Kubernetes 5 notes

Helm charts

Promethues

Grafana

EFK Stack

To install in Amazon Linux VM -->

sudo yum install git

sudo yum install java

sudo yum install maven

Ubuntu Linux VM

sudo apt install git

sudo apt install maven

HELM

HELM is a package manager, which is used to install required softwares in Kubernetes cluster. With metrics-server, it is very complex and time-consuming. Similar to yum/apt package manager in Linux distribution, Helm allows us to install applications on Kubernetes cluster. Helm uses charts in order to achieve this. Charts refer to collection of configuration files (manifest yml)

Chart repository

K8s API server

Helm client

Kubernetes cluster

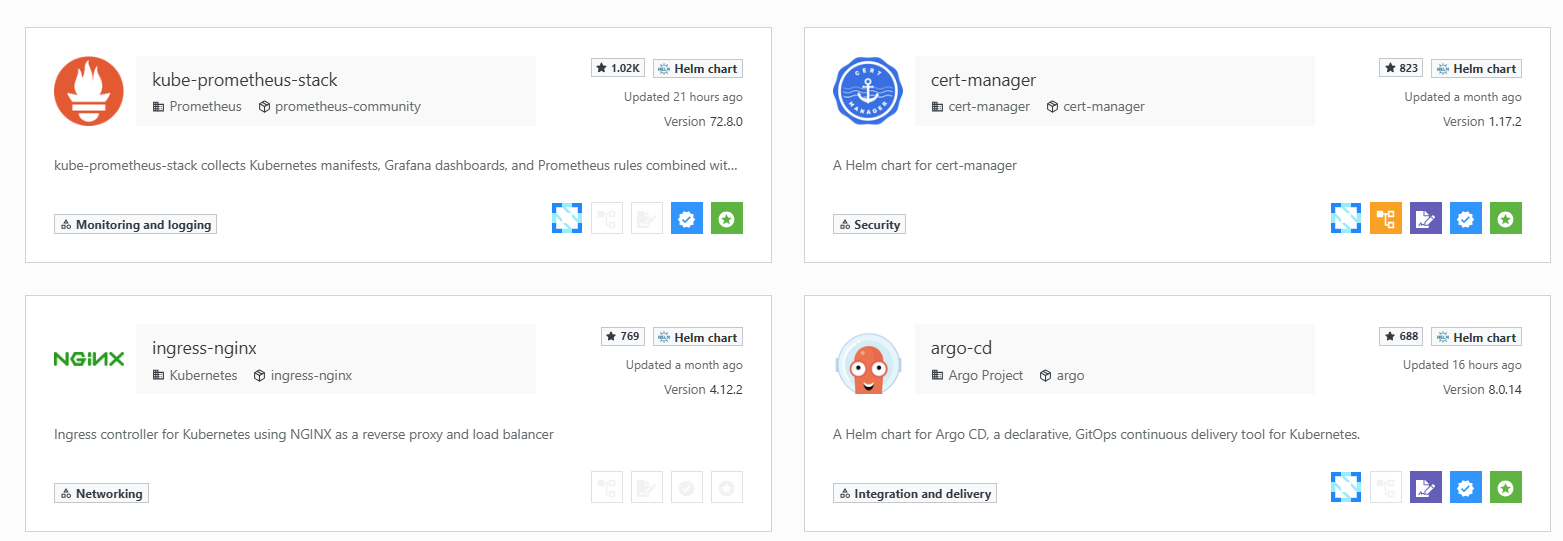
Helm talks to Kubernetes API server in order to install packages in the cluster

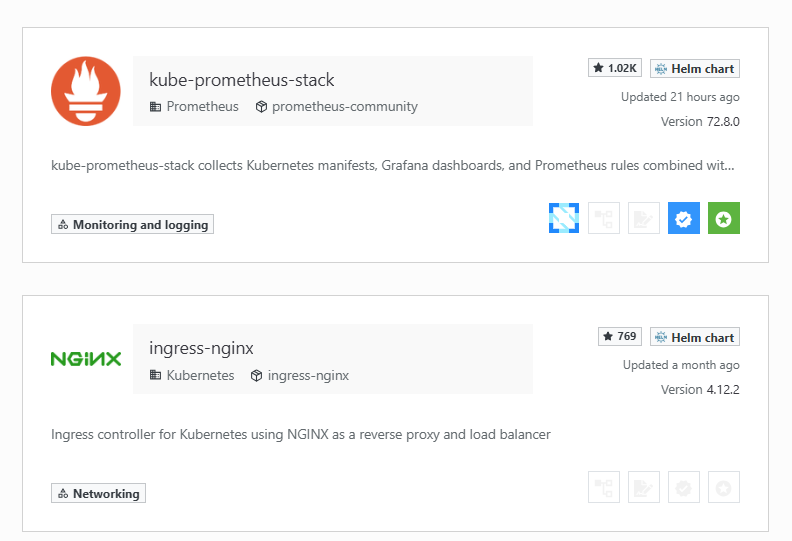
Chart repository is the collection of required configuration files (yaml files required to install specific software). With the help of Chart repository, Helm client will go and talk to K8s API server and install all required software specified in charts in Chart repository

Helm is the best way to find, share and use softwares built for Kubernetes

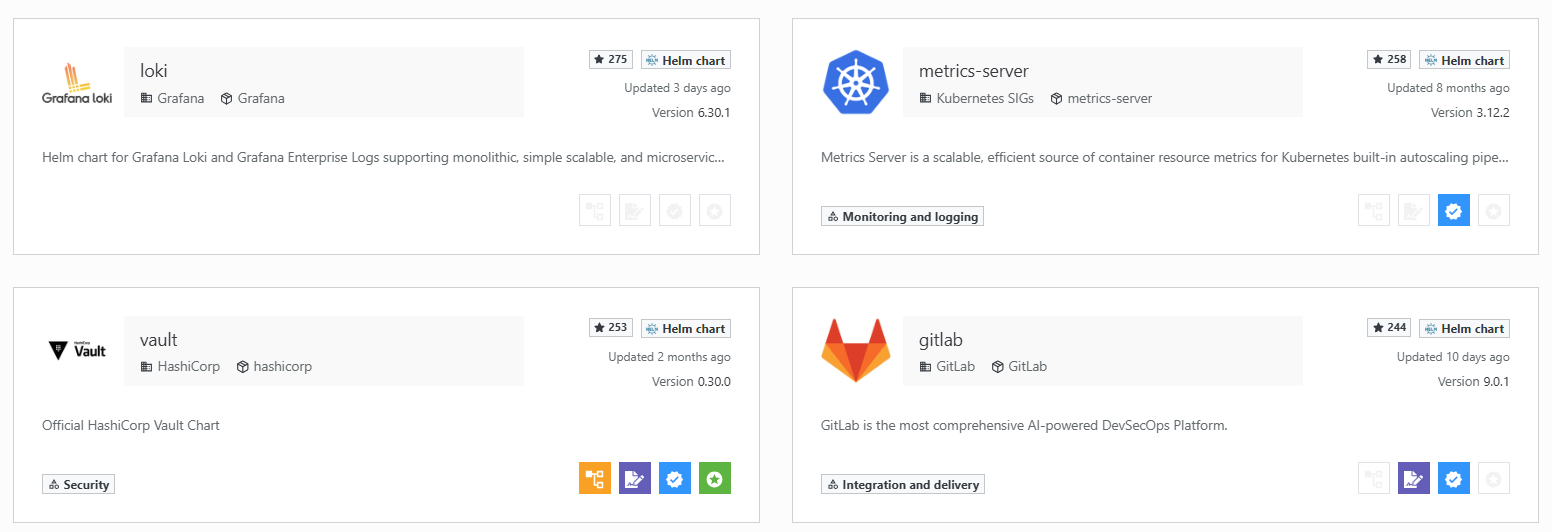
<https://artifacthub.io/>

<https://artifacthub.io/packages/search>

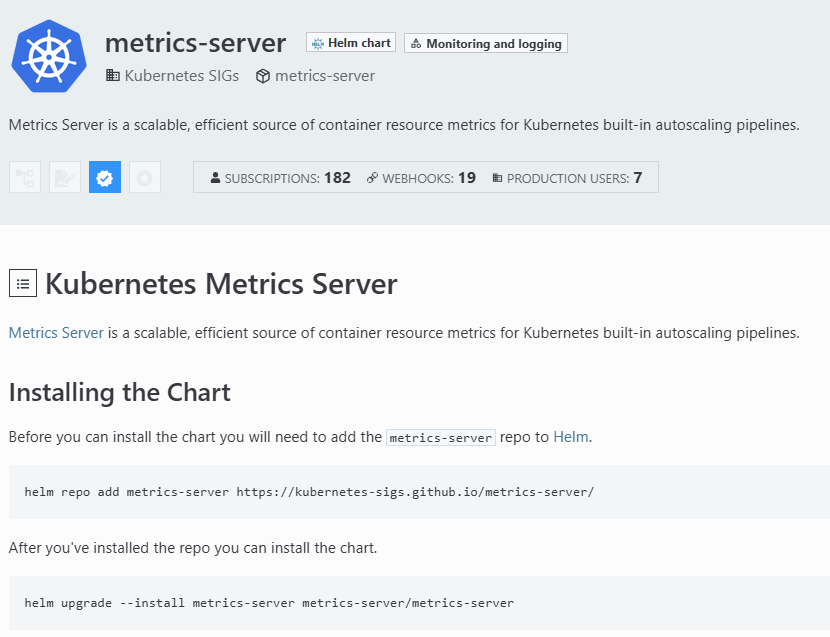




If you want ‘metrics-server’



<https://artifacthub.io/packages/helm/metrics-server/metrics-server>



If youo run this one command, it will take care of all metrics-server installation

helm repo add metrics-server https://kubernetes-sigs.github.io/metrics-server/

Helm charts --> Using Helm charts, we can install Prometheus server, Grafana server, Metrics server, Gitlab etc

What’s Prometheus or Grafana server?

Monitoring tools

Prometheus is an open-source monitoring tool, which is used for altering purpose, it collects and stores metrics of the entire cluster. Grafana shows that visually. Grafana gives interactive visualization of what’s happening. If you want to monitor specific application in this entire cluster, then EFK comes into picture. Actual tool that will monitor your entire cluster is Prometheus. Grafana will talk to Prometheus and Promotheus will talk to metrics-server and get the information. Grafana will again talk to Prometheus and visually show you how your entire Kubernetes cluster is behaving.

Prometheus -> it is an open-source system monitoring and alerting toolkit

It collects and stores its metrics as a time-series data

It provides out-of-box monitoring capabilities for k8s

Grafana

It is an analysis and monitoring tool, which provides visualization for monitoring of your k8s cluster

It provides graphs, charts and alerts for web when connected to supported data sources

Grafana connects with Prometheus for data source

Which is the easiest way to install servers into Kubernetes cluster?

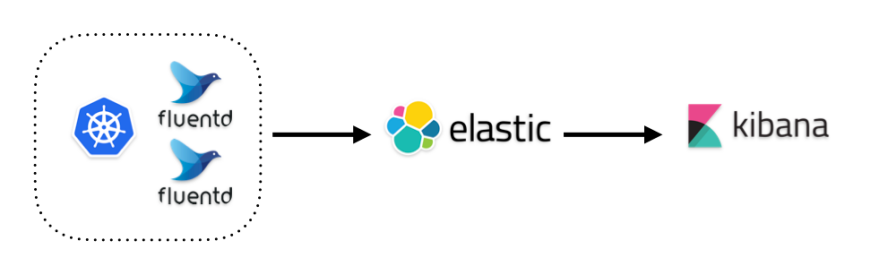
Helm

EFK refers to Elastic search, FluentD, Kibana --> all these are different products.

By using kubectl logs, could you get to know what’s happening in a particular pod?

Splunk is an alternative for EFK. Splunk has upper edge without a doubt but it is paid

EFK stack provides centralized logging in order to identify problems with servers or applications. It will help us to search all the logs in a single place. Using FluentD, it will read all the logs and those logs will be stored within the Elastic search. Then if you want to see those logs, then we have something called as Kibana. Kibana provides UI for this Elastic search. We need to understand daemonset, statefulset, deployment.



Stores logs

Kibana

Elastic search

FluentD

Logs

Microservice1

End user

Logs

Microservice2

Logs

Microservice3

FluentD will read logs and index into Elastic search, stores logs in Elastic search.How do we know these logs are stored here. What component is there to know the logs stored here? Kibana, it provides UI for clients

It will take all the logs and index those logs.

If you want to install or update these servers in the Kubernetes cluster, which component is used? HELM

If you want to monitor the entire Kubernetes cluster, which tool is used? Prometheus

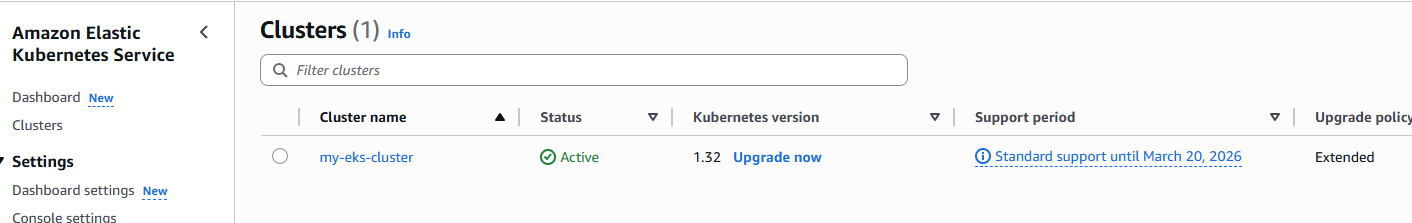
In order to display, cluster monitoring in graphical format? Grafana

If you want to monitor your application which are there, then EFK comes into picture

Create K8s cluster

eksctl create cluster --name my-eks-cluster --region ca-central-1 --node-type t2.medium --zones ca-central-1a,ca-central-1b

Go to EKS



Go into EKS host VM

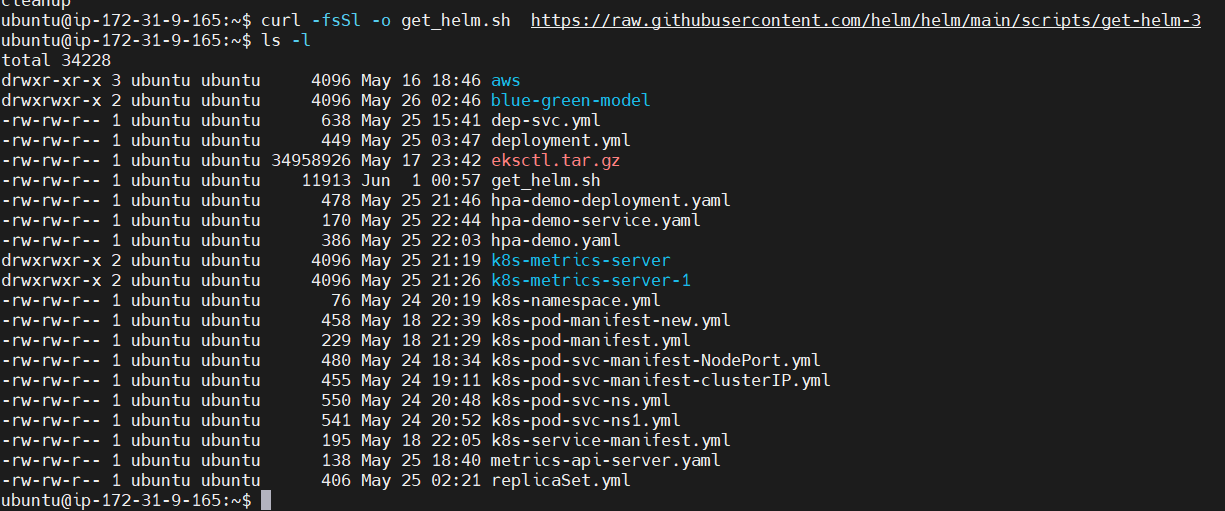
HELM installation:

$ curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

Run this without bash

ubuntu@ip-172-31-9-165:~$ curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

ubuntu@ip-172-31-9-165:~$ curl -fsSl -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3



-rw-rw-r-- 1 ubuntu ubuntu 11913 Jun 1 00:57 get\_helm.sh

We have read and write permission but not execute. So ‘chmod 700 get\_helm.sh’ to give execute permission. To check it is is installed ‘helm’

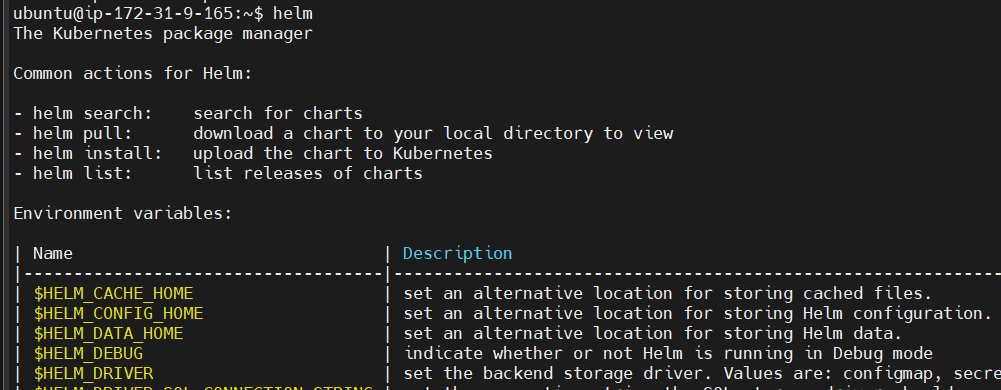
ubuntu@ip-172-31-9-165:~$ chmod 700 get\_helm.sh

-rwx------ 1 ubuntu ubuntu 11913 Jun 1 00:57 get\_helm.sh

ubuntu@ip-172-31-9-165:~$ ./get\_helm.sh

Helm v3.18.1 is already latest

Helm is successfully installed



ubuntu@ip-172-31-9-165:~$ kubectl top pods

No resources found in default namespace.

ubuntu@ip-172-31-9-165:~$ kubectl top nodes

NAME CPU(cores) CPU(%) MEMORY(bytes) MEMORY(%)

ip-192-168-1-194.ca-central-1.compute.internal 30m 1% 535Mi 15%

ip-192-168-37-54.ca-central-1.compute.internal 31m 1% 533Mi 15%

If you want to install the metrics-server,

-> Execute manifest yml files

-> Use Helm charts

ubuntu@ip-172-31-9-165:~$ helm repo ls

Error: no repositories to show

ubuntu@ip-172-31-9-165:~$ helm repo add metrics-server https://kubernetes-sigs.github.com/metrics-server

ubuntu@ip-172-31-9-165:~$ helm repo add metrics-server https://kubernetes-sigs.github.io/metrics-server

"metrics-server" has been added to your repositories

ubuntu@ip-172-31-9-165:~$

ubuntu@ip-172-31-9-165:~$ helm repo update

Hang tight while we grab the latest from your chart repositories...

...Successfully got an update from the "metrics-server" chart repository

Update Complete. ⎈Happy Helming!⎈

ubuntu@ip-172-31-9-165:~$ helm repo ls

NAME URL

metrics-server https://kubernetes-sigs.github.io/metrics-server

ubuntu@ip-172-31-9-165:~$ helm upgrade --install metrics-server metrics-server/metrics-server

ubuntu@ip-172-31-9-165:~$ helm upgrade --install metrics-server metrics-server/metrics-server

Release "metrics-server" does not exist. Installing it now.

Error: Unable to continue with install: ClusterRole "system:metrics-server-aggregated-reader" in namespace "" exists and cannot be imported into the current release: invalid ownership metadata; label validation error: key "app.kubernetes.io/managed-by" must equal "Helm": current value is "EKS"; annotation validation error: missing key "meta.helm.sh/release-name": must be set to "metrics-server"; annotation validation error: missing key "meta.helm.sh/release-namespace": must be set to "default"

These things are already there in the system so we delete one by one

ubuntu@ip-172-31-9-165:~$ kubectl delete deployment metrics-server -n kube-system

deployment.apps "metrics-server" deleted

ubuntu@ip-172-31-9-165:~$ kubectl delete clusterrole system:metrics-server-aggregated-reader

clusterrole.rbac.authorization.k8s.io "system:metrics-server-aggregated-reader" deleted

ubuntu@ip-172-31-9-165:~$ kubectl delete clusterrolebinding metrics-server:system:auth-delegator

clusterrolebinding.rbac.authorization.k8s.io "metrics-server:system:auth-delegator" deleted

ubuntu@ip-172-31-9-165:~$ kubectl delete apiservice v1beta1.metrics.k8s.io

apiservice.apiregistration.k8s.io "v1beta1.metrics.k8s.io" deleted

ubuntu@ip-172-31-9-165:~$ kubectl delete service metrics-server -n kube-system

service "metrics-server" deleted

ubuntu@ip-172-31-9-165:~$ kubectl delete serviceaccount metrics-server -n kube-systen

Error from server (NotFound): serviceaccounts "metrics-server" not found

ubuntu@ip-172-31-9-165:~$ kubectl delete serviceaccount metrics-server -n kube-system

serviceaccount "metrics-server" deleted

ubuntu@ip-172-31-9-165:~$ helm upgrade --install metrics-server metrics-server/metrics-server

Release "metrics-server" does not exist. Installing it now.

Error: Unable to continue with install: ClusterRole "system:metrics-server" in namespace "" exists and cannot be imported into the current release: invalid ownership metadata; label validation error: key "app.kubernetes.io/managed-by" must equal "Helm": current value is "EKS"; annotation validation error: missing key "meta.helm.sh/release-name": must be set to "metrics-server"; annotation validation error: missing key "meta.helm.sh/release-namespace": must be set to "default"

ubuntu@ip-172-31-9-165:~$ kubectl get clusterrole system:metrics-server -o yaml

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

creationTimestamp: "2025-05-31T23:51:10Z"

labels:

app.kubernetes.io/instance: metrics-server

app.kubernetes.io/managed-by: EKS

app.kubernetes.io/name: metrics-server

app.kubernetes.io/version: 0.7.2

name: system:metrics-server

resourceVersion: "970"

uid: 41670649-cfa2-404c-aa5a-32eea7cd85b3

rules:

- apiGroups:

- ""

resources:

- nodes/metrics

verbs:

- get

- apiGroups:

- ""

resources:

- pods

- nodes

- namespaces

- configmaps

verbs:

- get

- list

- watch

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

creationTimestamp: "2025-05-31T23:51:10Z"

labels:

app.kubernetes.io/instance: metrics-server

app.kubernetes.io/managed-by: EKS

app.kubernetes.io/name: metrics-server

app.kubernetes.io/version: 0.7.2

name: system:metrics-server

resourceVersion: "970"

uid: 41670649-cfa2-404c-aa5a-32eea7cd85b3

rules:

- apiGroups:

- ""

resources:

- nodes/metrics

verbs:

- get

- apiGroups:

- ""

resources:

- pods

- nodes

- namespaces

- configmaps

verbs:

- get

- list

- watch

Change EKS to Helm

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

creationTimestamp: "2025-05-31T23:51:10Z"

labels:

app.kubernetes.io/instance: metrics-server

app.kubernetes.io/managed-by: Helm

app.kubernetes.io/name: metrics-server

app.kubernetes.io/version: 0.7.2

name: system:metrics-server

resourceVersion: "32569"

uid: 41670649-cfa2-404c-aa5a-32eea7cd85b3

rules:

- apiGroups:

- ""

resources:

- nodes/metrics

verbs:

- get

- apiGroups:

- ""

resources:

- pods

- nodes

- namespaces

- configmaps

verbs:

- get

- list

- watch

ubuntu@ip-172-31-9-165:~$ helm upgrade --install metrics-server metrics-server/metrics-server

Release "metrics-server" does not exist. Installing it now.

Error: Unable to continue with install: ClusterRole "system:metrics-server" in namespace "" exists and cannot be imported into the current release: invalid ownership metadata; annotation validation error: missing key "meta.helm.sh/release-name": must be set to "metrics-server"; annotation validation error: missing key "meta.helm.sh/release-namespace": must be set to "default"

kubectl patch clusterrole system:metrics-server --type='merge' -p '{

"metadata": {

“annotations”: {

“meta.helm.sh/release-namespace”: “kube-system”

}

}

}’

kubectl patch clusterrolebinding system:metrics-server --type=’merge’ -p {

“metadata”: {

“annotations”: {

“meta.helm.sh/release-namespace”: “kube-system”

}

}

}’

ubuntu@ip-172-31-9-165:~$ kubectl patch clusterrole system:metrics-server --type='merge' -p '{

"metadata": {

"annotations": {

"meta.helm.sh/release-namespace": "kube-system"

}

}

}'

kubectl patch clusterrolebinding system:metrics-server --type='merge' -p '{

"metadata": {

"annotations": {

"meta.helm.sh/release-namespace": "kube-system"

}

}

}'

clusterrole.rbac.authorization.k8s.io/system:metrics-server patched (no change)

clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server patched

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

annotations:

meta.helm.sh/release-name: metrics-server

meta.helm.sh/release-namespace: kube-system

creationTimestamp: "2025-05-31T23:51:10Z"

labels:

app.kubernetes.io/instance: metrics-server

app.kubernetes.io/managed-by: Helm

app.kubernetes.io/name: metrics-server

app.kubernetes.io/version: 0.7.2

name: system:metrics-server

resourceVersion: "35453"

uid: 41670649-cfa2-404c-aa5a-32eea7cd85b3

rules:

- apiGroups:

- ""

resources:

- nodes/metrics

verbs:

- get

- apiGroups:

- ""

resources:

- pods

- nodes

- namespaces

- configmaps

verbs:

- get

- list

- watch

Run this in case of error

eksctl delete addon --name metrics-server --cluster my-eks-cluster --region ca-central-1

eksctl create addon --name metrics-server --cluster my-eks-cluster --region ca-central-1 --force

ubuntu@ip-172-31-9-165:~$ helm upgrade --install metrics-server metrics-server/metrics-server --namespace kube-system

Release "metrics-server" does not exist. Installing it now.

NAME: metrics-server

LAST DEPLOYED: Sun Jun 1 03:32:18 2025

NAMESPACE: kube-system

STATUS: deployed

REVISION: 1

TEST SUITE: None

NOTES:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Metrics Server \*

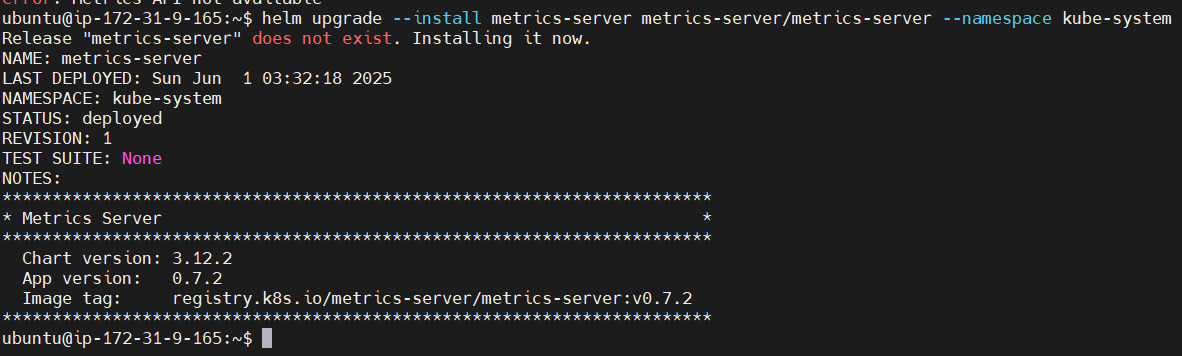
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Chart version: 3.12.2

App version: 0.7.2

Image tag: registry.k8s.io/metrics-server/metrics-server:v0.7.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



ubuntu@ip-172-31-9-165:~$ kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

aws-node-p8twr 2/2 Running 0 3h44m

aws-node-rjz8f 2/2 Running 0 3h44m

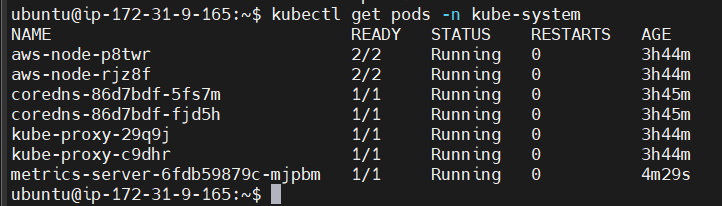
coredns-86d7bdf-5fs7m 1/1 Running 0 3h45m

coredns-86d7bdf-fjd5h 1/1 Running 0 3h45m

kube-proxy-29q9j 1/1 Running 0 3h44m

kube-proxy-c9dhr 1/1 Running 0 3h44m

metrics-server-6fdb59879c-mjpbm 1/1 Running 0 4m29s



metrics-server is running

If required:

kubectl patch rolebinding metrics-server-auth-reader -n kube-system --type=’merge’ -p ‘{

“metadata”: {

“labels”: {

“app.kubernetes.io/managed-by”: “Helm”

},

“annotations” : {

“meta.helm.sh/release-name”: “metrics-server”,

“meta.helm.sh/release-namespace”: “kube-system”

}

}

}’

ubuntu@ip-172-31-9-165:~$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-192-168-1-194.ca-central-1.compute.internal Ready <none> 4h v1.32.3-eks-473151a

ip-192-168-37-54.ca-central-1.compute.internal Ready <none> 4h v1.32.3-eks-473151a

ubuntu@ip-172-31-9-165:~$

ubuntu@ip-172-31-9-165:~$

ubuntu@ip-172-31-9-165:~$ kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

aws-node-p8twr 2/2 Running 0 4h

aws-node-rjz8f 2/2 Running 0 4h

coredns-86d7bdf-5fs7m 1/1 Running 0 4h2m

coredns-86d7bdf-fjd5h 1/1 Running 0 4h2m

kube-proxy-29q9j 1/1 Running 0 4h

kube-proxy-c9dhr 1/1 Running 0 4h

metrics-server-6fdb59879c-mjpbm 1/1 Running 0 21m

Deploy Grafana and Prometheus in K8s: --> Using Helm charts

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

ubuntu@ip-172-31-9-165:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

"prometheus-community" has been added to your repositories

ubuntu@ip-172-31-9-165:~$ helm repo update

Hang tight while we grab the latest from your chart repositories...

...Successfully got an update from the "metrics-server" chart repository

...Successfully got an update from the "prometheus-community" chart repository

Update Complete. ⎈Happy Helming!⎈

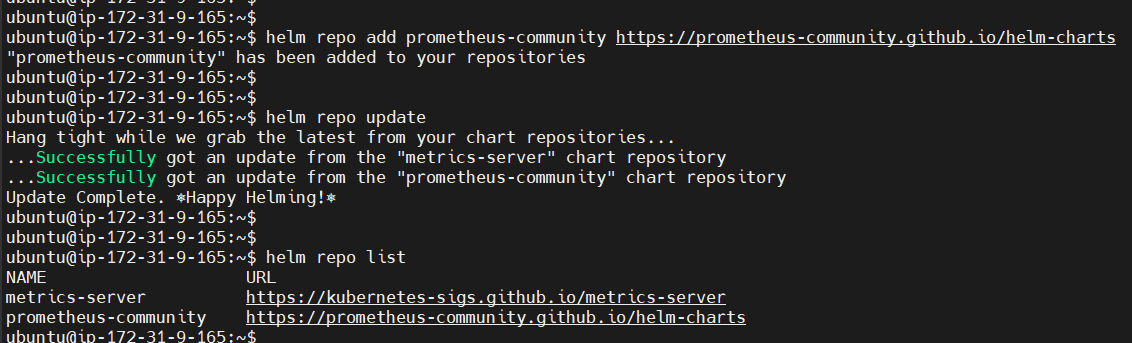
ubuntu@ip-172-31-9-165:~$ helm repo list

NAME URL

metrics-server https://kubernetes-sigs.github.io/metrics-server

prometheus-community https://prometheus-community.github.io/helm-charts

Both metrics-server and Prometheus are there



helm install prometheus prometheus-community/prometheus --namespace kube-system

ubuntu@ip-172-31-9-165:~$ helm install prometheus prometheus-community/prometheus --namespace kube-system

ubuntu@ip-172-31-9-165:~$ kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

aws-node-gzwnp 2/2 Running 0 80m

aws-node-p4qmb 2/2 Running 0 80m

coredns-86d7bdf-4mt5t 1/1 Running 0 81m

coredns-86d7bdf-5whrf 1/1 Running 0 81m

kube-proxy-79dk5 1/1 Running 0 80m

kube-proxy-jh7dr 1/1 Running 0 80m

metrics-server-d9fd456dc-ksgn2 1/1 Running 0 82m

metrics-server-d9fd456dc-n5fn9 1/1 Running 0 82m

prometheus-alertmanager-0 0/1 Pending 0 26s

prometheus-kube-state-metrics-5b9cfb448c-p8dzx 1/1 Running 0 26s

prometheus-prometheus-node-exporter-2f465 1/1 Running 0 26s

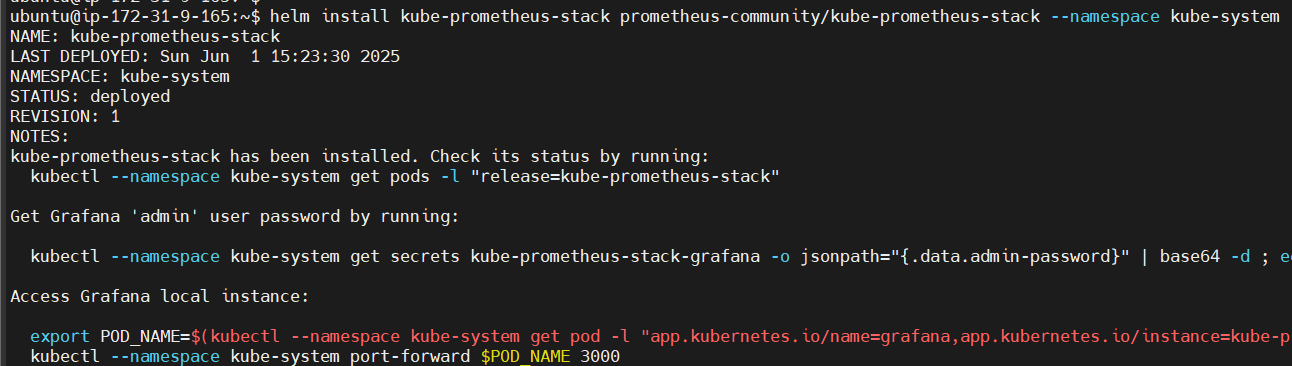
prometheus-prometheus-node-exporter-qj278 1/1 Running 0 26s

prometheus-prometheus-pushgateway-56d6b84f88-m2n88 1/1 Running 0 26s

prometheus-server-56d79479fc-ghbkz 0/2 Pending 0 26s

helm install kube-prometheus-stack prometheus-community/kube-prometheus-stack --namespace kube-system

ubuntu@ip-172-31-9-165:~$ helm install kube-prometheus-stack prometheus-community/kube-prometheus-stack --namespace kube-system



ubuntu@ip-172-31-9-165:~$ kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

alertmanager-kube-prometheus-stack-alertmanager-0 2/2 Running 0 3m5s

aws-node-gzwnp 2/2 Running 0 86m

aws-node-p4qmb 2/2 Running 0 86m

coredns-86d7bdf-4mt5t 1/1 Running 0 87m

coredns-86d7bdf-5whrf 1/1 Running 0 87m

kube-prometheus-stack-grafana-67bb9f4bc6-8d2dd 3/3 Running 0 3m9s

kube-prometheus-stack-kube-state-metrics-77678594d6-kxt7h 1/1 Running 0 3m9s

kube-prometheus-stack-operator-5d8cc87fbb-bfc7x 1/1 Running 0 3m9s

kube-prometheus-stack-prometheus-node-exporter-5x2sr 0/1 Pending 0 3m9s

kube-prometheus-stack-prometheus-node-exporter-kgh94 0/1 Pending 0 3m9s

kube-proxy-79dk5 1/1 Running 0 86m

kube-proxy-jh7dr 1/1 Running 0 86m

metrics-server-d9fd456dc-ksgn2 1/1 Running 0 87m

metrics-server-d9fd456dc-n5fn9 1/1 Running 0 87m

prometheus-alertmanager-0 0/1 Pending 0 6m22s

prometheus-kube-prometheus-stack-prometheus-0 2/2 Running 0 3m5s

prometheus-kube-state-metrics-5b9cfb448c-p8dzx 1/1 Running 0 6m22s

prometheus-prometheus-node-exporter-2f465 1/1 Running 0 6m22s

prometheus-prometheus-node-exporter-qj278 1/1 Running 0 6m22s

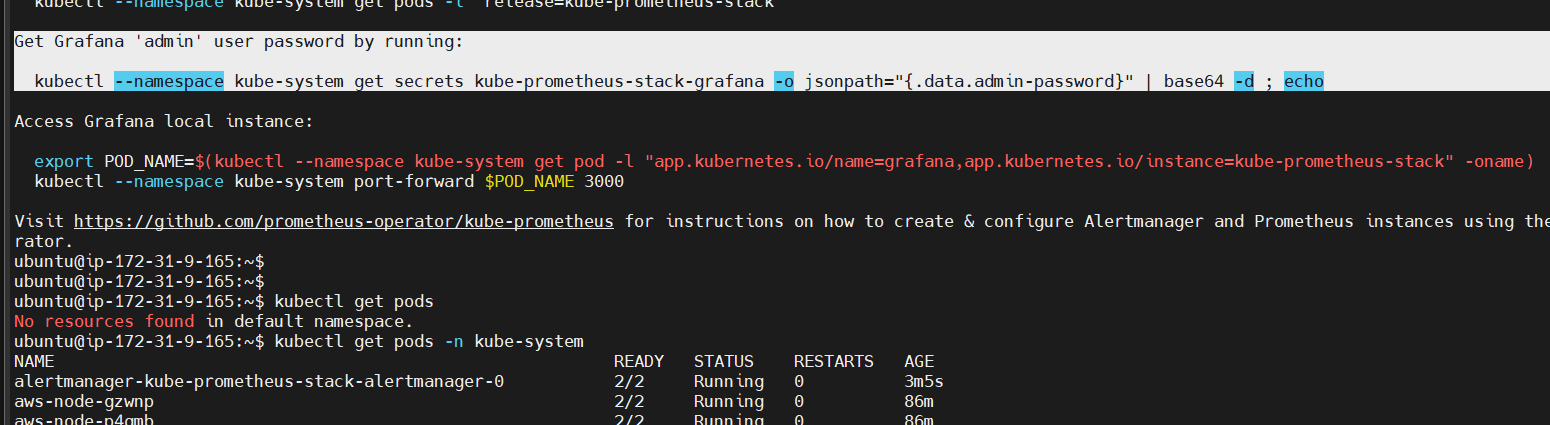
prometheus-prometheus-pushgateway-56d6b84f88-m2n88 1/1 Running 0 6m22s

prometheus-server-56d79479fc-ghbkz 0/2 Pending 0 6m22s

For Grafana, see this

Get Grafana 'admin' user password by running:

kubectl --namespace kube-system get secrets kube-prometheus-stack-grafana -o jsonpath="{.data.admin-password}" | base64 -d ; echo



$ helm install prometheus prometheus-community/kube-prometheus-stack --namespace default --> original command from Notes

ubuntu@ip-172-31-9-165:~$ kubectl get svc -n kube-system

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

alertmanager-operated ClusterIP None <none> 9093/TCP,9094/TCP,9094/UDP 8m30s

eks-extension-metrics-api ClusterIP 10.100.156.120 <none> 443/TCP 97m

kube-dns ClusterIP 10.100.0.10 <none> 53/UDP,53/TCP,9153/TCP 93m

kube-prometheus-stack-alertmanager ClusterIP 10.100.112.139 <none> 9093/TCP,8080/TCP 8m34s

kube-prometheus-stack-coredns ClusterIP None <none> 9153/TCP 8m34s

kube-prometheus-stack-grafana ClusterIP 10.100.141.227 <none> 80/TCP 8m34s

kube-prometheus-stack-kube-controller-manager ClusterIP None <none> 10257/TCP 8m34s

kube-prometheus-stack-kube-etcd ClusterIP None <none> 2381/TCP 8m34s

kube-prometheus-stack-kube-proxy ClusterIP None <none> 10249/TCP 8m34s

kube-prometheus-stack-kube-scheduler ClusterIP None <none> 10259/TCP 8m34s

kube-prometheus-stack-kube-state-metrics ClusterIP 10.100.201.64 <none> 8080/TCP 8m34s

kube-prometheus-stack-kubelet ClusterIP None <none> 10250/TCP,10255/TCP,4194/TCP 8m30s

kube-prometheus-stack-operator ClusterIP 10.100.148.166 <none> 443/TCP 8m34s

kube-prometheus-stack-prometheus ClusterIP 10.100.176.99 <none> 9090/TCP,8080/TCP 8m34s

kube-prometheus-stack-prometheus-node-exporter ClusterIP 10.100.16.211 <none> 9100/TCP 8m34s

metrics-server ClusterIP 10.100.138.211 <none> 443/TCP 93m

prometheus-alertmanager ClusterIP 10.100.134.106 <none> 9093/TCP 11m

prometheus-alertmanager-headless ClusterIP None <none> 9093/TCP 11m

prometheus-kube-state-metrics ClusterIP 10.100.22.2 <none> 8080/TCP 11m

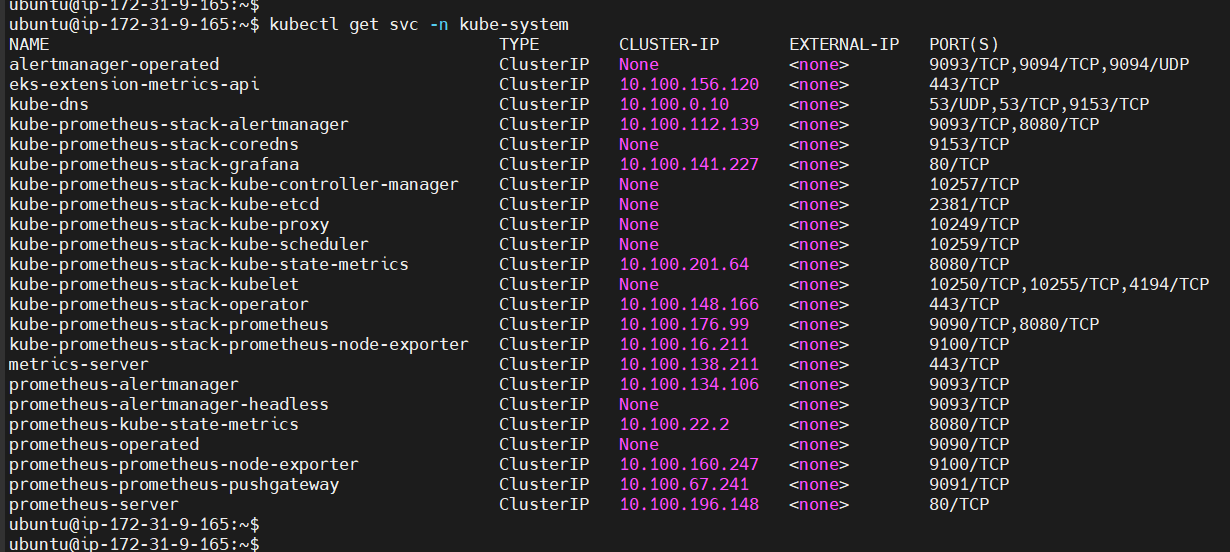
prometheus-operated ClusterIP None <none> 9090/TCP 8m30s

prometheus-prometheus-node-exporter ClusterIP 10.100.160.247 <none> 9100/TCP 11m

prometheus-prometheus-pushgateway ClusterIP 10.100.67.241 <none> 9091/TCP 11m

prometheus-server ClusterIP 10.100.196.148 <none> 80/TCP 11m

Prometheus, Grafana ---> They are all using ClusterIP service, if you using ClusterIP service, we can access them within the Cluster

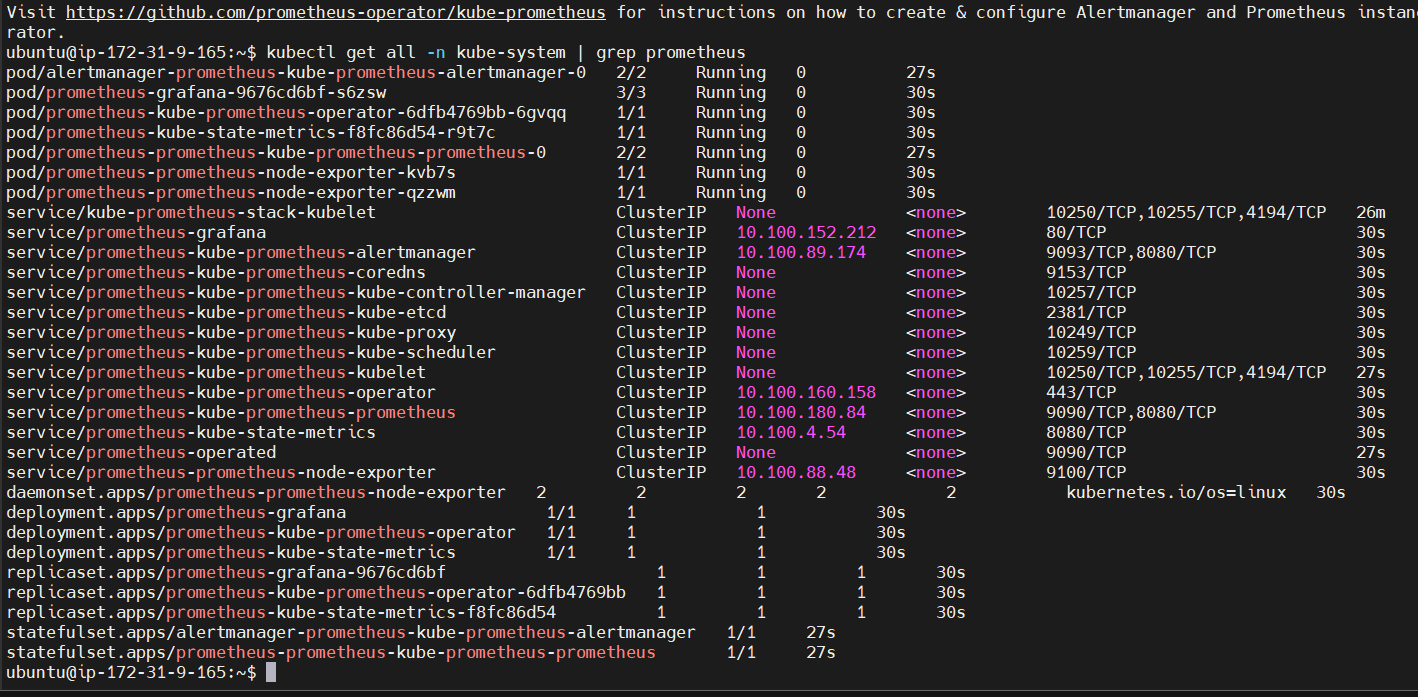


Expose these services with LoadBalancer: By default, Prometheus and Grafana services are ClusterIP (within Cluster we can access)

To access them externally, change their Type to LoadBalancer

ubuntu@ip-172-31-9-165:~$ helm uninstall kube-prometheus-stack --namespace kube-system

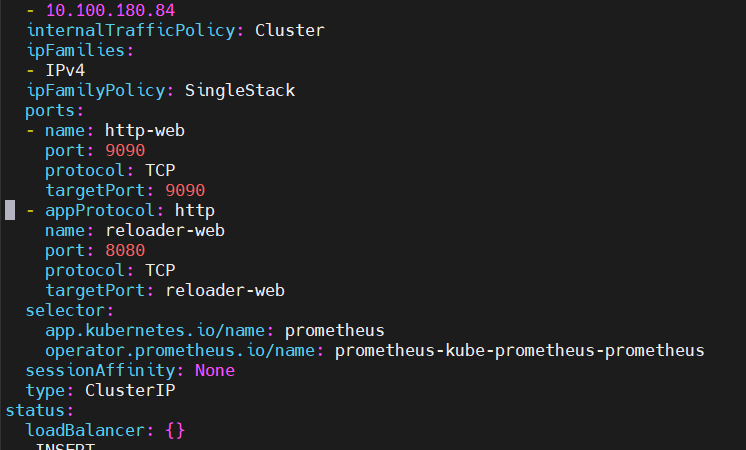
ubuntu@ip-172-31-9-165:~$ helm install prometheus prometheus-community/kube-prometheus-stack --namespace kube-system



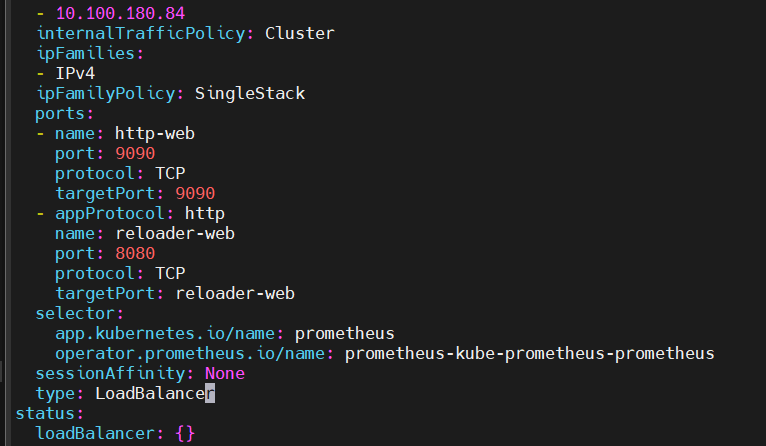
Now we edit the svc

ubuntu@ip-172-31-9-165:~$ kubectl edit svc -n kube-system prometheus-kube-prometheus-prometheus

In this file we can see type: ClusterIP



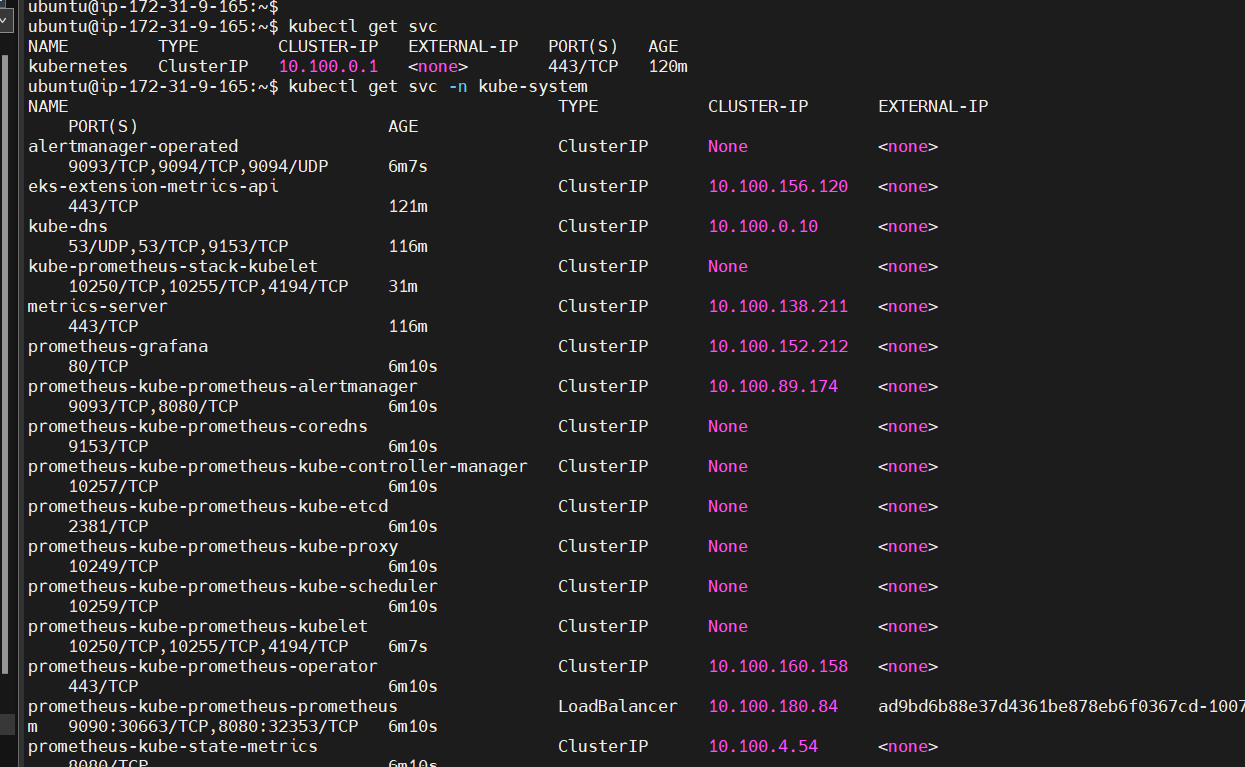
Change ClusterIP to LoadBalancer



ubuntu@ip-172-31-9-165:~$ kubectl edit svc -n kube-system prometheus-kube-prometheus-prometheus

service/prometheus-kube-prometheus-prometheus edited

ubuntu@ip-172-31-9-165:~$ kubectl get svc



Changed to LoadBalancer

Spec:

---

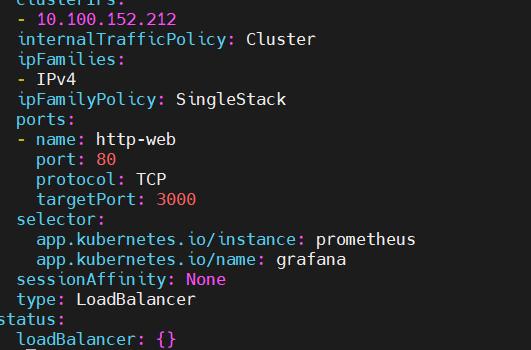
---

type: LoadBalancer

Next edit the Prometheus Grafana file

ubuntu@ip-172-31-9-165:~$ kubectl edit svc kube-system prometheus-grafana

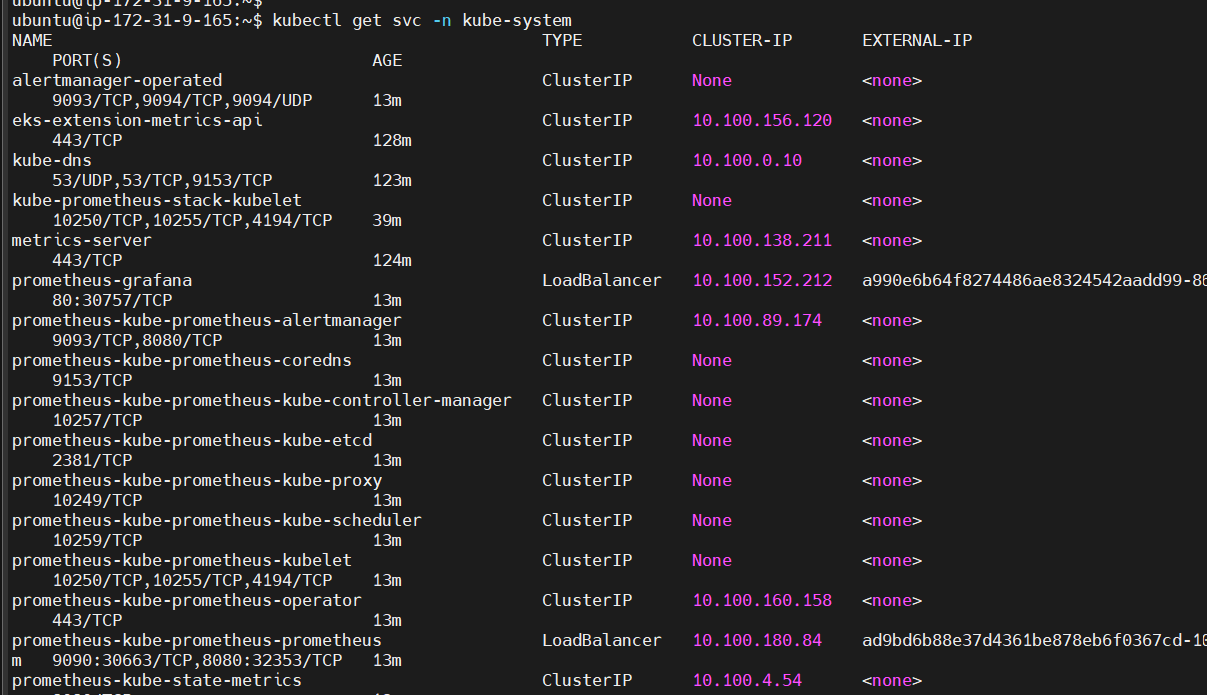
ubuntu@ip-172-31-9-165:~$ kubectl edit svc -n kube-system prometheus-grafana



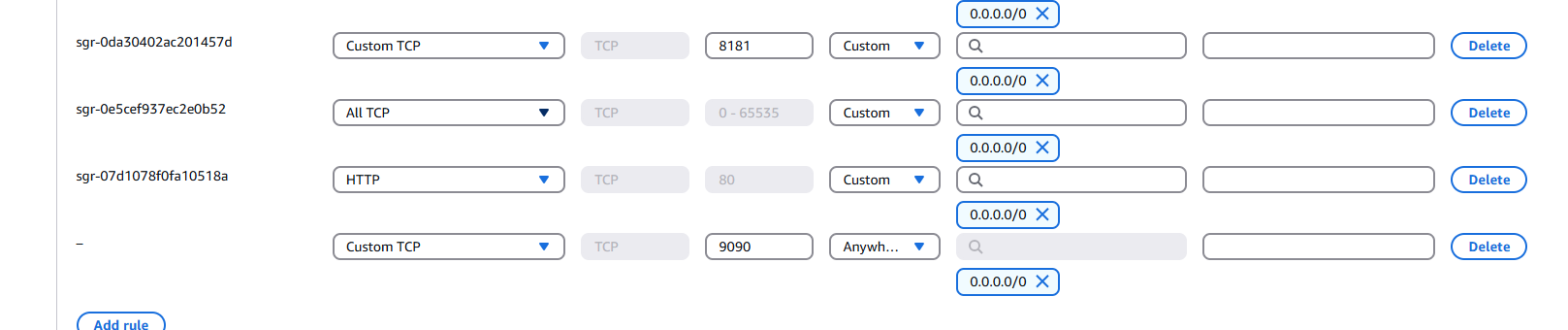
ubuntu@ip-172-31-9-165:~$ kubectl edit svc -n kube-system prometheus-grafana

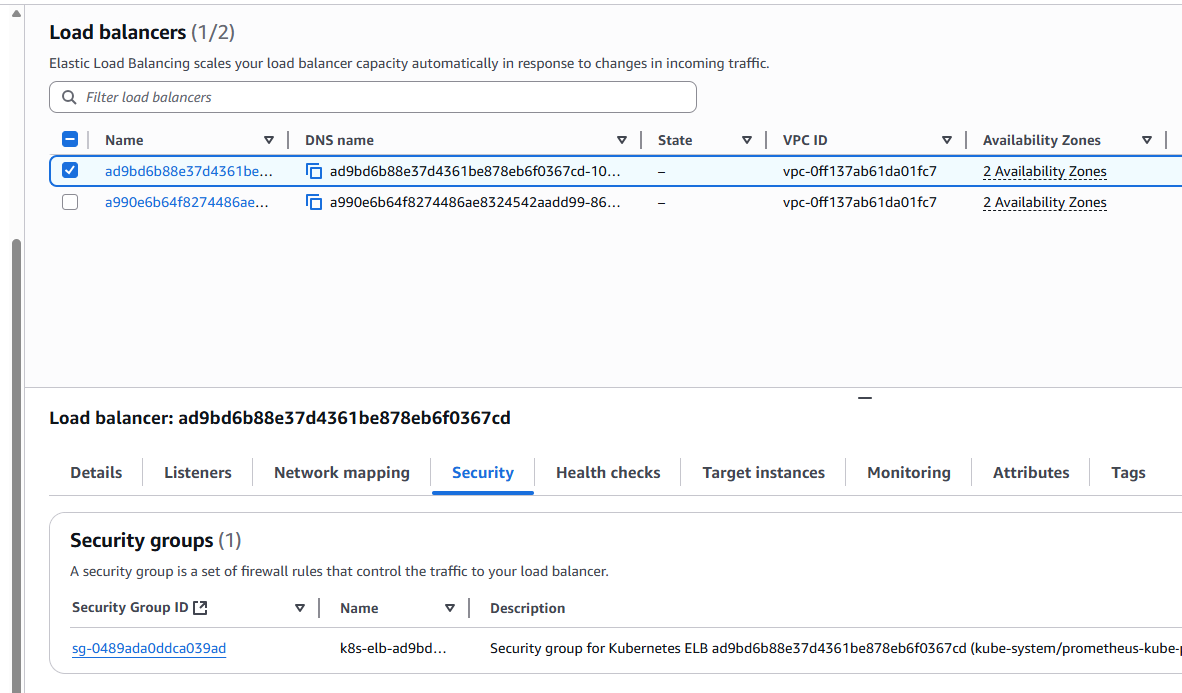
service/prometheus-grafana edited

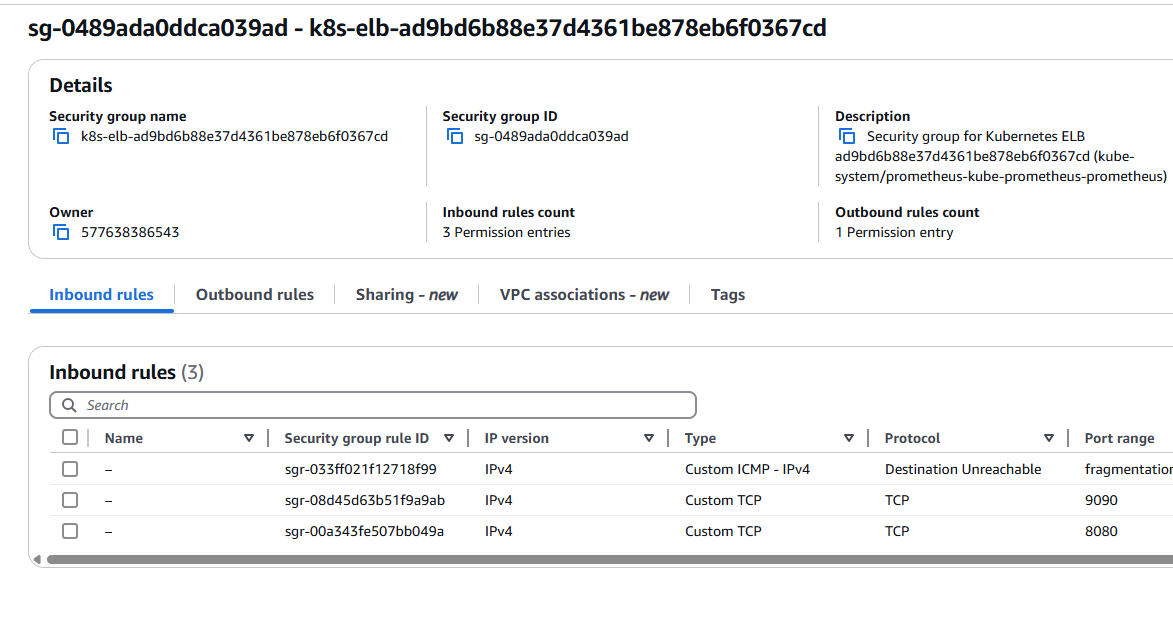
ubuntu@ip-172-31-9-165:~$ kubectl get svc -n kube-system



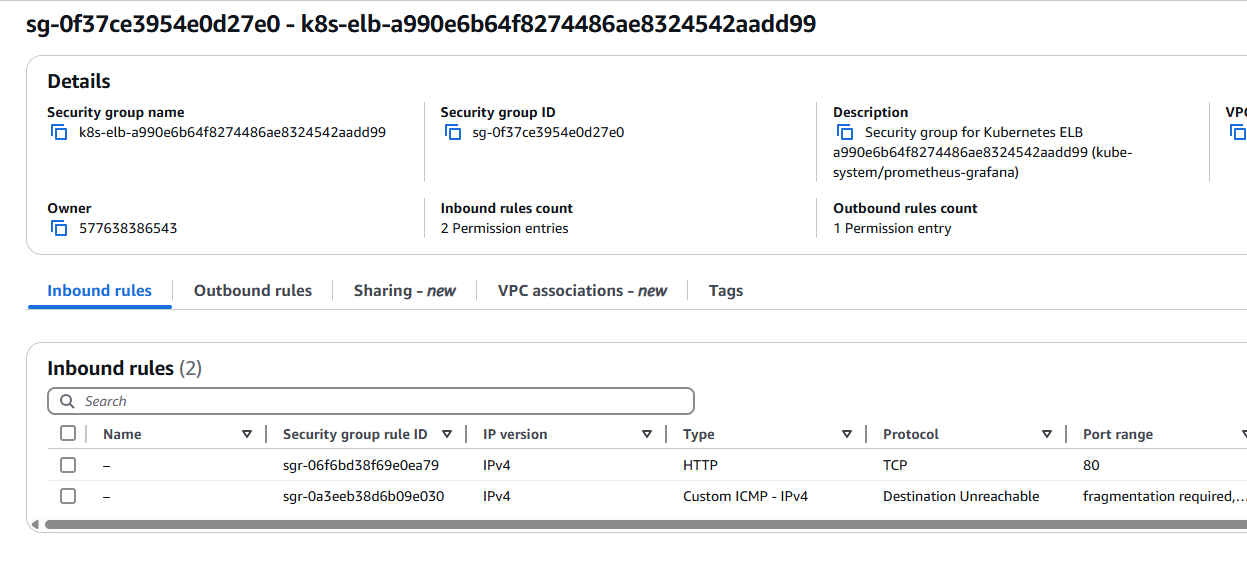
LoadBalancer ports 80 and 9090 should be enabled in Security groups



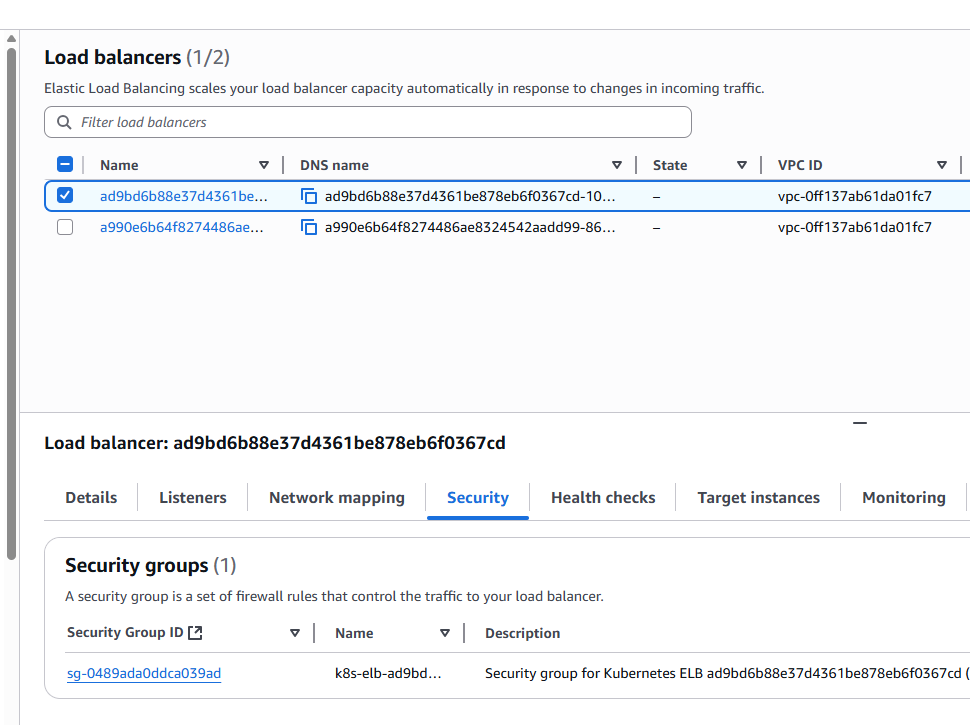




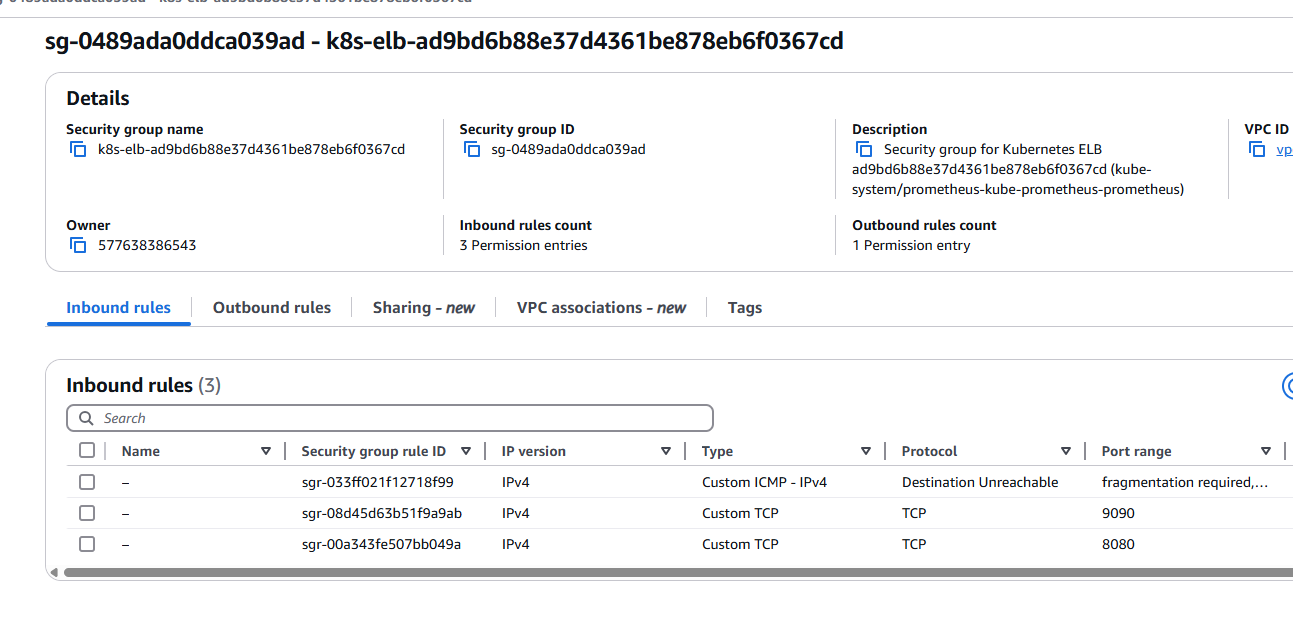
For the second LoadBalancer



Go back to Loadbalancer



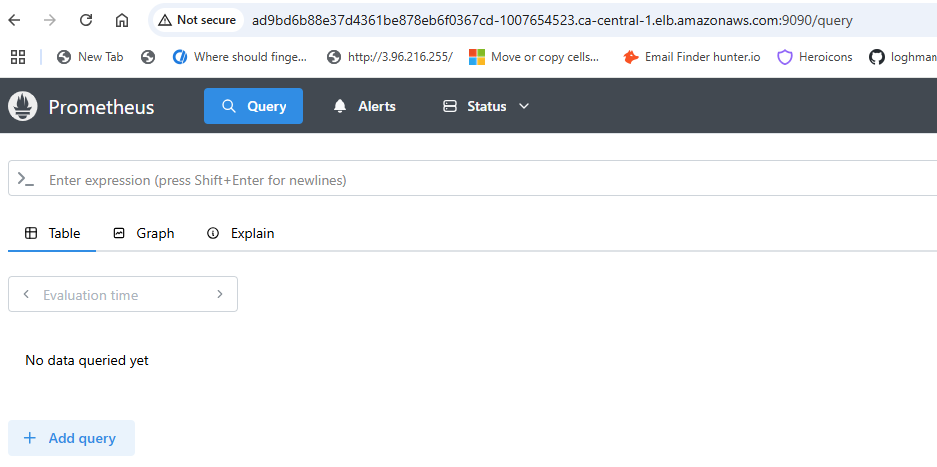
Go to its Security rules



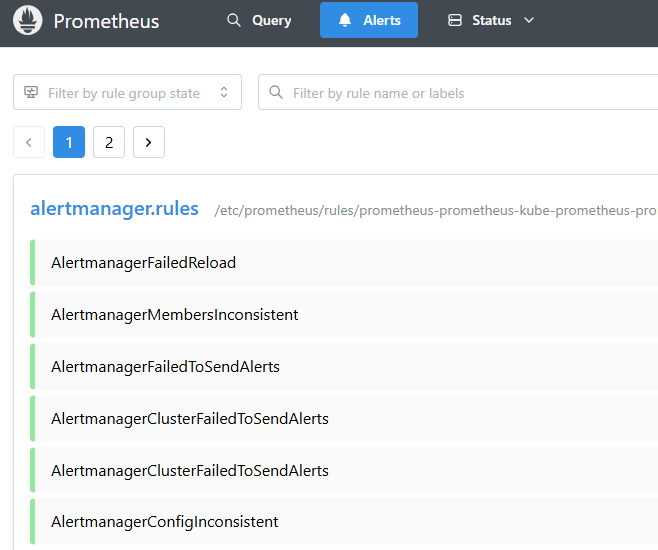
Port 9090 is already added in Inbound rules that means Prometheus is running on this LoadBalancer

Copy DNS add 9090 to the end

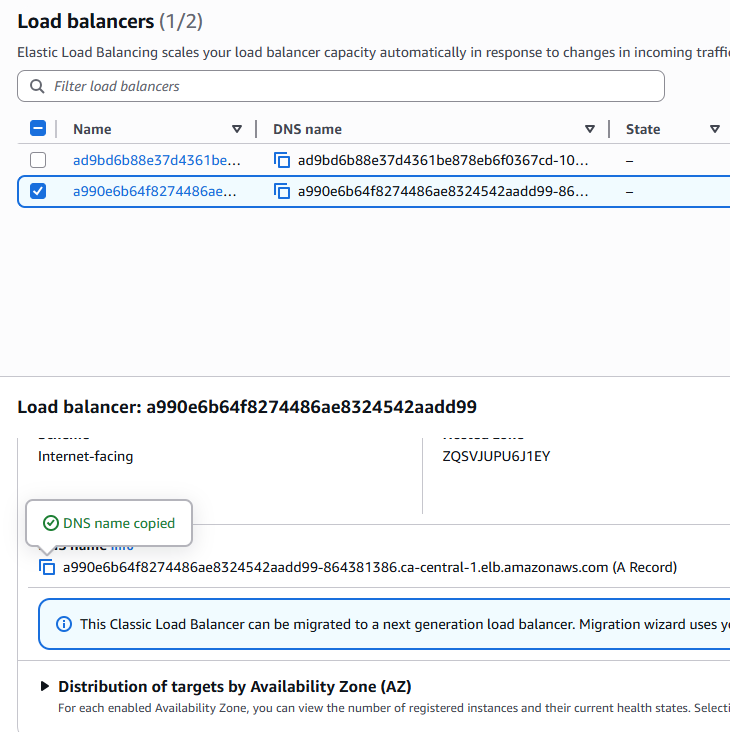
<http://ad9bd6b88e37d4361be878eb6f0367cd-1007654523.ca-central-1.elb.amazonaws.com:9090/query>



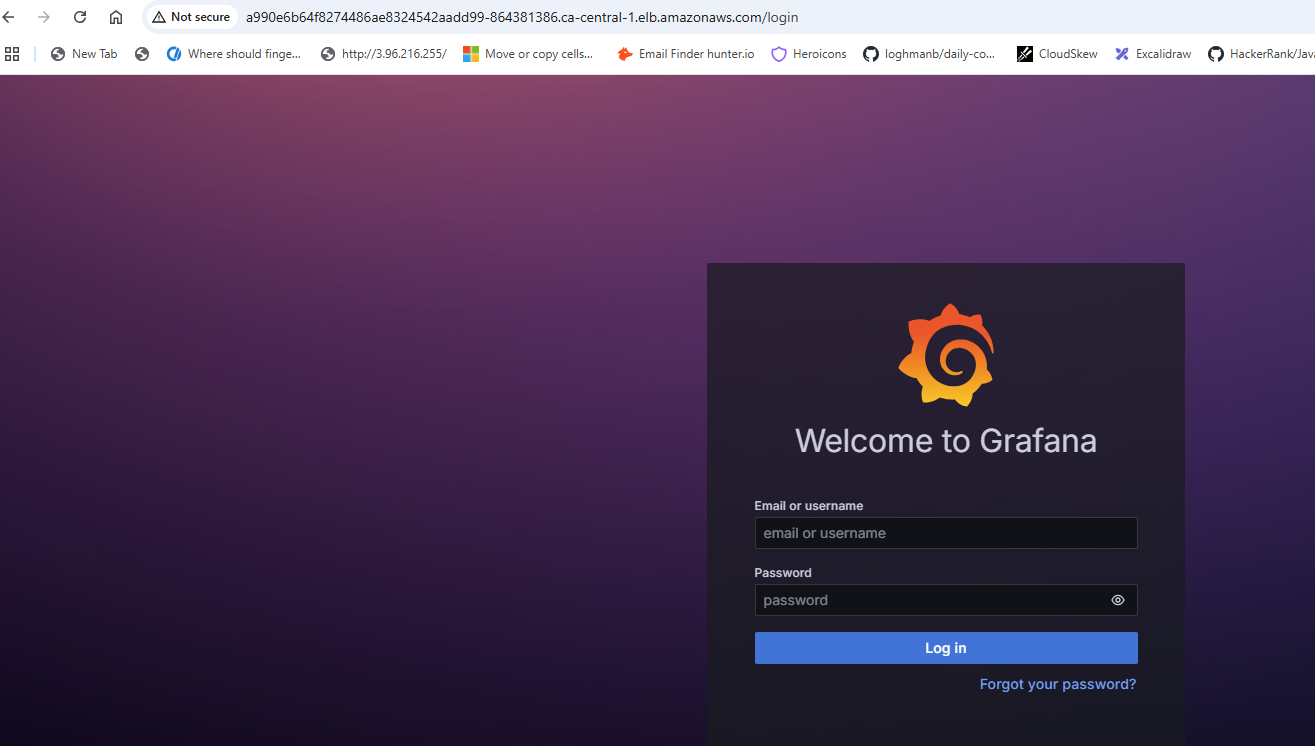
We can see Prometheus up and running

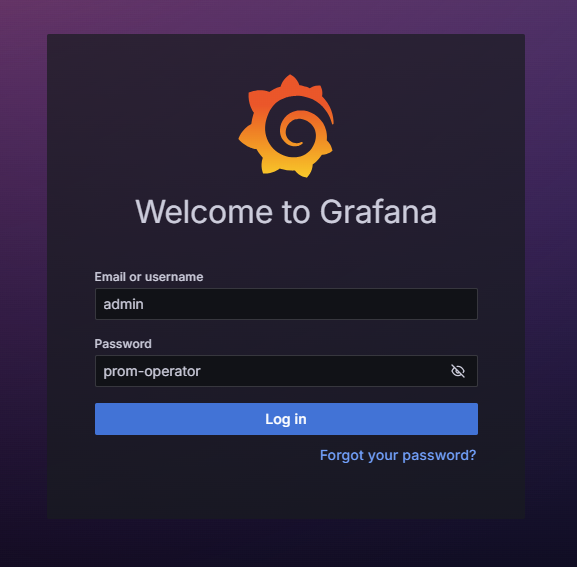


Go to the second LoadBalancer and copy DNS

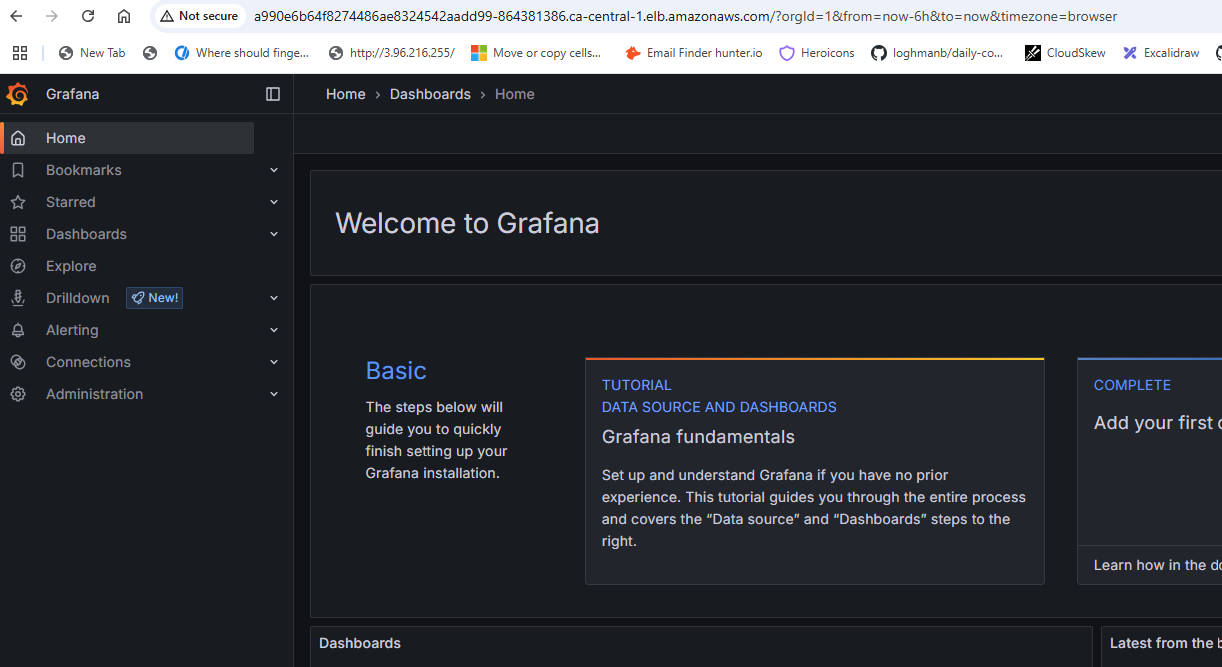


<http://a990e6b64f8274486ae8324542aadd99-864381386.ca-central-1.elb.amazonaws.com/login>

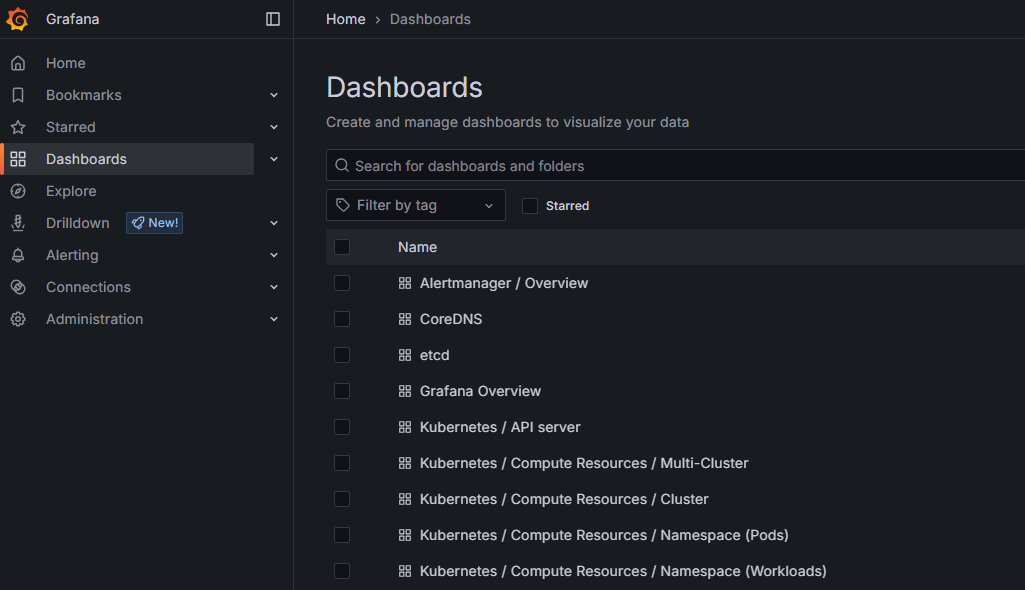




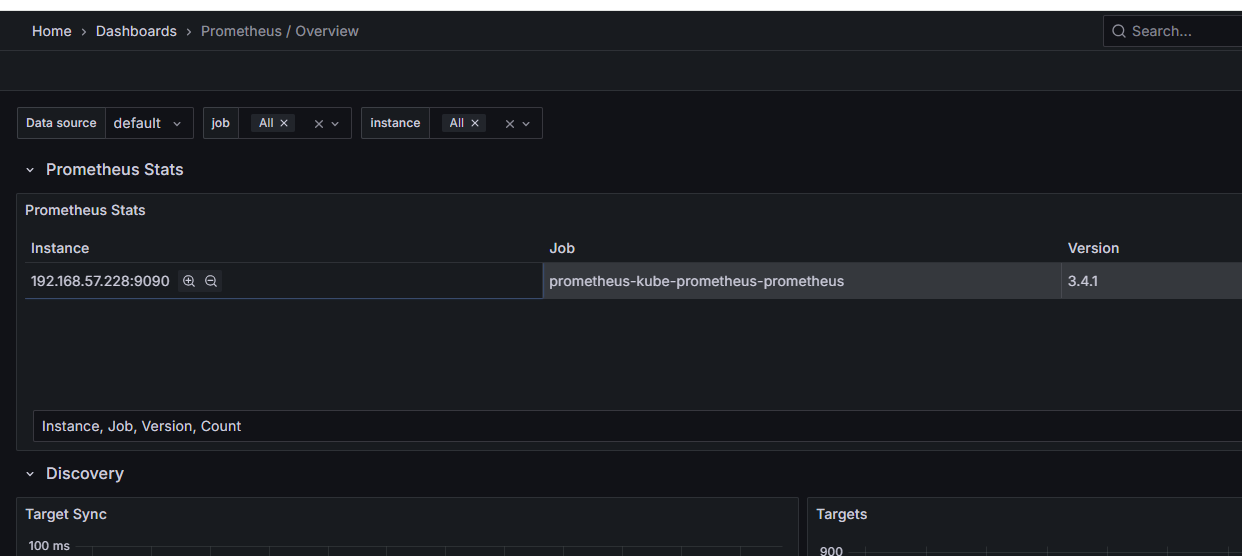
It Logs-in

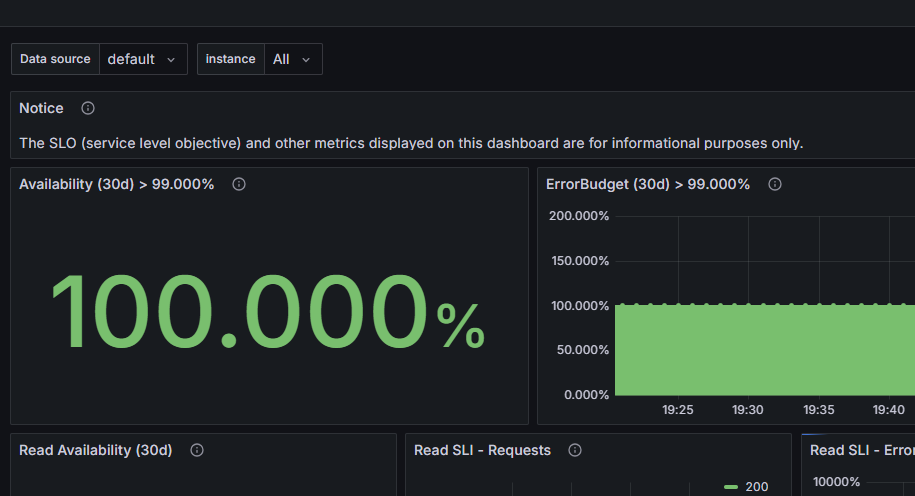


Go to Dashboard



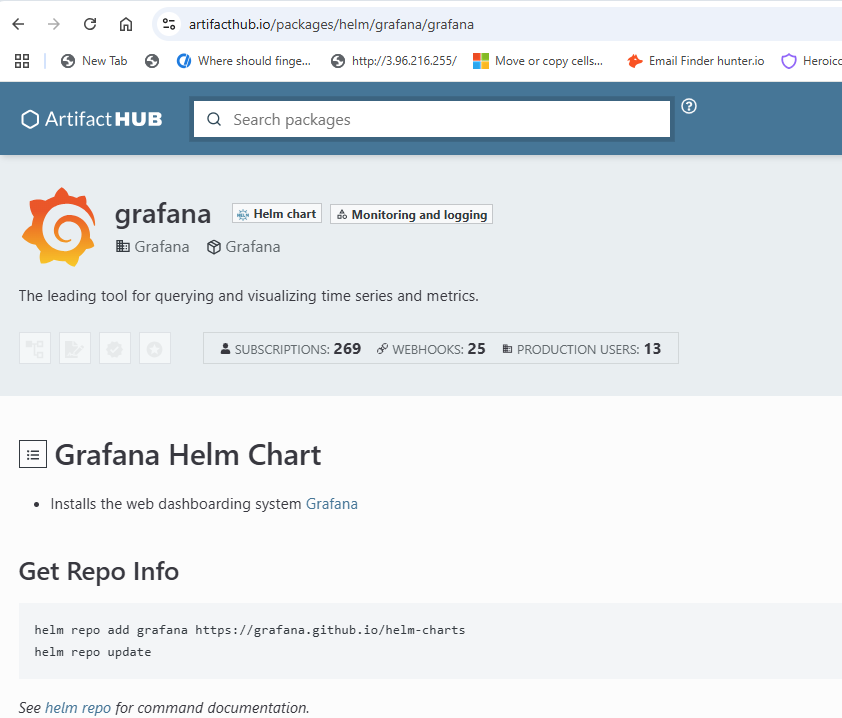
Prometheus overview





<https://artifacthub.io/packages/helm/grafana/grafana>

This page gives the charts



For Prometheus: https://artifacthub.io/packages/helm/prometheus-community/prometheus

Helm makes installation of K8s resources into K8s cluster easy

Access Prometheus Server: LBR\_DNS:9090/

Access Grafana Server: LBR\_DNS/

Use the below credentials to login into Grafana server

Username: admin, Password: prom-operator

After logging into Grafana, we can monitor our K8s cluster

EFK, DaemonSet, StatefulSet, ConfigMap and Secrets

PV & PVC

Ingress controller

Readiness and Liveness probe

<https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>

A DaemonSet ensures that all (or some) Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.

DaemonSet will make sure that if you have 10 worker nodes, your pods will be running in all the 10 worker nodes

WorkerNode 1

Control plane

kubectl

WorkerNode 2

EKS host VM

WorkerNode 3

If there is a Pod, the copy has to be maintained in all the Worker nodes. If you want to make sure Pod copy is there in all the Worker nodes then DaemonSet comes into picture

When you remove the WorkerNode only then those pods will be deleted

Some typical uses of a DaemonSet are:

* running a cluster storage daemon on every node
* running a logs collection daemon on every node
* running a node monitoring daemon on every node

Dev environment

SIT

UAT environment

Production (Live environment)

Pilot

JAR/WAR file

Docker image

Source code

Database for Production

One Docker image is created and it is run on multiple environments

If different environments are using different databases, don’t you think passwords and usernames will also be different? If an application is packaged with configuration details with the same database credentials that cannot be changed now. that’s where we need the concept that allows you to add the credentials dynamically. Config map and secrets come into picture.

Kubernetes cluster with Pod

JAR/WAR file

Docker image

Source code

Secrets

Config map

We are going to dynamically pass Secrets from Config map and Secrets

Config map will have data that’s not sensitive

Secrets will have data that’s sensitive

We are not going to hard-code the DB credentials.

Each environment usually has different configuration settings

Database properties

SMTP (email) settings

Kafka configurations

Redis settings

And more…

In this case, if we hard-code the properties it is a problem

Best practice:

Make your application loosely coupled, so it can run in any environment with minimal changes

Use Kubernetes Config Maps and Secrets to externalize environment-specific values like database credentials, URLs, and keys

ConfigMaps and Secrets allow us to separate application configuration from Docker image

This makes our application portable and environment-independent, meaning:

We can deploy the same Docker image into different environments (Dev, SIT, UAT, etc) without modifying the image itself

What’s a ConfigMap?

Used to store non-sensitive configuration as key-value pairs

Example: API URLs, File paths, Environment names

What’s a Secret?

Used to store sensitive data like passwords, tokens, API keys also in key-value format

Data is stored in base64-encoded form

With ConfigMap and Secrets, we can use the same Docker image into multiple environments

Environment is the basic-setup to run application

Hard-coded configuration (Not recommended)

spring:

datasource:

driver-class-name: com.mysql.cj.jdbc.Driver

url: jdbc:mysql://mysqldb:3306/finisher

username: root

password: root123

jpa:

hibernate:

ddl-auto: update

show-sql: true

What’s the problem?

Tied to a single environment (eg: development)

Passwords and sensitive info are exposed

You need to rebuild the Docker image to change configs

Environment-based configuration (Recommended)

spring:

datasource:

driver-class-name: ${DB\_DRIVER:com.mysql.cj.jdbc.Driver}

url: ${DB\_URL:jdbc:mysql://mysqldb:3306/finisher}

username: ${DB\_USERNAME:root}

password: ${DB\_PASSWORD:root123}

jpa:

hibernate:

ddl-auto: update

show-sql: true

Delete cluster

eksctl delete cluster --name my-eks-cluster --region ca-central-1