

1. Install tools and download the application code.

Copy code

```
{
curl -sLO https://get.helm.sh/helm-v3.7.1-linux-amd64.tar.gz
tar -xvf helm-v3.7.1-linux-amd64.tar.gz
sudo mv linux-amd64/helm /usr/local/bin
}
```



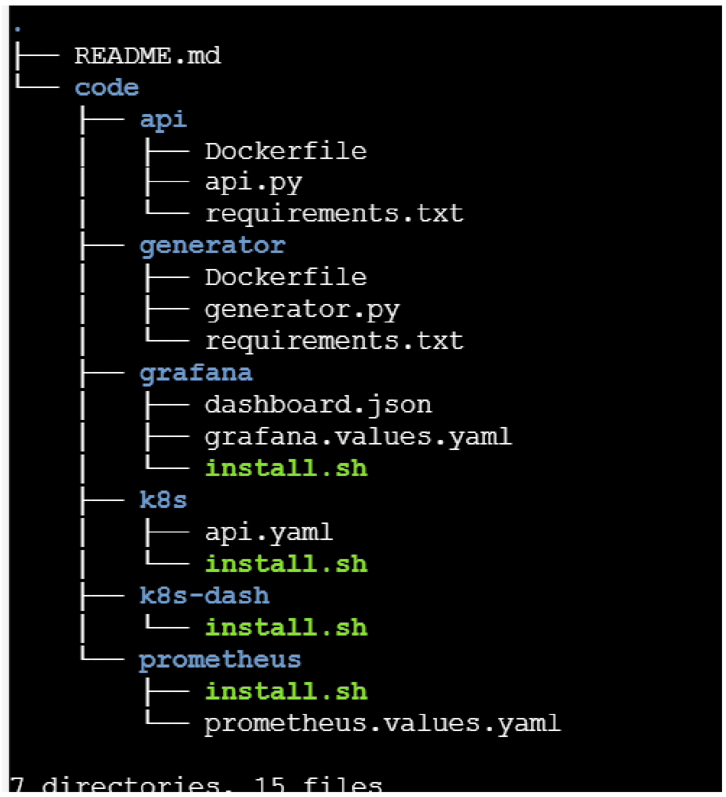
```
[ec2-user@ip-192-168-96-113 ~]$ {
curl -sLO https://get.helm.sh/helm-v3.7.1-linux-amd64.tar.gz
tar -xvf helm-v3.7.1-linux-amd64.tar.gz
sudo mv linux-amd64/helm /usr/local/bin
}
linux-amd64/
linux-amd64/helm
linux-amd64/LICENSE
linux-amd64/README.md
```

2. Download the observability application code and scripts. In the terminal run the following commands:

Copy code

```
{
mkdir -p ~/cloudacademy/observability
cd ~/cloudacademy/observability
curl -sL https://api.github.com/repos/cloudacademy/k8s-lab-observability1/releases/latest | \
 jq -r '.zipball_url' | \
 wget -qi -
unzip release* && find -type f -name *.sh -exec chmod +x {} \;
cd cloudacademy-k8s-lab-observability*
tree
}
```

A tree view of the downloaded project directory structure is provided:



```
.
├── README.md
└── code
    ├── api
    │   ├── Dockerfile
    │   ├── api.py
    │   └── requirements.txt
    ├── generator
    │   ├── Dockerfile
    │   ├── generator.py
    │   └── requirements.txt
    ├── grafana
    │   ├── dashboard.json
    │   ├── grafana.values.yaml
    │   └── install.sh
    ├── k8s
    │   ├── api.yaml
    │   └── install.sh
    ├── k8s-dash
    │   └── install.sh
    └── prometheus
        ├── install.sh
        └── prometheus.values.yaml

7 directories, 15 files
```

3. Install the sample **API** and **API client** cluster resources.

Note: All of the source code as used within this lab is located online here:

https://github.com/cloudacademy/k8s-lab-observability1

Note: Before installing, examine the source code that makes up the Python based API and API client.

The ./code/api directory contains 3 files which have been used to build the **cloudacademydevops/api-metrics** container image.

Open each of the following files within your editor of choice.

- **api.py**
- **Dockerfile**
- **requirements.txt**

The **api.py** file contains the Python source code which implements the example API.

In particular take note of the following:

- **Line 5** - imports a **PromethusMetrics** module to automatically generate Flask based metrics and provide them for collection at the default endpoint **/metrics**
- **Line 10-32**- implements 5 x API endpoints:
  - /one
  - /two
  - /three
  - /four
  - /error
- All example endpoints, except for the error endpoint, introduce a small amount of latency which will be measured and observed within both Prometheus and Grafana.
- The error endpoint returns an HTTP 500 server error response code, which again will be measured and observed within both Prometheus and Grafana.
- The Docker container image containing this source code has already been built using the tag **cloudacademydevops/api-metrics**

./code/k8s directory contains the **api.yaml** Kubernetes manifest file used to deploy the API into the cluster.

Open each of the following files within your editor of choice.

- **api.yaml**

In particular note the following:

- **Lines 1-25**: API Deployment containing 2 pods
- **Line 22**: Pods are based off the container image **cloudacademydevops/api-metrics**
- **Lines 27-46**: API Service - load balances traffic across the 2 API Deployment pods
- **Lines 34-37**: API Service is annotated to ensure that the Prometheus scraper will automatically discover the API pods behind it. Prometheus will then collect their metrics from the discovered targets

3. The API and API client need to be deployed into the cloudacademy namespace which needs to be created. Create the new namespace. In the terminal execute the following command:

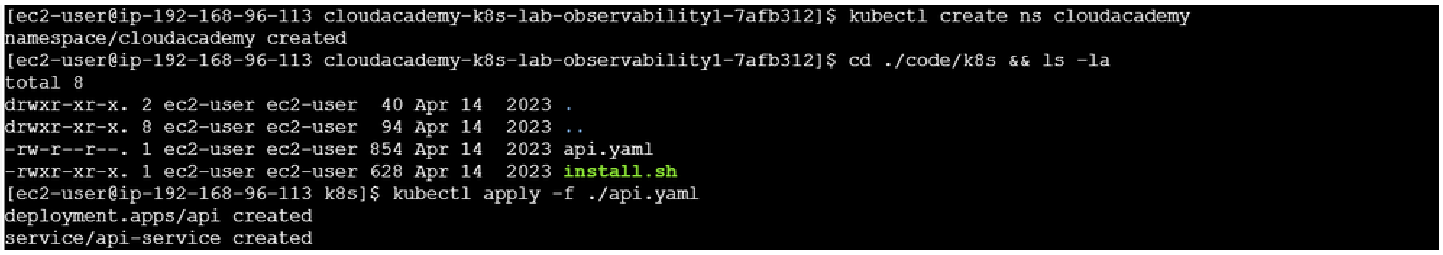
Copy code

```
kubectl create ns cloudacademy
```

4. Navigate into the **k8s** directory and perform a directory listing. In the terminal execute the following commands:

Copy code

```
cd ./code/k8s && ls -la
```



```
[ec2-user@ip-192-168-96-113 cloudacademy-k8s-lab-observability1-7afb312]$ kubectl create ns cloudacademy
namespace/cloudacademy created
[ec2-user@ip-192-168-96-113 cloudacademy-k8s-lab-observability1-7afb312]$ cd ./code/k8s && ls -la
total 8
drwxr-xr-x. 2 ec2-user ec2-user  40 Apr 14  2023 .
drwxr-xr-x. 8 ec2-user ec2-user  94 Apr 14  2023 ..
-rw-r--r--. 1 ec2-user ec2-user 854 Apr 14  2023 api.yaml
-rwxr-xr-x. 1 ec2-user ec2-user 628 Apr 14  2023 install.sh
[ec2-user@ip-192-168-96-113 k8s]$ kubectl apply -f ./api.yaml
deployment.apps/api created
service/api-service created
```

5. Deploy the sample **API** into the newly created namespace. In the terminal execute the following command

Copy code

```
kubectl apply -f ./api.yaml
```

6. Deploy the sample **API client** into the newly created namespace. In the terminal execute the following command

Copy code

```
kubectl run api-client \

 --namespace=cloudacademy \

 --image=cloudacademydevops/api-generator \

 --env="API_URL=http://api-service:5000" \

 --image-pull-policy IfNotPresent
```

7. Wait for the sample API and API client resources to start up. In the terminal run the following commands:

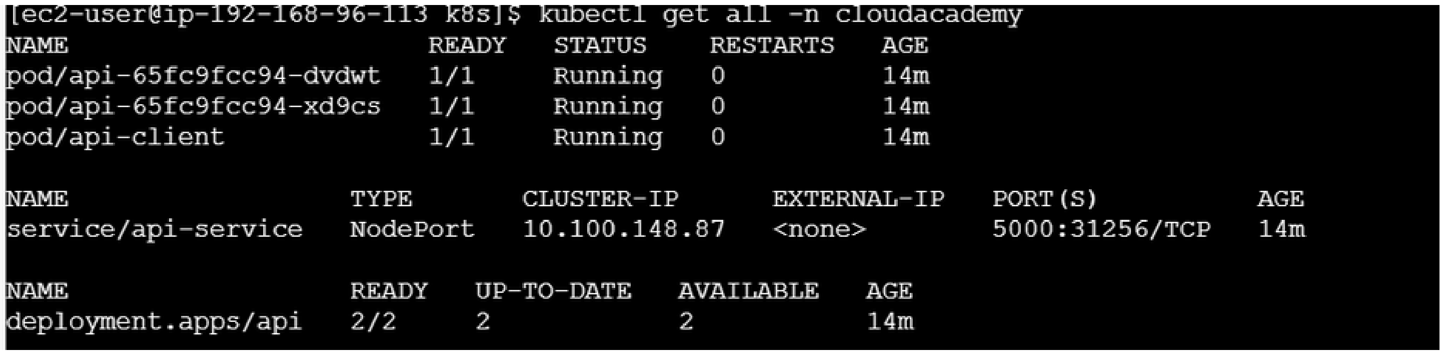
Copy code

```
{

kubectl wait --for=condition=available --timeout=300s deployment/api -n cloudacademy

kubectl wait --for=condition=ready --timeout=300s pod/api-client -n cloudacademy

}
```

8. Confirm that the sample API and API client cluster resources have all started successfully. In the terminal run the following commands:

Copy code

```
kubectl get all -n cloudacademy
```



```
[ec2-user@ip-192-168-96-113 k8s]$ kubectl get all -n cloudacademy
NAME                          READY   STATUS    RESTARTS   AGE
pod/api-65fc9fcc94-dvdwt      1/1     Running   0          14m
pod/api-65fc9fcc94-xd9cs      1/1     Running   0          14m
pod/api-client                1/1     Running   0          14m

NAME                  TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/api-service   NodePort   10.100.148.87   <none>        5000:31256/TCP   14m

NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/api   2/2     2            2           14m
```

9. Install **Prometheus** into the cluster.

Prometheus will be deployed into the prometheus namespace which needs to be created. Create a new namespace named prometheus. In the terminal run the follwing command:

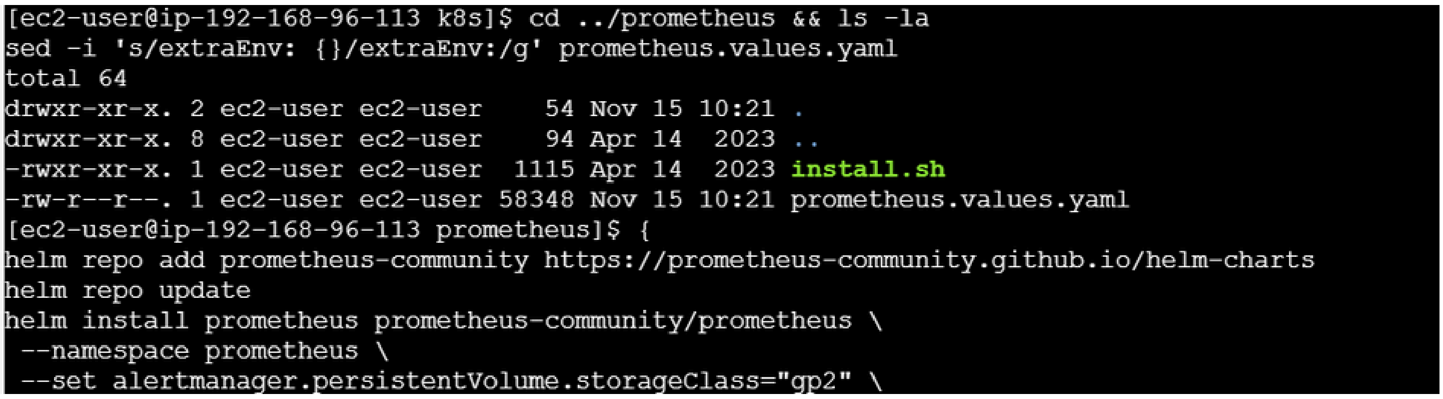
Copy code

```
kubectl create ns prometheus
```

10. Navigate into the **prometheus** directory and perform a directory listing. In the terminal execute the following commands:

Copy code

```
cd ../prometheus && ls -la

sed -i 's/extraEnv: {}/extraEnv:/g' prometheus.values.yaml
```



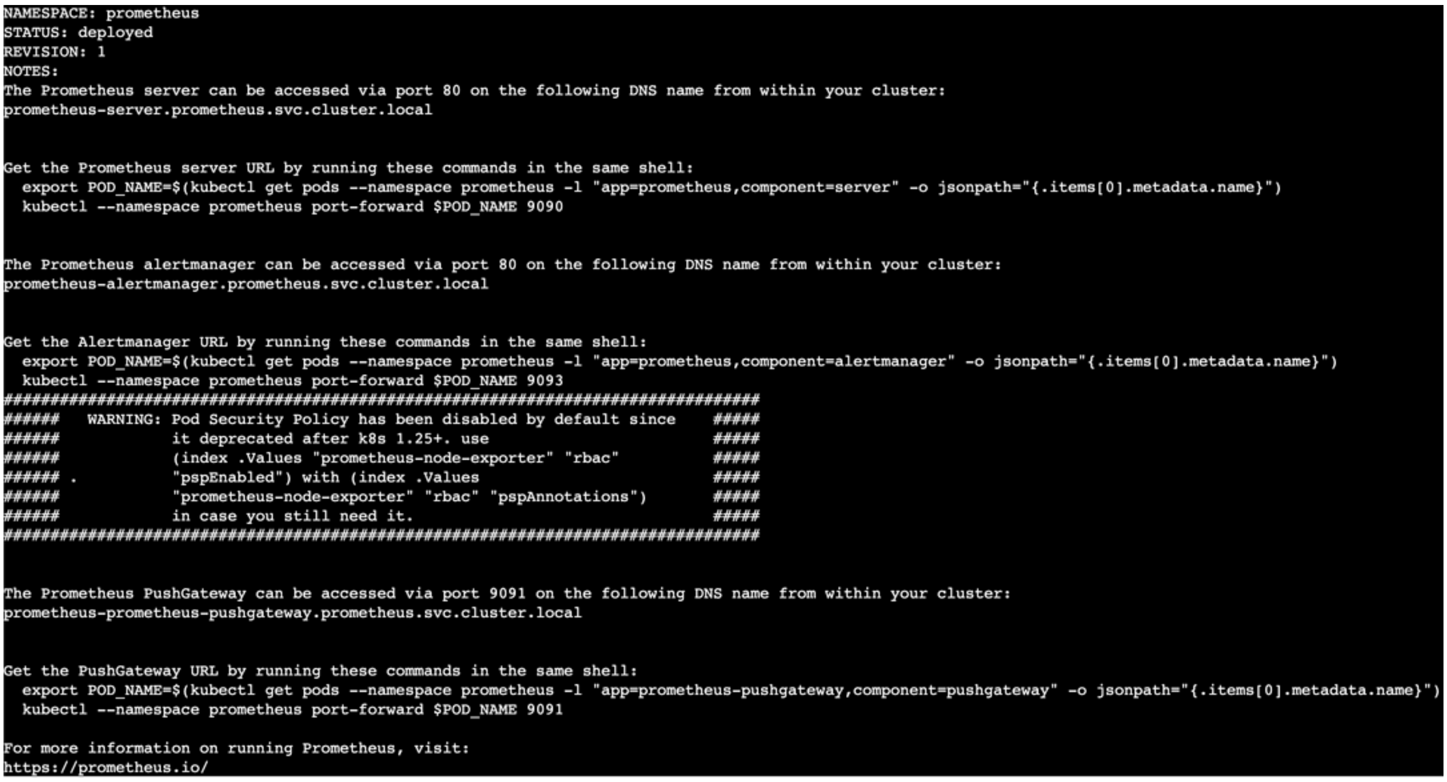
```
[ec2-user@ip-192-168-96-113 k8s]$ cd ../prometheus && ls -la
sed -i 's/extraEnv: {}/extraEnv:/g' prometheus.values.yaml
total 64
drwxr-xr-x. 2 ec2-user ec2-user    54 Nov 15 10:21 .
drwxr-xr-x. 8 ec2-user ec2-user    94 Apr 14  2023 ..
-rwxr-xr-x. 1 ec2-user ec2-user  1115 Apr 14  2023 install.sh
-rw-r--r--. 1 ec2-user ec2-user 58348 Nov 15 10:21 prometheus.values.yaml
[ec2-user@ip-192-168-96-113 prometheus]$ {
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
helm install prometheus prometheus-community/prometheus \
 --namespace prometheus \
 --set alertmanager.persistentVolume.storageClass="gp2" \
```

11. With Helm, install Prometheus using it's publicly available Helm Chart. Deploy Prometheus into the prometheus namespace. In the terminal execute the following commands:

Copy code

```
{
```

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

helm repo update

helm install prometheus prometheus-community/prometheus \

 --namespace prometheus \

 --set alertmanager.persistentVolume.storageClass="gp2" \

 --set server.persistentVolume.storageClass="gp2" \

 --values ./prometheus.values.yaml

}
```



```
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.prometheus.svc.cluster.local


Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace prometheus -l "app=prometheus,component=server" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace prometheus port-forward $POD_NAME 9090


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


Get the Alertmanager URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace prometheus -l "app=prometheus,component=alertmanager" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace prometheus port-forward $POD_NAME 9093
#################################################################################
######   WARNING: Pod Security Policy has been disabled by default since    #####
######            it deprecated after k8s 1.25+. use                        #####
######            (index .Values "prometheus-node-exporter" "rbac"          #####
###### .          "pspEnabled") with (index .Values                         #####
######            "prometheus-node-exporter" "rbac" "pspAnnotations")       #####
######            in case you still need it.                                #####
#################################################################################


The Prometheus PushGateway can be accessed via port 9091 on the following DNS name from within your cluster:
prometheus-prometheus-pushgateway.prometheus.svc.cluster.local


Get the PushGateway URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace prometheus -l "app=prometheus-pushgateway,component=pushgateway" -o jsonpath="{.items[0].metadata.name}")
  kubectl --namespace prometheus port-forward $POD_NAME 9091

For more information on running Prometheus, visit:
https://prometheus.io/
```

12. Expose the Prometheus web application. In the terminal run the following commands:

Copy code

```
kubectl expose deployment prometheus-server \

 --namespace prometheus \

 --name=prometheus-server-loadbalancer \

 --type=LoadBalancer \
 --port=80 \
 --target-port=9090
```

13. Wait for the Prometheus application to start up. In the terminal run the following commands:

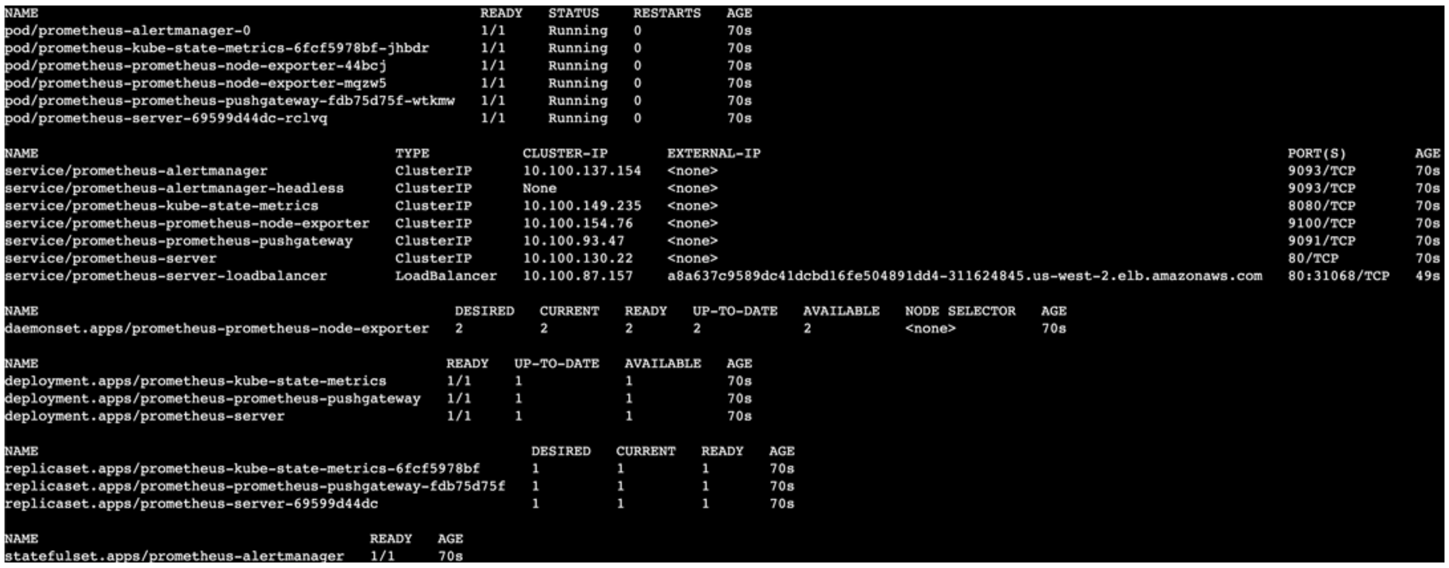
Copy code

```
{
kubectl wait --for=condition=available --timeout=300s deployment/prometheus-kube-state-metrics -n prometheus
kubectl wait --for=condition=available --timeout=300s deployment/prometheus-prometheus-pushgateway -n prometheus
kubectl wait --for=condition=available --timeout=300s deployment/prometheus-server -n prometheus
}
```

14. Confirm that the Prometheus cluster resources have all started successfully. In the terminal run the following commands:

Copy code

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



```
NAME                                                  READY   STATUS    RESTARTS   AGE
pod/prometheus-alertmanager-0                         1/1     Running   0          70s
pod/prometheus-kube-state-metrics-6fcf5978bf-jhbdr    1/1     Running   0          70s
pod/prometheus-prometheus-node-exporter-44bcj         1/1     Running   0          70s
pod/prometheus-prometheus-node-exporter-mqzw5         1/1     Running   0          70s
pod/prometheus-prometheus-pushgateway-fdb75d75f-wtkmw 1/1     Running   0          70s
pod/prometheus-server-69599d44dc-rclvq                1/1     Running   0          70s

NAME                                          TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)        AGE
service/prometheus-alertmanager               ClusterIP      10.100.137.154   <none>                                                                     9093/TCP       70s
service/prometheus-alertmanager-headless      ClusterIP      None             <none>                                                                     9093/TCP       70s
service/prometheus-kube-state-metrics         ClusterIP      10.100.149.235   <none>                                                                     8080/TCP       70s
service/prometheus-prometheus-node-exporter   ClusterIP      10.100.154.76    <none>                                                                     9100/TCP       70s
service/prometheus-prometheus-pushgateway     ClusterIP      10.100.93.47     <none>                                                                     9091/TCP       70s
service/prometheus-server                     ClusterIP      10.100.130.22    <none>                                                                     80/TCP         70s
service/prometheus-server-loadbalancer        LoadBalancer   10.100.87.157    a8a637c9589dc41dcbd16fe504891dd4-311624845.us-west-2.elb.amazonaws.com   80:31068/TCP   49s

NAME                                                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-prometheus-node-exporter   2         2         2       2            2           <none>          70s

NAME                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-kube-state-metrics       1/1     1            1           70s
deployment.apps/prometheus-prometheus-pushgateway   1/1     1            1           70s
deployment.apps/prometheus-server                   1/1     1            1           70s

NAME                                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-kube-state-metrics-6fcf5978bf      1         1         1       70s
replicaset.apps/prometheus-prometheus-pushgateway-fdb75d75f   1         1         1       70s
replicaset.apps/prometheus-server-69599d44dc                  1         1         1       70s

NAME                                      READY   AGE
statefulset.apps/prometheus-alertmanager  1/1     70s
```

15. Retrieve the Prometheus web application URL.

Confirm that the Prometheus ELB FQDN has propageted and resolves. In the terminal run the following commands:

Copy code

```
{
PROMETHEUS_ELB_FQDN=$(kubectl get svc -n prometheus prometheus-server-loadbalancer
-o jsonpath='{.status.loadBalancer.ingress[0].hostname}')

until nslookup $PROMETHEUS_ELB_FQDN >/dev/null 2>&1; do sleep 2 && echo waiting for DNS
to propagate...; done

curl -sD - -o /dev/null $PROMETHEUS_ELB_FQDN/graph

}
```

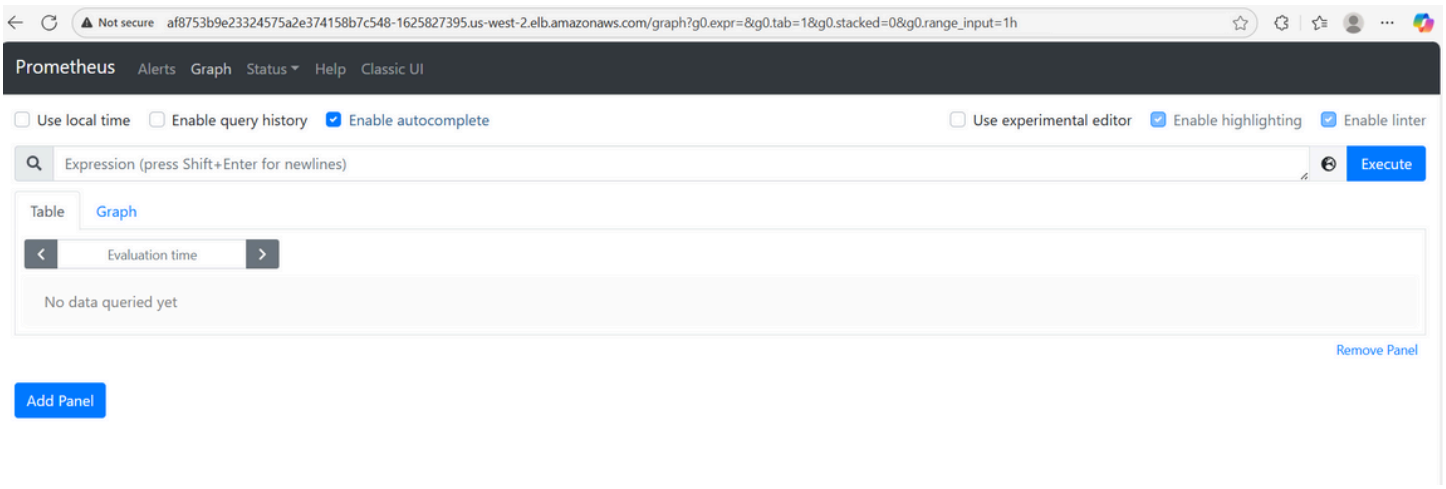
Note: DNS propagation can take up to 2-5 minutes, please be patient while the propagation proceeds - it will eventually complete.

16. Generate the Prometheus web application URL:

Copy code

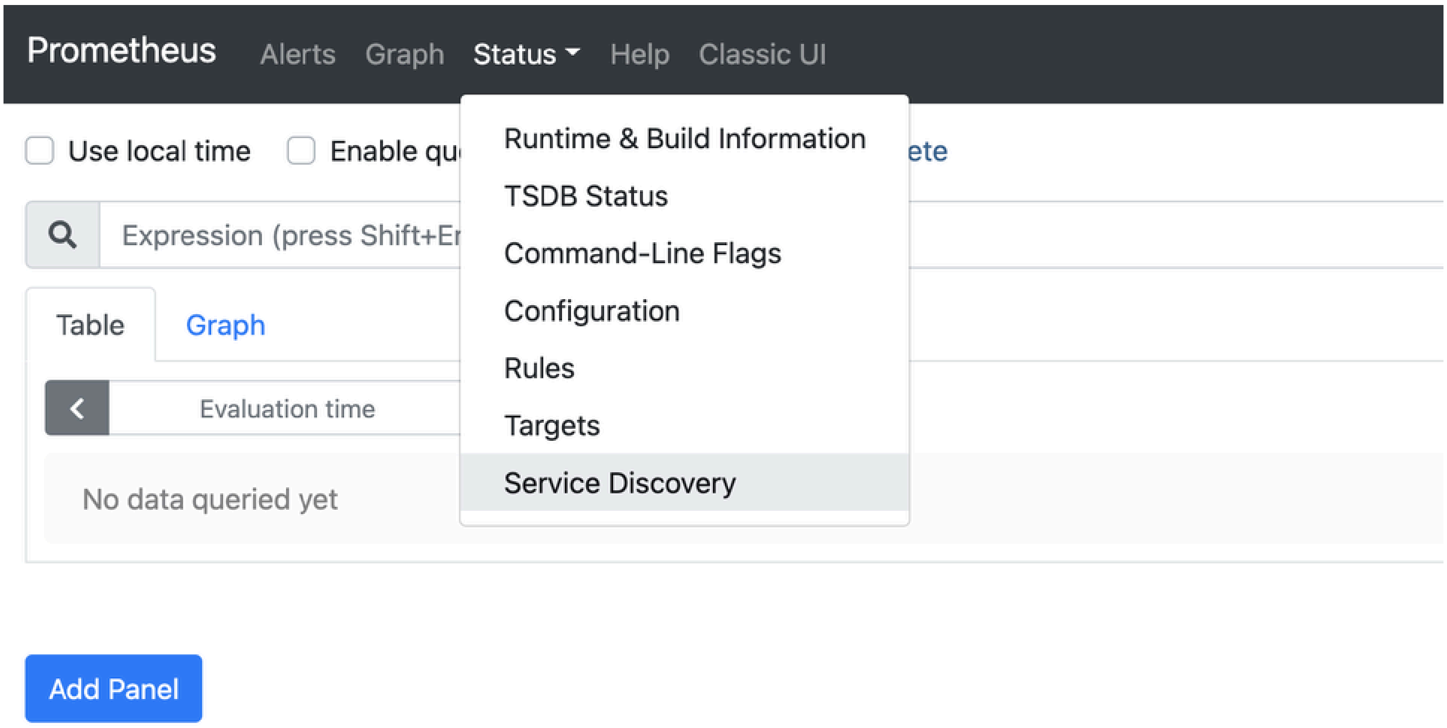
```
echo http://$PROMETHEUS_ELB_FQDN
```

17. Using your workstation's browser, copy the Prometheus URL from the previous output and browse to it:

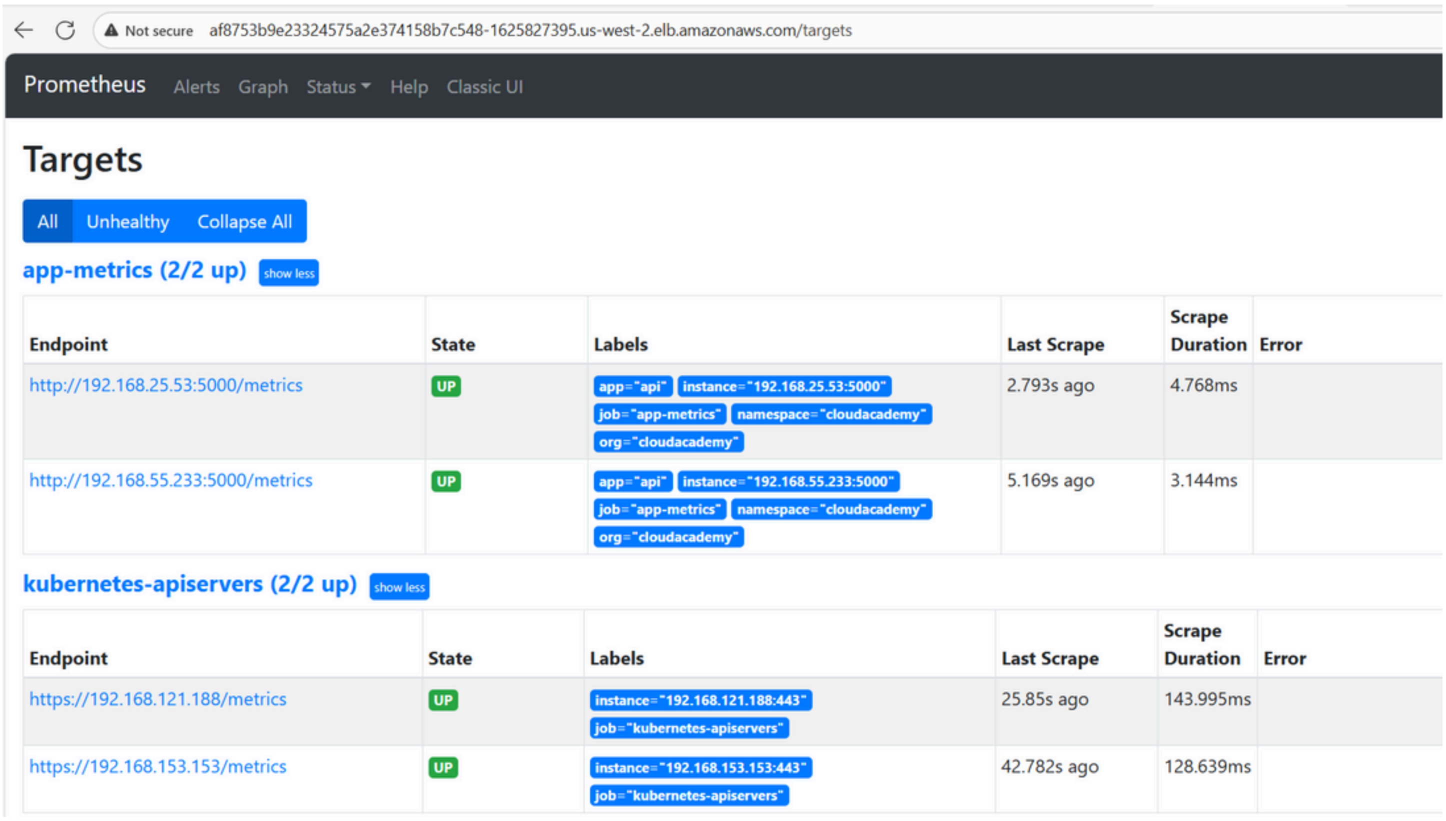


Not secure af8753b9e23324575a2e374158b7c548-1625827395.us-west-2.elb.amazonaws.com/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.range_input=1h

18. Within the Prometheus web application, click the **Status** top menu item and then select **Service Discovery**



19. Confirm that the following service discovery targets have been configured:


Not secure af8753b9e23324575a2e374158b7c548-1625827395.us-west-2.elb.amazonaws.com/targets
Prometheus
Alerts
Graph
Status
Help
Classic UI
All
Unhealthy
Collapse All
show less

Targets

app-metrics (2/2 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.25.53:5000/metrics	UP	app="api" instance="192.168.25.53:5000" job="app-metrics" namespace="cloudacademy" org="cloudacademy"	2.793s ago	4.768ms	
http://192.168.55.233:5000/metrics	UP	app="api" instance="192.168.55.233:5000" job="app-metrics" namespace="cloudacademy" org="cloudacademy"	5.169s ago	3.144ms	

kubernetes-apiservers (2/2 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.121.188/metrics	UP	instance="192.168.121.188:443" job="kubernetes-apiservers"	25.85s ago	143.995ms	
https://192.168.153.153/metrics	UP	instance="192.168.153.153:443" job="kubernetes-apiservers"	42.782s ago	128.639ms	

20. Within Prometheus, click the **Status** top menu item and then select **Targets**

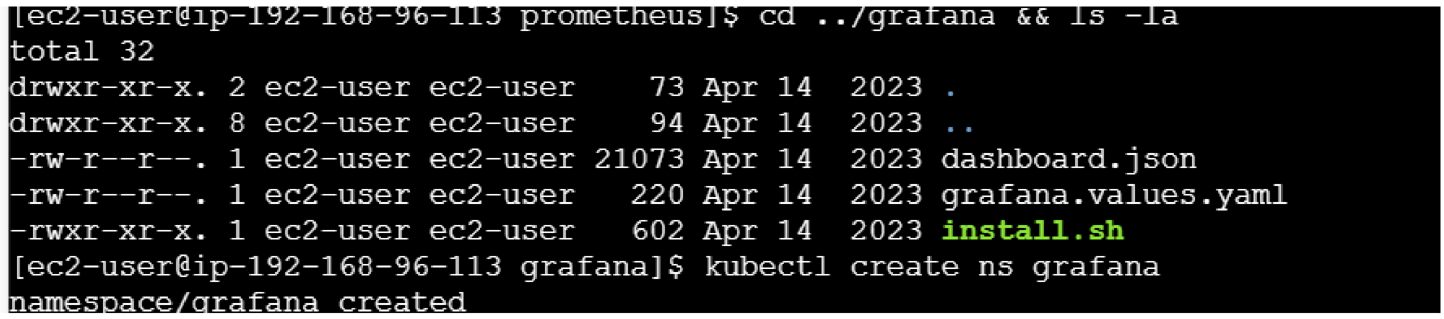
21. Confirm that the following Targets are available

22. Install **Grafana** into the cluster. Grafana will be deployed into the grafana namespace which needs to be created.

Navigate into the **grafana** directory and perform a directory listing. In the terminal execute the following commands:

Copy code

```
cd ../grafana && ls -la
```



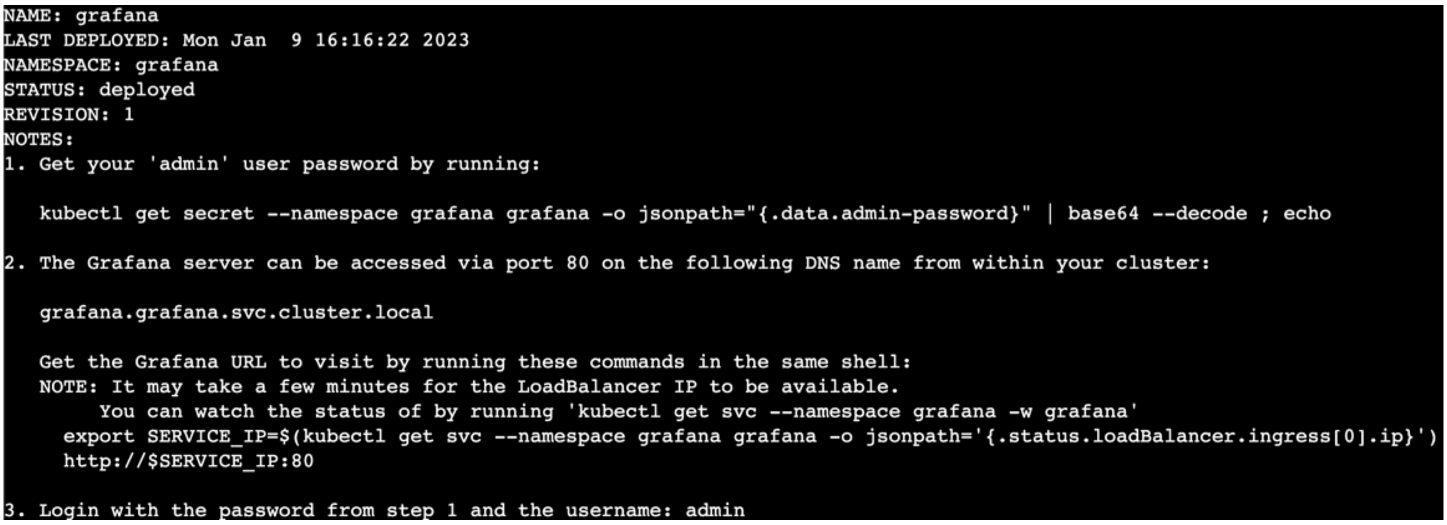
```
[ec2-user@ip-192-168-96-113 prometheus]$ cd ../grafana && ls -la
total 32
drwxr-xr-x. 2 ec2-user ec2-user    73 Apr 14  2023 .
drwxr-xr-x. 8 ec2-user ec2-user    94 Apr 14  2023 ..
-rw-r--r--. 1 ec2-user ec2-user 21073 Apr 14  2023 dashboard.json
-rw-r--r--. 1 ec2-user ec2-user   220 Apr 14  2023 grafana.values.yaml
-rwxr-xr-x. 1 ec2-user ec2-user   602 Apr 14  2023 install.sh
[ec2-user@ip-192-168-96-113 grafana]$ kubectl create ns grafana
namespace/grafana created
```

23. Create a new namespace named grafana. In the terminal run the follwing command:

```
kubectl create ns grafana
```

24. install Grafana using it's publicly available Helm Chart. Deploy Grafana into the grafana namespace. In the terminal execute the following commands:

```
{
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update
helm install grafana grafana/grafana \
 --namespace grafana \
 --set persistence.storageClassName="gp2" \
 --set persistence.enabled=true \
 --set adminPassword="EKS:l3t5g0" \
 --set service.type=LoadBalancer \
 --values ./grafana.values.yaml
}
```



```
NAME: grafana
LAST DEPLOYED: Mon Jan  9 16:16:22 2023
NAMESPACE: grafana
STATUS: deployed
REVISION: 1
NOTES:
1. Get your 'admin' user password by running:

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

2. The Grafana server can be accessed via port 80 on the following DNS name from within your cluster:

   grafana.grafana.svc.cluster.local

   Get the Grafana URL to visit by running these commands in the same shell:
   NOTE: It may take a few minutes for the LoadBalancer IP to be available.
        You can watch the status of by running 'kubectl get svc --namespace grafana -w grafana'
     export SERVICE_IP=$(kubectl get svc --namespace grafana grafana -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
     http://$SERVICE_IP:80

3. Login with the password from step 1 and the username: admin
```

Note: The PodSecurityPolicy deprecation warnings can be safely ignored.

25. Wait for the Grafana application to start up. In the terminal run the following command:

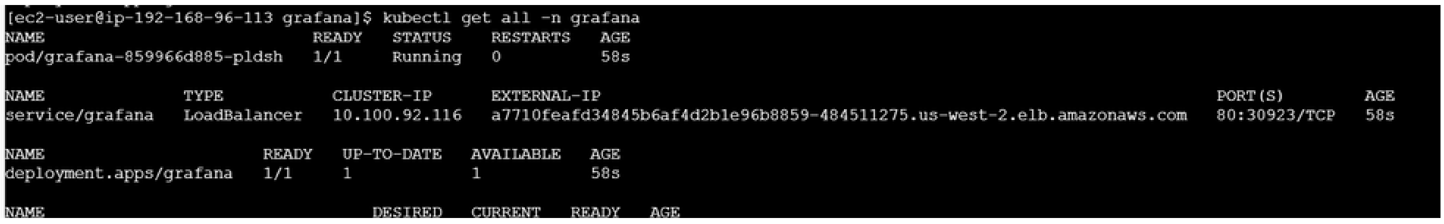
Copy code

```
kubectl wait --for=condition=available --timeout=300s deployment/grafana -n grafana
```

26. Confirm that the Grafana cluster resources have all started successfully. In the terminal run the following commands:

Copy code

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



```
[ec2-user@ip-192-168-96-113 grafana]$ kubectl get all -n grafana
NAME                           READY   STATUS    RESTARTS   AGE
pod/grafana-859966d885-pldsh   1/1     Running   0          58s

NAME              TYPE           CLUSTER-IP      EXTERNAL-IP                                                                PORT(S)        AGE
service/grafana   LoadBalancer   10.100.92.116   a7710feafd34845b6af4d2b1e96b8859-484511275.us-west-2.elb.amazonaws.com   80:30923/TCP   58s

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/grafana   1/1     1            1           58s

NAME                                 DESIRED   CURRENT   READY   AGE
```

27. Retrieve the Grafana web application URL.

Confirm that the Grafana ELB FQDN has propagated and resolves. In the terminal run the following commands:

Copy code

```
{
GRAFANA_ELB_FQDN=$(kubectl get svc -n grafana grafana -o
jsonpath='{.status.loadBalancer.ingress[0].hostname}')
until nslookup $GRAFANA_ELB_FQDN >/dev/null 2>&1; do sleep 2 && echo waiting for DNS to propagate...; done

curl -I $GRAFANA_ELB_FQDN/login

}
```

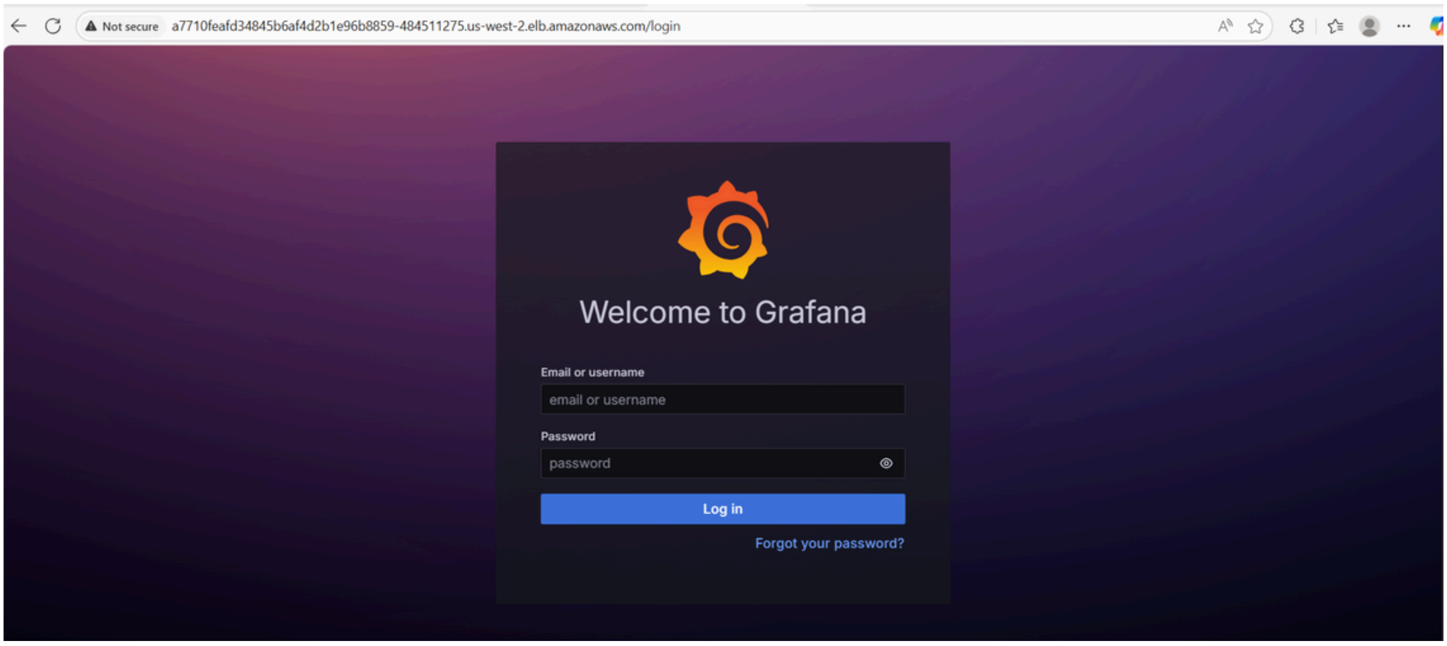
Note: DNS propagation can take up to 2-5 minutes, please be patient while the propagation proceeds - it will eventually complete.

28. Generate the Grafana web application URL:

Copy code

```
echo http://$GRAFANA_ELB_FQDN
```

29. Using your workstation's browser, copy the Grafana URL from the previous output and browse to it:



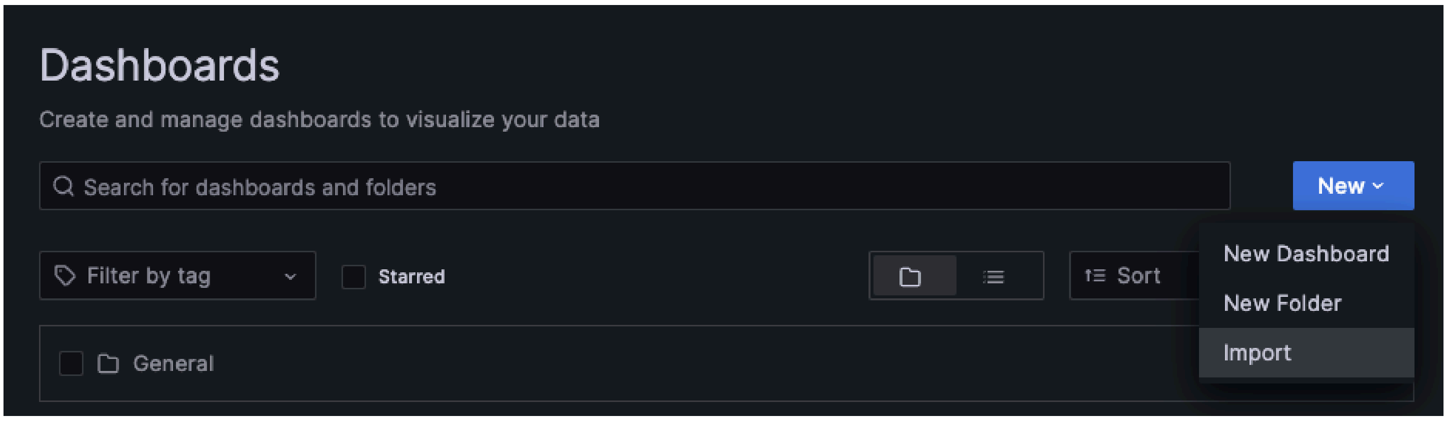
30. Login into Grafana, using the following credentials:

Email or username: admin

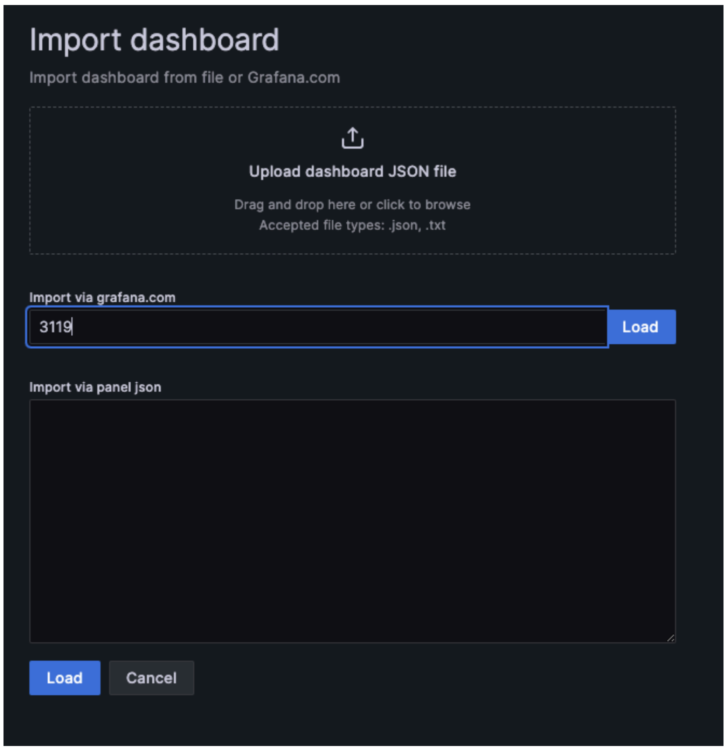
Password: EKS:l3t5g0

31. Once authenticated, the **Welcome to Grafana** home page is displayed

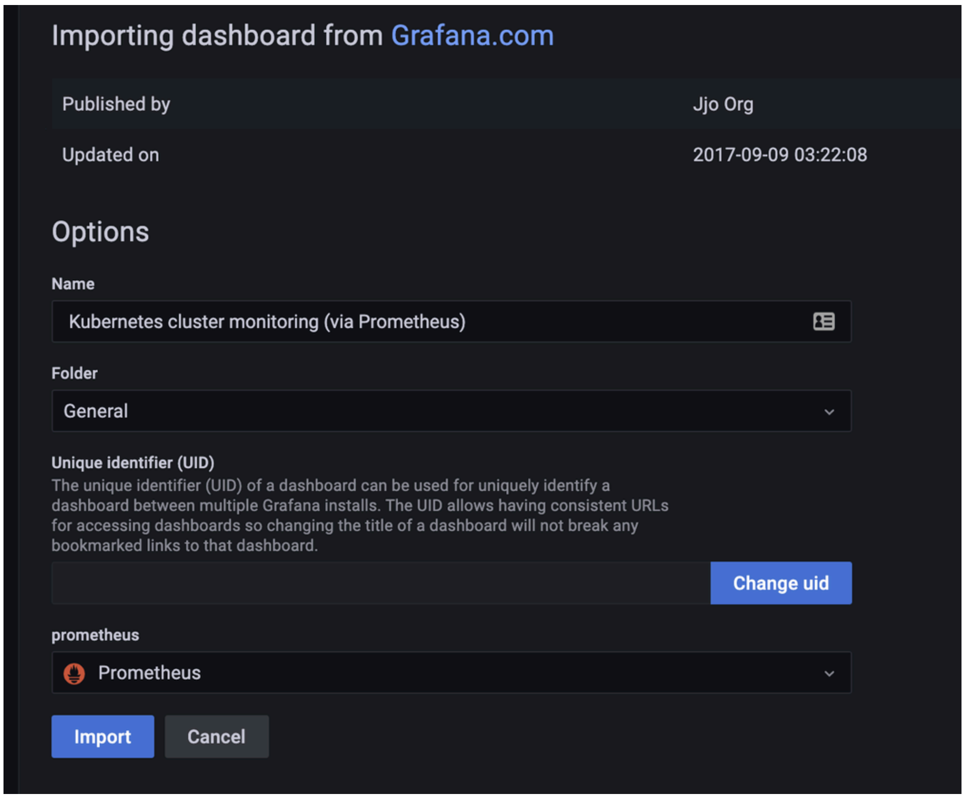
32. Click the **Dashboards** icon on the main left-hand side menu, and then select the dashboard **Import** option like so


Dashboards
Create and manage dashboards to visualize your data
Search for dashboards and folders
New
Filter by tag
Starred
Sort
New Dashboard
New Folder
Import
General

33. In the **Import** pane, enter the dashboard ID **3119** and click on the right hand side **Load** button


Import dashboard
Import dashboard from file or Grafana.com
Upload dashboard JSON file
Drag and drop here or click to browse
Accepted file types: .json, .txt
Import via grafana.com
3119
Load
Import via panel json
Load
Cancel

34. The **Import** pane now provides details for the **Kubernetes cluster monitoring** dashboard that you are about to install. Select and set the **Prometheus** datasource at the bottom of the pane, then click **Import** to proceed


Importing dashboard from Grafana.com

Published by Jjo Org

Updated on 2017-09-09 03:22:08

Options

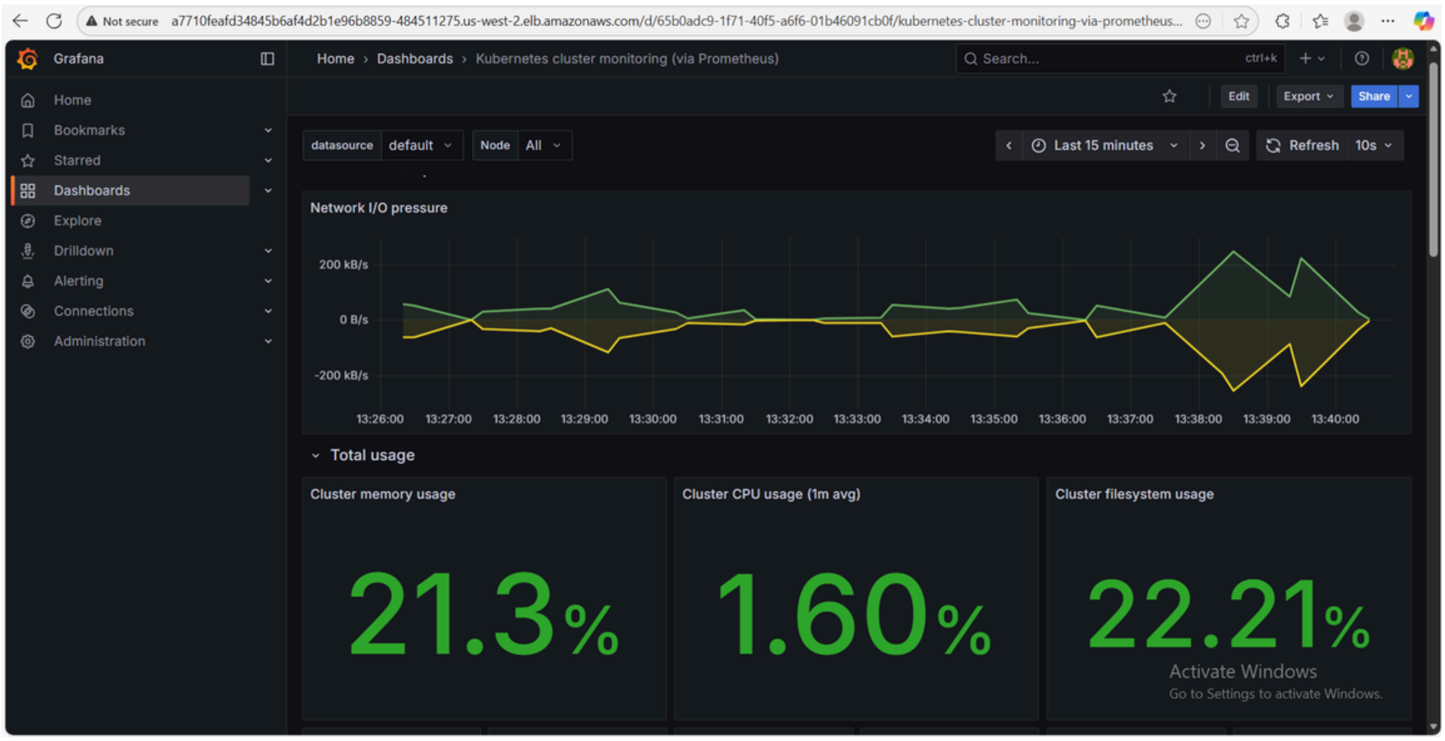
Name: Kubernetes cluster monitoring (via Prometheus)

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.

prometheus: Prometheus

35. Grafana now loads and displays the prebuilt **Kubernetes cluster monitoring** dashboard and immediately renders various visualisations using live monitoring data streams taken from Prometheus. The dashboard view automatically refreshes every 10 seconds


Not secure a7710feafd34845b6af4d2b1e96b8859-484511275.us-west-2.elb.amazonaws.com/d/65b0adc9-1f71-40f5-a6f6-01b46091cb0f/kubernetes-cluster-monitoring-via-prometheus...

Home › Dashboards › Kubernetes cluster monitoring (via Prometheus)

datasource default · Node All · Last 15 minutes · Refresh 10s

Network I/O pressure

Total usage

Cluster memory usage 21.3%

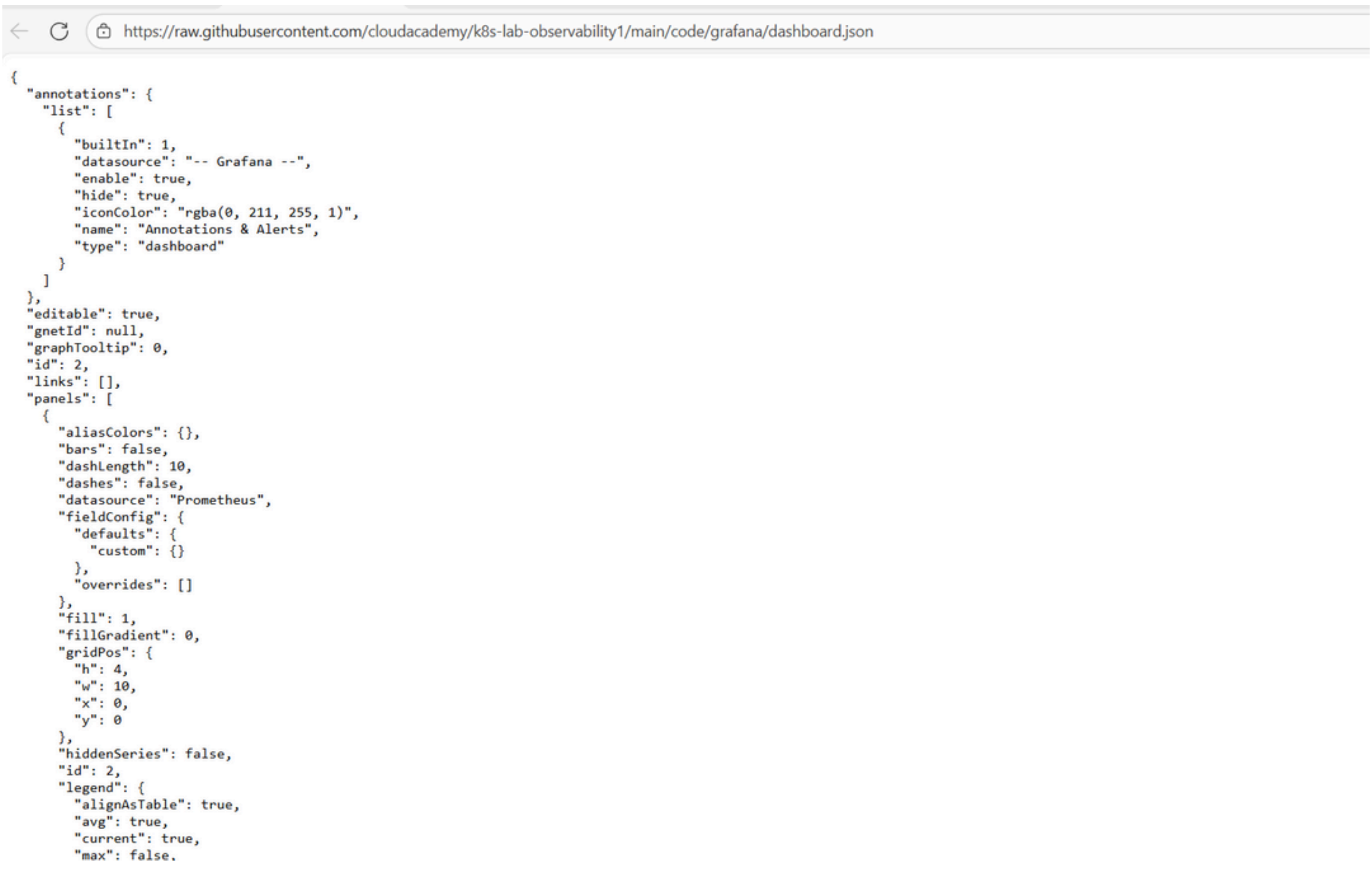
Cluster CPU usage (1m avg) 1.60%

Cluster filesystem usage 22.21%

36. Return to the **Dashboards** on the main left-hand side menu, and again select the dashboard **Import** option like before

37. Open a new browser tab and navigate to the following sample API monitoring dashboard URL

https://raw.githubusercontent.com/cloudacademy/k8s-lab-observability1/main/code/grafana/dashboard.json

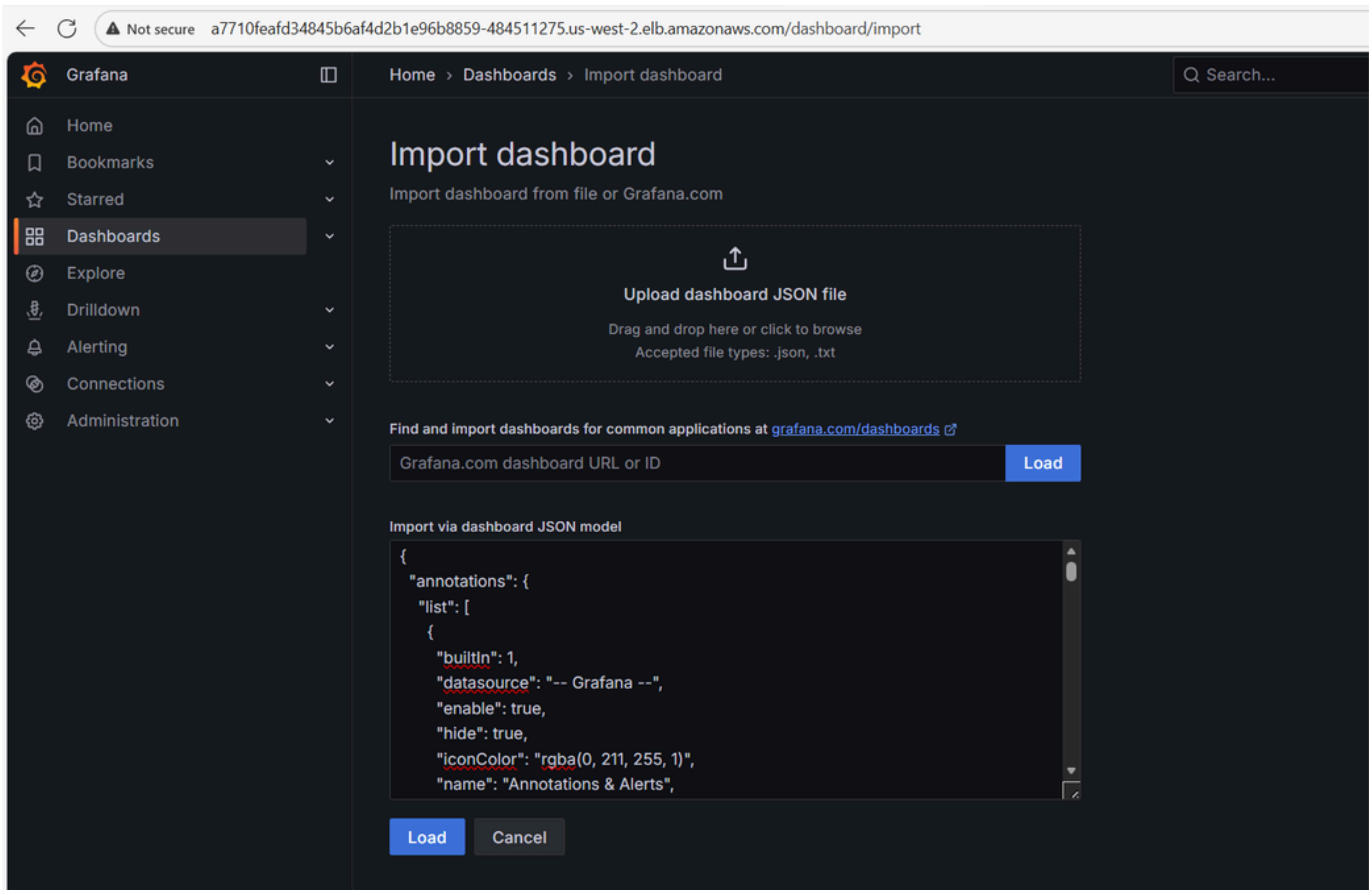


https://raw.githubusercontent.com/cloudacademy/k8s-lab-observability1/main/code/grafana/dashboard.json

```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "gnetId": null,
  "graphTooltip": 0,
  "id": 2,
  "links": [],
  "panels": [
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "Prometheus",
      "fieldConfig": {
        "defaults": {
          "custom": {}
        },
        "overrides": []
      },
      "fill": 1,
      "fillGradient": 0,
      "gridPos": {
        "h": 4,
        "w": 10,
        "x": 0,
        "y": 0
      },
      "hiddenSeries": false,
      "id": 2,
      "legend": {
        "alignAsTable": true,
        "avg": true,
        "current": true,
        "max": false,
```

Note: The same dashboard.json file is located in the project directory (./code/grafana/dashboard.json) and can be copied directly from there if easier.

38. Copy the sample API monitoring dashboard JSON to the local clipboard:


Not secure a7710feafd34845b6af4d2b1e96b8859-484511275.us-west-2.elb.amazonaws.com/dashboard/import
Grafana
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
Alerting
Connections
Administration
Home › Dashboards › Import dashboard
Search...

Import dashboard

Import dashboard from file or Grafana.com

Upload dashboard JSON file

Drag and drop here or click to browse
Accepted file types: .json, .txt

Find and import dashboards for common applications at grafana.com/dashboards

Grafana.com dashboard URL or ID | Load

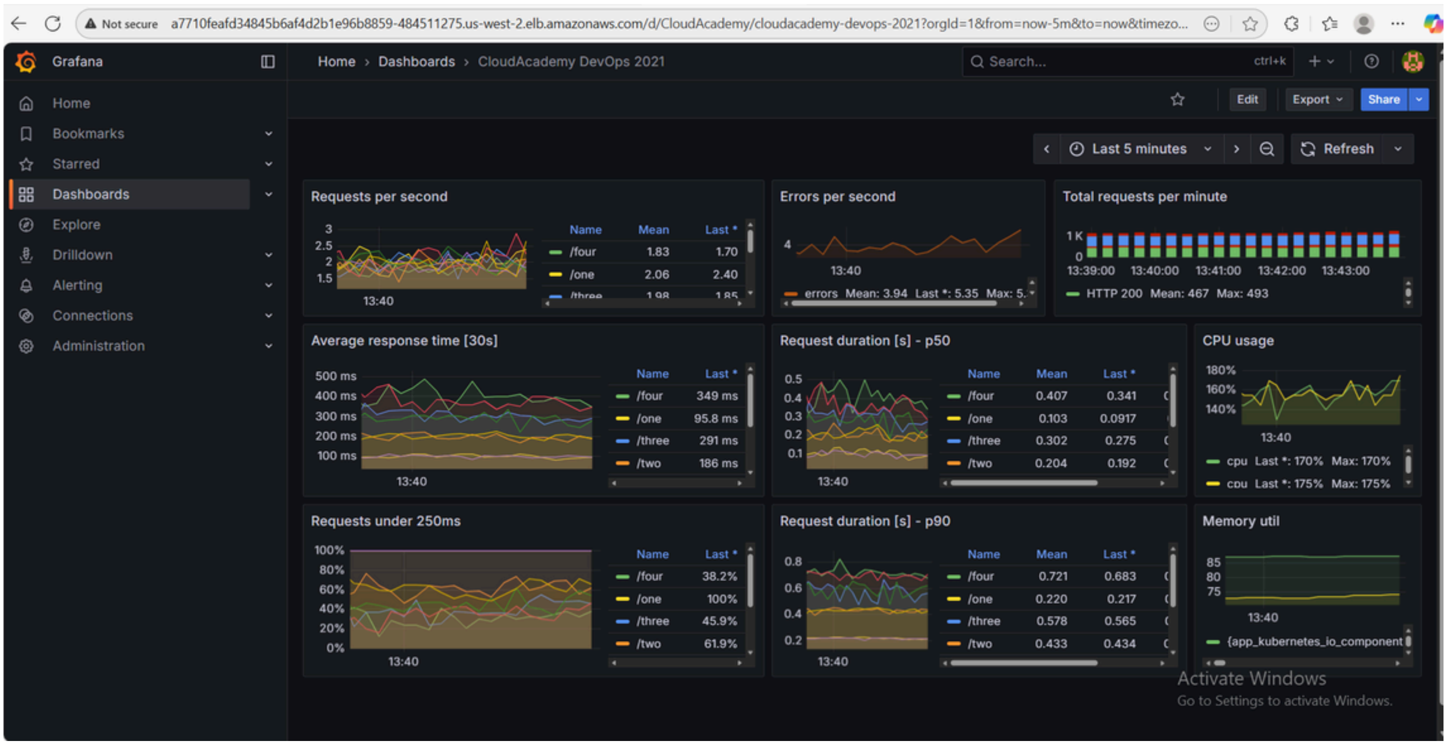
Import via dashboard JSON model

```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
```

Load   Cancel

39. In the **Import** pane, paste the contents of the local clipboard containing the sample API monitoring dashboard configuration JSON into the **Import via panel json** input box. Click the bottom **Load** button to import the dashboard:

40. The **Import** pane now provides details for the sample **API monitoring** dashboard that you are about to install. Click the **Import** button to proceed:


Not secure a7710feafd34845b6af4d2b1e96b8859-484511275.us-west-2.elb.amazonaws.com/d/CloudAcademy/cloudacademy-devops-2021?orgId=1&from=now-5m&to=now&timezo...
Grafana
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
Alerting
Connections
Administration
Home › Dashboards › CloudAcademy DevOps 2021
Search... ctrl+k
Edit
Export
Share
Last 5 minutes
Refresh
Requests per second
Errors per second
Total requests per minute
Average response time [30s]
Request duration [s] - p50
CPU usage
Requests under 250ms
Request duration [s] - p90
Memory util
13:40
errors Mean: 3.94 Last *: 5.35 Max: 5.
HTTP 200 Mean: 467 Max: 493
cpu Last *: 170% Max: 170%
cpu Last *: 175% Max: 175%
{app_kubernetes_io_component
Activate Windows
Go to Settings to activate Windows.

Requests per second

Name	Mean	Last *
/four	1.83	1.70
/one	2.06	2.40
/three	1.98	1.85

Average response time [30s]

Name	Last *
/four	349 ms
/one	95.8 ms
/three	291 ms
/two	186 ms

Request duration [s] - p50

Name	Mean	Last *
/four	0.407	0.341
/one	0.103	0.0917
/three	0.302	0.275
/two	0.204	0.192

Requests under 250ms

Name	Last *
/four	38.2%
/one	100%
/three	45.9%
/two	61.9%

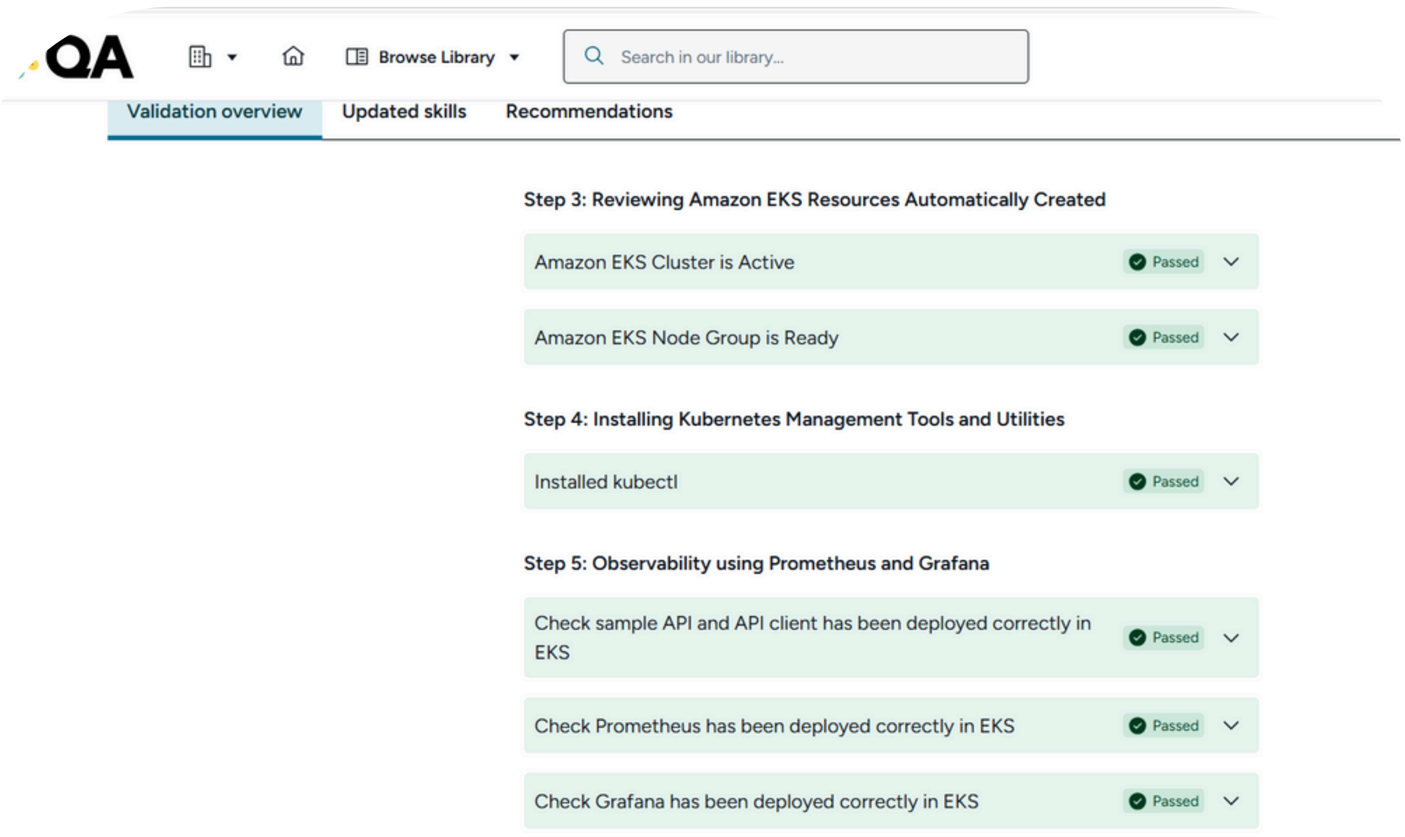
Request duration [s] - p90

Name	Mean	Last *
/four	0.721	0.683
/one	0.220	0.217
/three	0.578	0.565
/two	0.433	0.434

41. Grafana now loads and displays the prebuilt sample **CloudAcademy DevOps 2021 API** dashboard and immediately renders various visualisations using live monitoring data streams taken from Prometheus. The dashboard view automatically refreshes every 5 seconds:

Summary

In this lab step, you installed a sample API and API client. Next, using Helm you installed Prometheus into the prometheus namespace within the cluster. You set up and exposed the Prometheus web admin interface using a LoadBalancer based Service resource. You then browsed to the Prometheus web admin interface and confirmed that service discovery was working correctly, and the expected targets had been registered. Using Helm you then installed Grafana into the grafana namespace. You then authenticated into the Grafana web admin interface and setup 2 dashboards, one for the cluster, and another one for the sample API you previously deployed. Grafana then loaded the dashboards and starting pulling live monitoring data and metrics directly from the Prometheus data source.



Step 3: Reviewing Amazon EKS Resources Automatically Created

Amazon EKS Cluster is Active — Passed

Amazon EKS Node Group is Ready — Passed

Step 4: Installing Kubernetes Management Tools and Utilities

Installed kubectl — Passed

Step 5: Observability using Prometheus and Grafana

Check sample API and API client has been deployed correctly in EKS — Passed

Check Prometheus has been deployed correctly in EKS — Passed

Check Grafana has been deployed correctly in EKS — Passed