AWS EKS Implementation Guide Part -3

Prometheus & Grafana

01.01.2020

Singaravelan Palani

Minfy Technologies Pvt Ltd.

Table of Contents

Overview: Helm

What is Helm ?

Helm is a tool for managing Charts. Charts are packages of pre-configured Kubernetes resources.

Use Helm to:

- Find and use popular software packaged as Helm Charts to run in Kubernetes
- Share your own applications as Helm Charts
- Create reproducible builds of your Kubernetes applications
- Intelligently manage your Kubernetes manifest files
- Manage releases of Helm packages

Helm is a tool that streamlines installing and managing Kubernetes applications. Think of it like apt/yum/homebrew for Kubernetes.

- Helm renders your templates and communicates with the Kubernetes API
- Helm runs on your laptop, CI/CD, or wherever you want it to run.
- Charts are Helm packages that contain at least two things:
    - A description of the package (Chart.yaml)
    - One or more templates, which contain Kubernetes manifest files
- Charts can be stored on disk, or fetched from remote chart repositories (like Debian or RedHat packages)

Helm Installation in Kops Server

- Install the helm using following commands

```
$ curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get > install-helm.sh
$ chmod +x install-helm.sh
$ ./install-helm.sh
$ helm init
$ helm repo update
```

```
[ec2-user@ip-12-0-14-210 ~]$ curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get > install-helm.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  7164  100  7164    0     0  25225      0 --:--:-- --:--:-- --:--:-- 25136
[ec2-user@ip-12-0-14-210 ~]$ chmod +x install-helm.sh
[ec2-user@ip-12-0-14-210 ~]$ ./install-helm.sh
Helm v2.16.1 is already latest
Run 'helm init' to configure helm.
[ec2-user@ip-12-0-14-210 ~]$
```

```
[ec2-user@ip-12-0-14-210 ~]$ helm init
Creating /home/ec2-user/.helm
Creating /home/ec2-user/.helm/repository
Creating /home/ec2-user/.helm/repository/cache
Creating /home/ec2-user/.helm/repository/local
Creating /home/ec2-user/.helm/plugins
Creating /home/ec2-user/.helm/starters
Creating /home/ec2-user/.helm/cache/archive
Creating /home/ec2-user/.helm/repository/repositories.yaml
Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /home/ec2-user/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users' policy.
To prevent this, run `helm init` with the --tiller-tls-verify flag.
For more information on securing your installation see: https://docs.helm.sh/using_helm/#securing-your-helm-installation
[ec2-user@ip-12-0-14-210 ~]$
```

- Helm version check

```
[ec2-user@ip-12-0-14-210 ~]$ helm  version
Client: &version.Version{SemVer:"v2.16.1", GitCommit:"bbdfe5e7803a12bbdf97e94cd847859890cf4050", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.16.1", GitCommit:"bbdfe5e7803a12bbdf97e94cd847859890cf4050", GitTreeState:"clean"}
[ec2-user@ip-12-0-14-210 ~]$
```

Sample MySQL Application Deploy using Helm Kubernetes

- MySQL Application Deploy using Helm Kubernetes following commands

```
$ helm install stable/mysql --name my-mysql-install --set mysqlPassword=cloudredhat
```

```
[ec2-user@ip-12-0-14-210 ~]$
[ec2-user@ip-12-0-14-210 ~]$ helm install stable/mysql --name my-mysql-install --set mysqlPassword=cloudredhat
Error: release my-mysql-install failed: namespaces "default" is forbidden: User "system:serviceaccount:kube-system:default" cannot get res
ource "namespaces" in API group "" in the namespace "default"
[ec2-user@ip-12-0-14-210 ~]$
```

- Here we got the error, because of we don't have a privilege access –
- We have to get the privilege access use following commands

```
$ kubectl create serviceaccount --namespace kube-system tiller
$ kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
$ kubectl patch deploy --namespace kube-system tiller-deploy -p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}'
```

```
[ec2-user@ip-12-0-14-210 ~]$ kubectl create serviceaccount --namespace kube-system tiller
serviceaccount/tiller created
[ec2-user@ip-12-0-14-210 ~]$ kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-syste
m:tiller
clusterrolebinding.rbac.authorization.k8s.io/tiller-cluster-rule created
[ec2-user@ip-12-0-14-210 ~]$ kubectl patch deploy --namespace kube-system tiller-deploy -p '{"spec":{"template":{"spec":{"serviceAccount":
"tiller"}}}}'
deployment.extensions/tiller-deploy patched
[ec2-user@ip-12-0-14-210 ~]$ helm ls
Error: configmaps is forbidden: User "system:serviceaccount:kube-system:default" cannot list resource "configmaps" in API group "" in the
namespace "kube-system"
```

- Now try to install MySQL using helm

```
[ec2-user@ip-12-0-14-210 ~]$ helm install stable/mysql --name my-mysql-install --set mysqlPassword=cloudredhat
NAME:   my-mysql-install
LAST DEPLOYED: Fri Jan  3 05:11:42 2020
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/ConfigMap
NAME                   AGE
my-mysql-install-test  0s

==> v1/Deployment
NAME              AGE
my-mysql-install  0s

==> v1/PersistentVolumeClaim
NAME              AGE
my-mysql-install  0s

==> v1/Pod(related)
NAME                                AGE
my-mysql-install-6f896cfb8f-n7ssv  0s

==> v1/Secret
NAME              AGE
my-mysql-install  0s

==> v1/Service
NAME              AGE
```

```
To get your root password run:

    MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace default my-mysql-install -o jsonpath="{.data.mysql-root-password}" | base64 --dec
ode; echo)

To connect to your database:

1. Run an Ubuntu pod that you can use as a client:

    kubectl run -i --tty ubuntu --image=ubuntu:16.04 --restart=Never -- bash -il

2. Install the mysql client:

    $ apt-get update && apt-get install mysql-client -y

3. Connect using the mysql cli, then provide your password:
    $ mysql -h my-mysql-install -p

To connect to your database directly from outside the K8s cluster:
    MYSQL_HOST=127.0.0.1
    MYSQL_PORT=3306

    # Execute the following command to route the connection:
    kubectl port-forward svc/my-mysql-install 3306

    mysql -h ${MYSQL_HOST} -P${MYSQL_PORT} -u root -p${MYSQL_ROOT_PASSWORD}


[ec2-user@ip-12-0-14-210 ~]$
```

MySQL Application Deploy using Helm Kubernetes

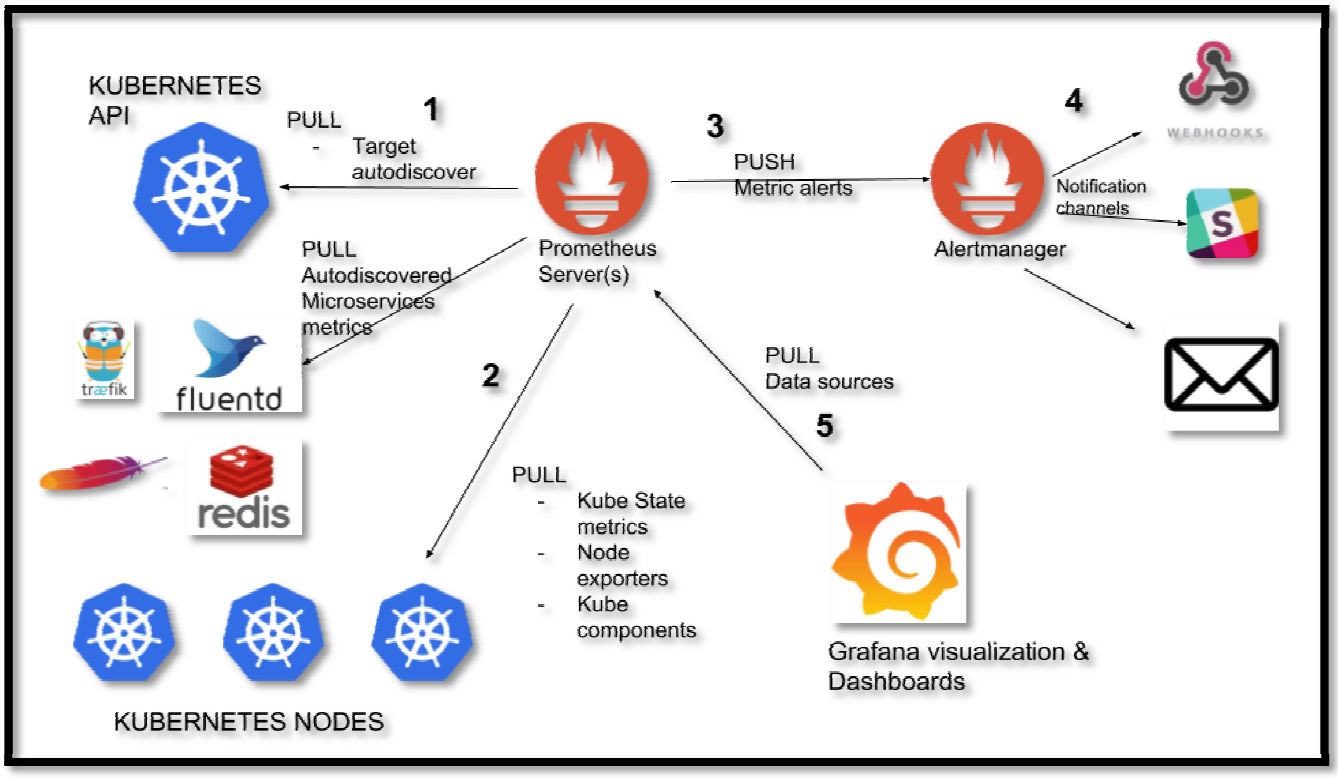
```
[ec2-user@ip-12-0-14-210 ~]$ helm ls
NAME                    REVISION        UPDATED                         STATUS          CHART                                   A
PP VERSION      NAMESPACE
my-mysql-install        1               Fri Jan  3 05:11:42 2020        DEPLOYED        mysql-1.6.2                             5
.7.28           default
[ec2-user@ip-12-0-14-210 ~]$
```

Kubernetes-Monitoring-System

- Kube-state-metrics
- Node exporter
- Alert Manager
- Prometheus Operator

We will discuss in following diagram, this documents not available for Alart manger in real time operations.

```
NAME                                              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
service/alertmanager-operated                     ClusterIP   None            <none>        9093/TCP,9094/TCP,9094/UDP   5m6s
service/monitoring-grafana                        ClusterIP   100.71.249.233  <none>        80/TCP                       5m26s
service/monitoring-kube-state-metrics             ClusterIP   100.67.210.87   <none>        8080/TCP                     5m26s
service/monitoring-prometheus-node-exporter       ClusterIP   100.64.122.38   <none>        9100/TCP                     5m26s
service/monitoring-prometheus-oper-alertmanager   ClusterIP   100.68.254.29   <none>        9093/TCP                     5m26s
service/monitoring-prometheus-oper-operator       ClusterIP   100.68.63.27    <none>        8080/TCP,443/TCP             5m26s
service/monitoring-prometheus-oper-prometheus     ClusterIP   100.67.38.254   <none>        9090/TCP                     5m26s
service/prometheus-operated                       ClusterIP   None            <none>        9090/TCP                     4m56s
```



Kube-state-metrics

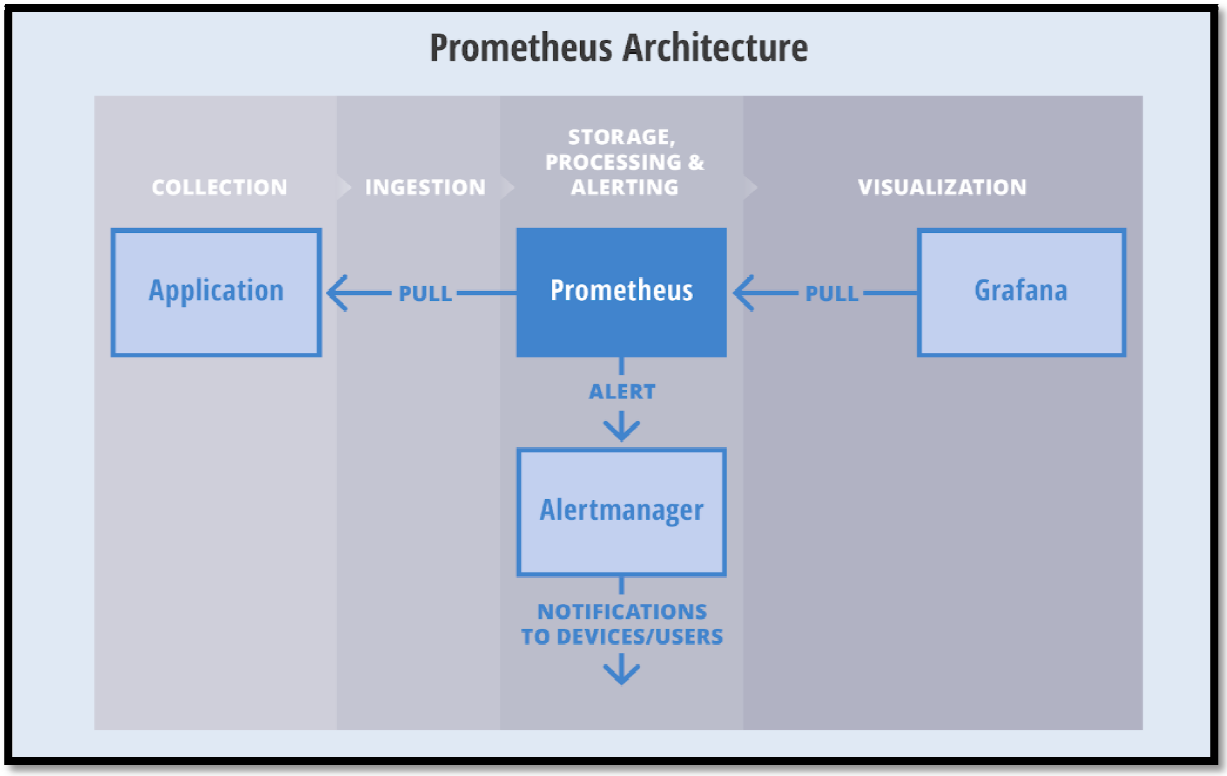
- kube-state-metrics is a simple service that listens to the Kubernetes API server and generates metrics about the state of the objects. (See examples in the Metrics section below.) It is not focused on the health of the individual Kubernetes components, but rather on the health of the various objects inside, such as deployments, nodes and pods.
- The metrics are exported through the Prometheus golang client on the HTTP endpoint /metrics on the listening port (default 80). They are served either as plaintext or protobuf depending on the Accept header. They are designed to be consumed either by Prometheus itself or by a scraper that is compatible with scraping a Prometheus client endpoint. You can also open /metrics in a browser to see the raw metrics.
- Ref Link: https://github.com/kubernetes/kube-state-metrics

Node exporter

- Node exporter is the best way to collect all the Linux server related metrics and statistics for monitoring.

Alert Manager

- Alert Manager is used to handle alerts for client applications (like Prometheus). It also takes care of alerts deduplicating, grouping and then routes them to different receivers such as E-mail, Slack, and Pager Duty.
- Alert Manager can be configured via command-line flags and configuration file. While command line flags configure system parameters for Alert Manager, the configuration file defines inhibition rules, notification routing, and notification receivers.
- Here is a basic architecture of Alert manager with Prometheus.



- This is how Prometheus architecture works:-

- ✓ If you see in the above picture Prometheus is scraping the metrics from its client application (exporters).
- ✓ When the alert is generated then it pushes it to the Alert Manager, later Alert Manager validates the alerts groups on the basis of labels.
- ✓ and then forward it to the receivers like Email or Slack.

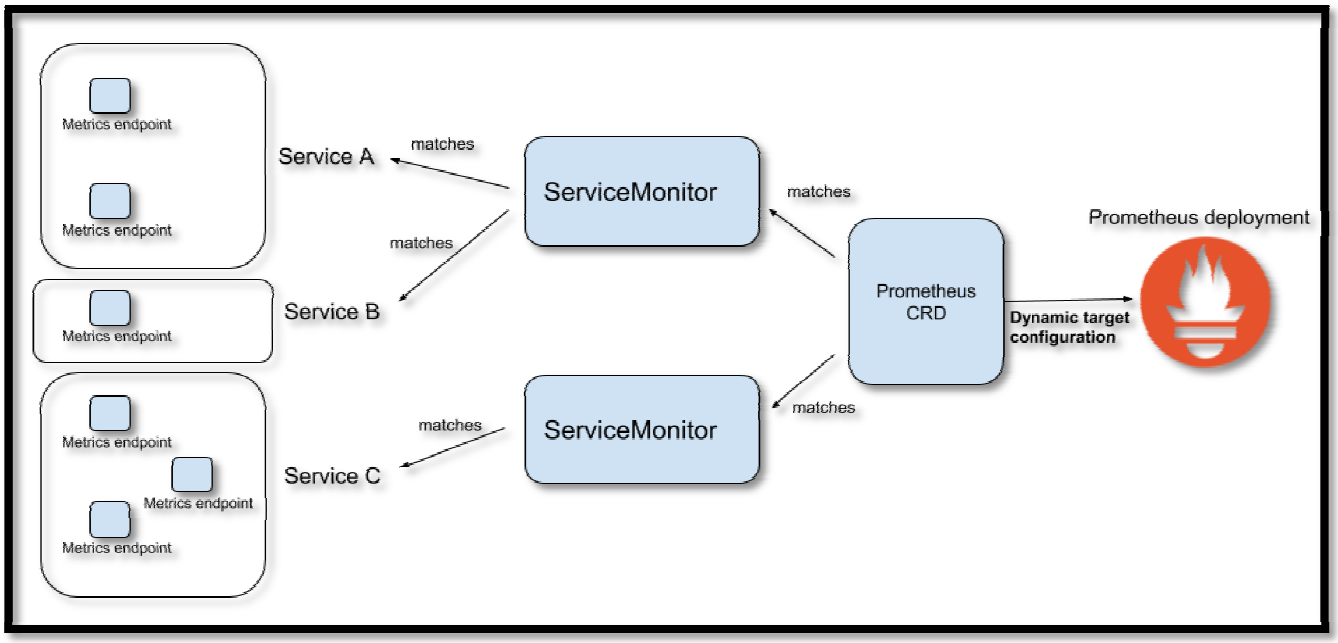
What is Prometheus?

- Prometheus is an open-source system monitoring and alerting toolkit originally built at SoundCloud.
- Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community.
- It is now a standalone open source project and maintained independently of any company.
- Prometheus has become the standard tool for monitoring and alerting in Kubernetes and Docker world. It provides by far the most detailed and actionable metrics and analysis. In the latest major release of 2.0 versions, the performance of Prometheus improved significantly and now Prometheus performs well under heavy loads and bursts. In addition to that, you get all of the benefits of a world-leading open source project. Prometheus is free at the point of use and covers many use cases with ease.

Prometheus Operator

- In late 2016, CoreOS introduced the Operator pattern and released the Prometheus Operator as a working example of the pattern. The Prometheus Operator automatically creates and manages Prometheus monitoring instances.
- The mission of the Prometheus Operator is to make running Prometheus on top of Kubernetes as easy as possible, while preserving configurability as well as making the configuration Kubernetes native.
  https://coreos.com/operators/prometheus/docs/latest/user-guides/getting-started.html

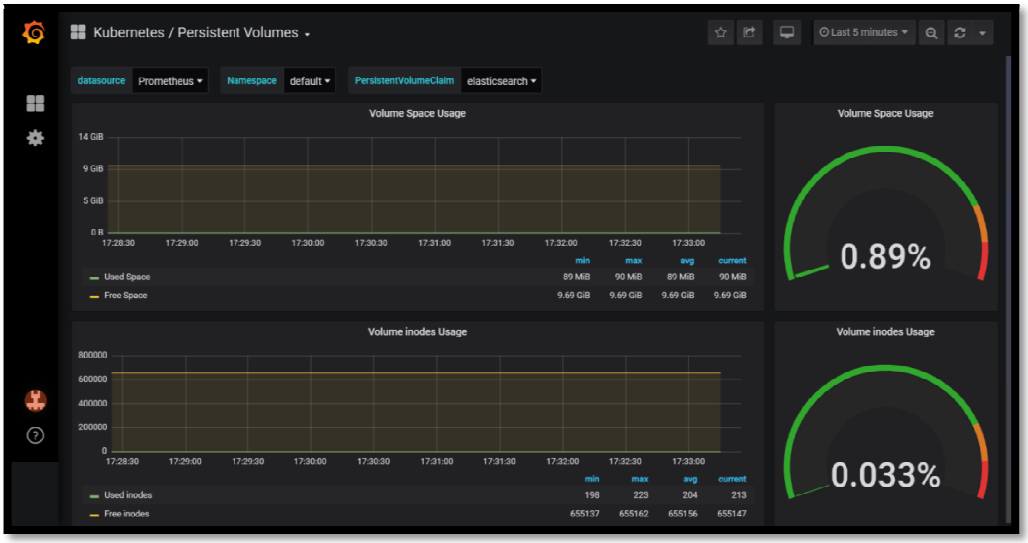
- The Prometheus Operator purpose is to make our life easier — deployment and maintenance.



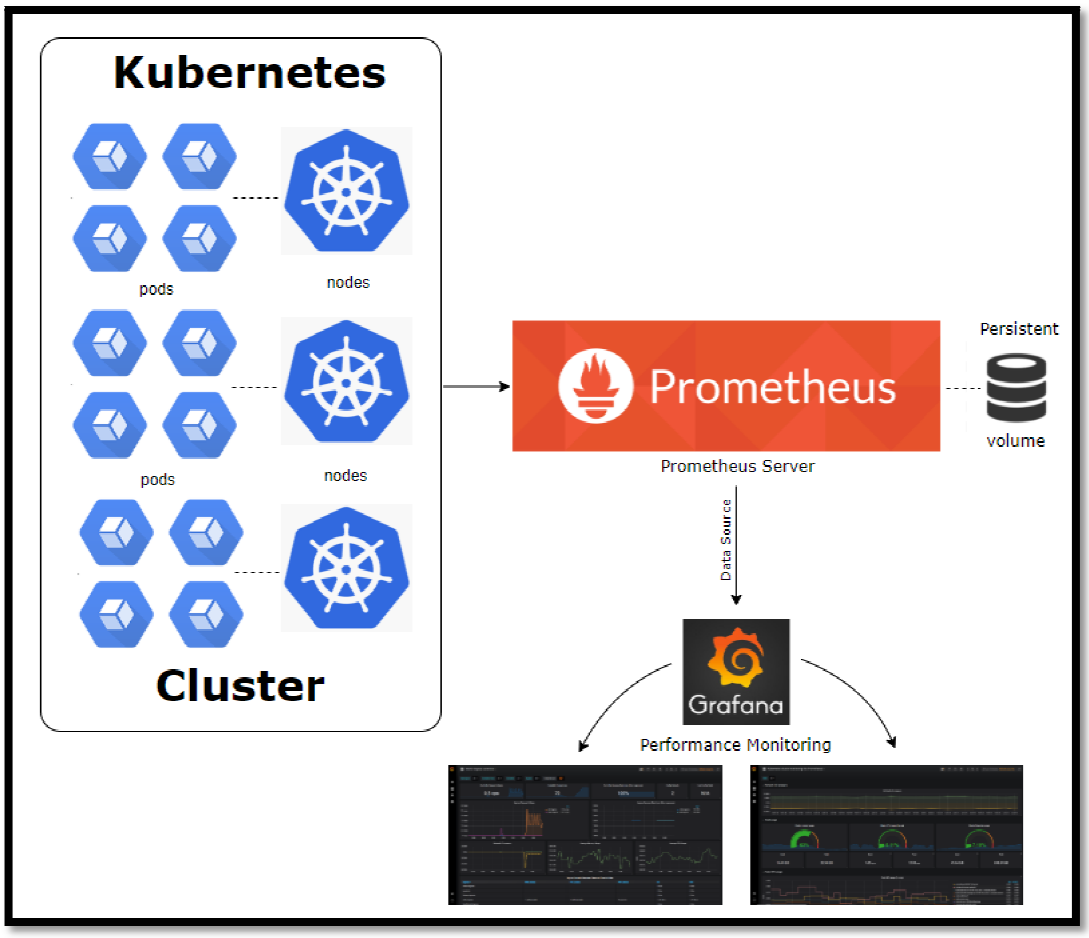
- After we successfully deployed Prometheus Operator we should see a new CRDs (Custom Resource Definition):
  - ✓ Prometheus, which defines a desired Prometheus deployment.
  - ✓ Service Monitor, which declaratively specifies how groups of services should be monitored. The Operator automatically generates Prometheus scrape configuration based on the definition.
  - ✓ Alert manager, which defines a desired Alert manager deployment.
- When a new version for your service is getting update a new pod is created. Prometheus is watching over k8s API so when it detects this kind of changes it will create a new set of configuration for this new service (pod).

What is Grafana & What Is It Used For?

- Grafana is an open source solution for running data analytics, pulling up metrics that make sense of the massive amount of data & to monitor our apps with the help of cool customizable dashboards.
- Grafana connects with every possible data source, commonly referred to as databases such as Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL etc.
- Grafana being an open source solution also enables us to write plugins from scratch for integration with several different data sources.
- The tool helps us study, analyze & monitor data over a period of time, technically called time series analytics.
- It helps us track the user behavior, application behavior, frequency of errors popping up in production or a pre-prod environment, type of errors popping up & the contextual scenarios by providing relative data.
- A big upside of the project is it can be deployed on-perm by organizations which do not want their data to be streamed over to a vendor cloud for security & other reasons.
- Over time this framework has gained a lot of popularity in the industry & is deployed by big guns such as PayPal, eBay, Intel & many more. I'll talk about the industry use cases up ahead in the article.
- Besides the core open source solution there are other two services offered by the Grafana team for businesses known as the Grafana Cloud & the Enterprise.
  Ref Link: https://grafana.com/docs/grafana/latest/



Overview: Kubernetes-Monitoring-System



Install the Prometheus in Kubernetes

Create the New namespace for monitoring

```
$ kubectl create namespace monitoring
$ kubectl get namespace
```

```
[ec2-user@ip-12-0-14-210 ~]$ kubectl create namespace monitoring
Error from server (AlreadyExists): namespaces "monitoring" already exists
[ec2-user@ip-12-0-14-210 ~]$
[ec2-user@ip-12-0-14-210 ~]$ kubectl get namespace
NAME          STATUS   AGE
default       Active   24m
kube-public   Active   24m
kube-system   Active   24m
monitoring    Active   119s
[ec2-user@ip-12-0-14-210 ~]$
```

Install the Prometheus-operator in kubernetes cluster

```
[ec2-user@ip-12-0-14-210 ~]$ helm install --name monitoring --namespace monitoring stable/prometheus-operator
```

View the monitoring namespace details here

```
[ec2-user@ip-12-0-14-210 ~]$ kubectl get all --namespace monitoring
NAME                                                     READY   STATUS    RESTARTS   AGE
pod/alertmanager-monitoring-prometheus-oper-alertmanager-0   2/2     Running   0          5m6s
pod/monitoring-grafana-76c7ff6545-nb7wl                  2/2     Running   0          5m26s
pod/monitoring-kube-state-metrics-6769d4bc56-kvlqr       1/1     Running   0          5m26s
pod/monitoring-prometheus-node-exporter-2pmll            1/1     Running   0          5m26s
pod/monitoring-prometheus-node-exporter-9vszz            1/1     Running   0          5m26s
pod/monitoring-prometheus-node-exporter-ddckt            1/1     Running   0          5m26s
pod/monitoring-prometheus-node-exporter-hn76w            1/1     Running   0          5m26s
pod/monitoring-prometheus-node-exporter-l4xwm            1/1     Running   0          5m26s
pod/monitoring-prometheus-node-exporter-r4957            1/1     Running   0          5m26s
pod/monitoring-prometheus-oper-operator-676c96954c-6226q 2/2     Running   0          5m26s
pod/prometheus-monitoring-prometheus-oper-prometheus-0   3/3     Running   1          4m56s

NAME                                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
service/alertmanager-operated                     ClusterIP   None             <none>        9093/TCP,9094/TCP,9094/UDP   5m6s
service/monitoring-grafana                        ClusterIP   100.71.249.233   <none>        80/TCP                       5m26s
service/monitoring-kube-state-metrics             ClusterIP   100.67.210.87    <none>        8080/TCP                     5m26s
service/monitoring-prometheus-node-exporter       ClusterIP   100.64.122.38    <none>        9100/TCP                     5m26s
service/monitoring-prometheus-oper-alertmanager   ClusterIP   100.68.254.29    <none>        9093/TCP                     5m26s
service/monitoring-prometheus-oper-operator       ClusterIP   100.68.63.27     <none>        8080/TCP,443/TCP             5m26s
service/monitoring-prometheus-oper-prometheus     ClusterIP   100.67.38.254    <none>        9090/TCP                     5m26s
service/prometheus-operated                       ClusterIP   None             <none>        9090/TCP                     4m56s

NAME                                                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/monitoring-prometheus-node-exporter   6         6         6       6            6           <none>          5m26s
```

```
NAME                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/monitoring-grafana                  1/1     1            1           5m26s
deployment.apps/monitoring-kube-state-metrics       1/1     1            1           5m26s
deployment.apps/monitoring-prometheus-oper-operator 1/1     1            1           5m26s

NAME                                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/monitoring-grafana-76c7ff6545                  1         1         1       5m26s
replicaset.apps/monitoring-kube-state-metrics-6769d4bc56       1         1         1       5m26s
replicaset.apps/monitoring-prometheus-oper-operator-676c96954c 1         1         1       5m26s

NAME                                                             READY   AGE
statefulset.apps/alertmanager-monitoring-prometheus-oper-alertmanager   1/1     5m6s
statefulset.apps/prometheus-monitoring-prometheus-oper-prometheus       1/1     4m56s
[ec2-user@ip-12-0-14-210 ~]$
```

To get a Prometheus dashboard to browser, we have to make a service to Lodbalancer.

So that we have to edit the Prometheus services

```
NAME                                              TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
service/alertmanager-operated                     ClusterIP      None             <none>        9093/TCP,9094/TCP,9094/UDP   13m
service/monitoring-grafana                        ClusterIP      100.71.249.233   <none>        80/TCP                       14m
service/monitoring-kube-state-metrics             ClusterIP      100.67.210.87    <none>        8080/TCP                     14m
service/monitoring-prometheus-node-exporter       ClusterIP      100.64.122.38    <none>        9100/TCP                     14m
service/monitoring-prometheus-oper-alertmanager   ClusterIP      100.68.254.29    <none>        9093/TCP                     14m
service/monitoring-prometheus-oper-operator       ClusterIP      100.68.63.27     <none>        8080/TCP,443/TCP             14m
service/monitoring-prometheus-oper-prometheus     LoadBalancer   100.67.38.254    <pending>     9090:30622/TCP               14m
service/prometheus-operated                       ClusterIP      None             <none>        9090/TCP                     13m
```

```
[ec2-user@ip-12-0-14-210 ~]$ kubectl edit -n monitoring service/monitoring-prometheus-oper-prometheus
service/monitoring-prometheus-oper-prometheus edited
[ec2-user@ip-12-0-14-210 ~]$
```

Here type is ClusterIP to LoadBalncer

```
  selector:
    app: prometheus
    prometheus: monitoring-prometheus-oper-prometheus
  sessionAffinity: None
  type: LoadBalancer
status:
  loadBalancer: {}
```

After changes the Service we got a loadbalncer.

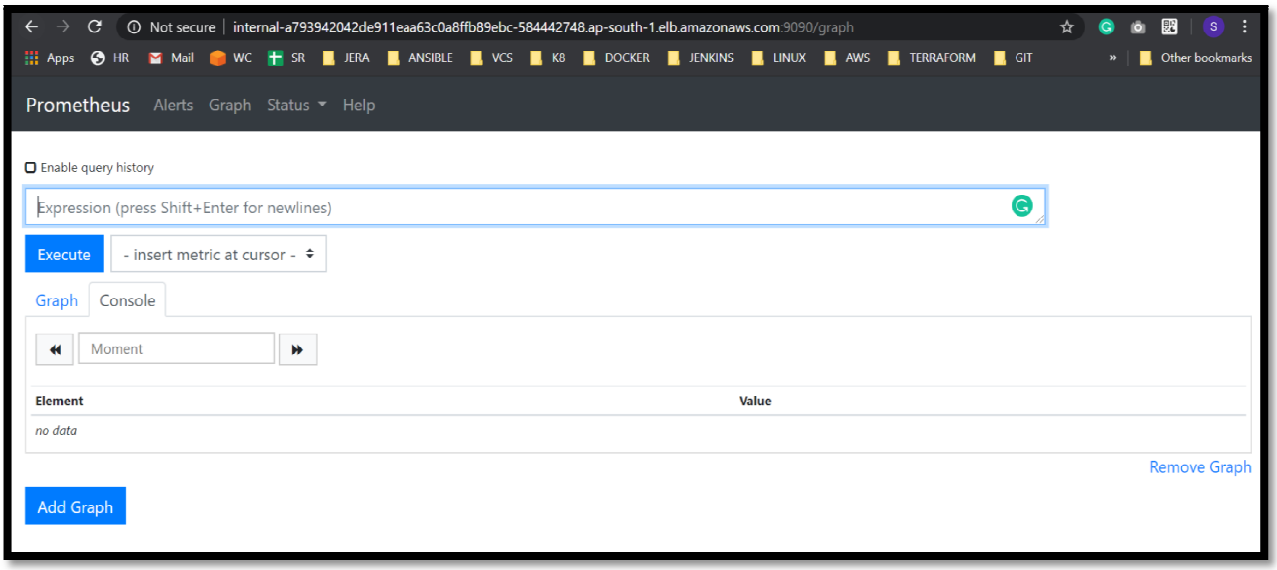
```
[ec2-user@ip-12-0-14-210 ~]$ kubectl get svc -n monitoring
NAME                                     TYPE           CLUSTER-IP       EXTERNAL-IP
                 PORT(S)                       AGE
alertmanager-operated                    ClusterIP      None             <none>
                 9093/TCP,9094/TCP,9094/UDP    49m
monitoring-grafana                       ClusterIP      100.71.249.233   <none>
                 80/TCP                        49m
monitoring-kube-state-metrics            ClusterIP      100.67.210.87    <none>
                 8080/TCP                      49m
monitoring-prometheus-node-exporter      ClusterIP      100.64.122.38    <none>
                 9100/TCP                      49m
monitoring-prometheus-oper-alertmanager  ClusterIP      100.68.254.29    <none>
                 9093/TCP                      49m
monitoring-prometheus-oper-operator      ClusterIP      100.68.63.27     <none>
                 8080/TCP,443/TCP              49m
monitoring-prometheus-oper-prometheus    LoadBalancer   100.67.38.254    internal-a793942042de911eaa63c0a8ffb89ebc-584442748.ap-south-1.e
lb.amazonaws.com   9090:30622/TCP              49m
prometheus-operated                      ClusterIP      None             <none>
                 9090/TCP                      48m
[ec2-user@ip-12-0-14-210 ~]$
```

In AWS Console

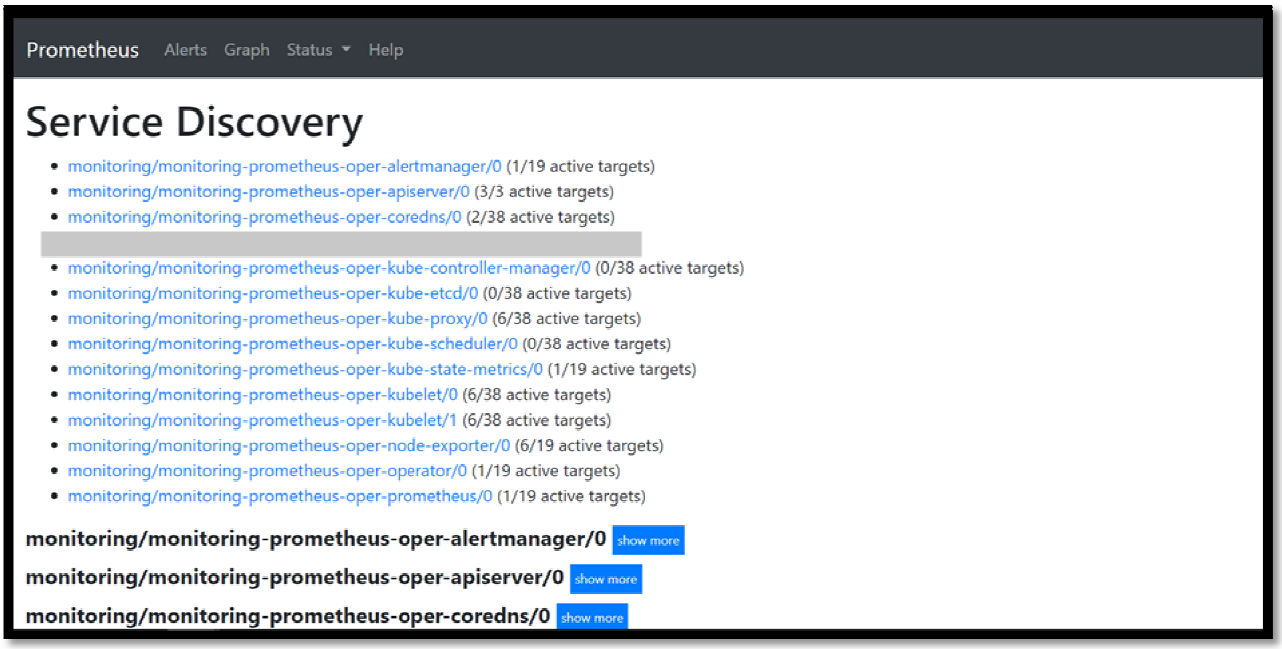
To access a Prometheus dashboard using load balancer URL.

http://internal-a793942042de911eaa63c0a8ffb89ebc-584442748.ap-south-1.elb.amazonaws.com:9090

Prometheus Port No is : 9090

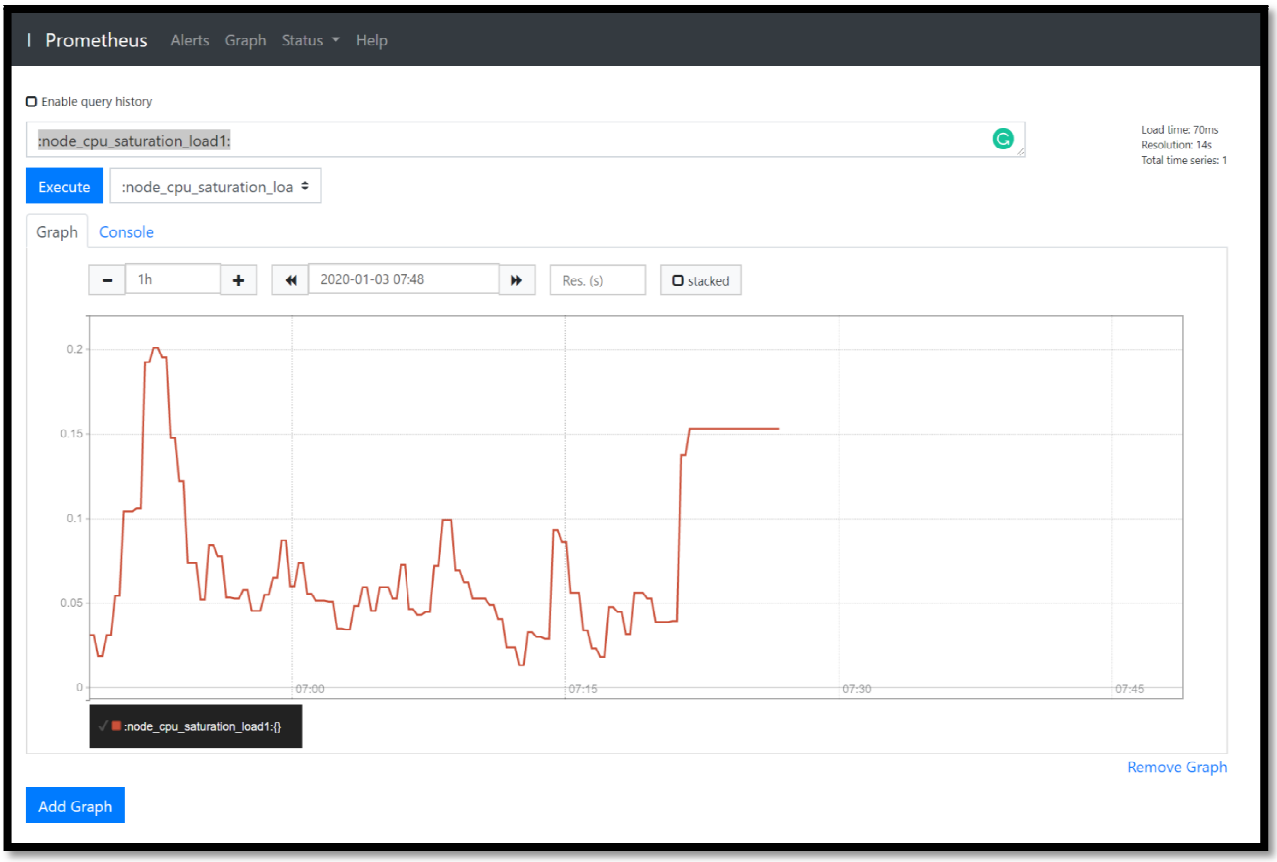


Service discovery



Sample query to execute & get the graph view.

In below picture will tell you last 1 hours node_cpu_stturation_load1 view.



Install the Grafana in Kubernetes

```
$ $ helm install --name monitoring --namespace monitoring stable/Grafana
```

We need to access the Grafana dashboard in browser to view or create our own dashboard for kubernetes details.

```
$ kubectl edit -n monitoring service/monitoring-grafana
```

Load balancer adding.

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-internal: 0.0.0.0/0
  creationTimestamp: "2020-01-03T05:25:37Z"
```

```
  selector:
    app: grafana
    release: monitoring
  sessionAffinity: None
  type: LoadBalancer
```

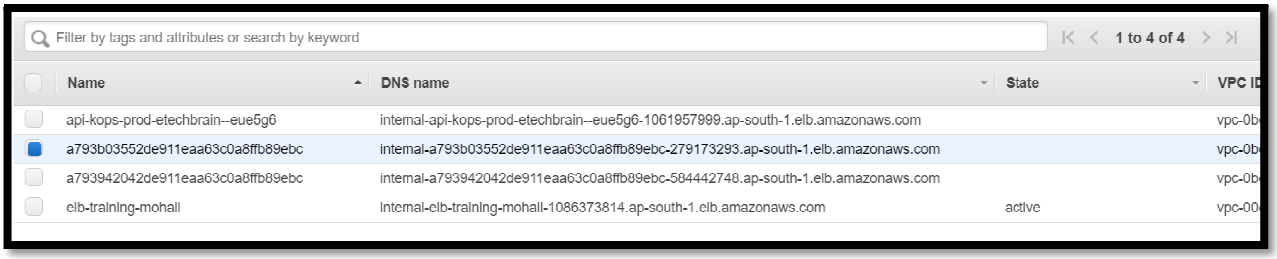
To describe the service

```
[ec2-user@ip-12-0-14-210 ~]$ kubectl edit -n monitoring service/monitoring-grafana
service/monitoring-grafana edited
[ec2-user@ip-12-0-14-210 ~]$ kubectl describe svc monitoring-grafana -n monitoring
Name:                     monitoring-grafana
Namespace:                monitoring
Labels:                   app=grafana
                          chart=grafana-4.2.2
                          heritage=Tiller
                          release=monitoring
Annotations:              service.beta.kubernetes.io/aws-load-balancer-internal: 0.0.0.0/0
Selector:                 app=grafana,release=monitoring
Type:                     LoadBalancer
IP:                       100.71.249.233
LoadBalancer Ingress:     internal-a793b03552de911eaa63c0a8ffb89ebc-279173293.ap-south-1.elb.amazonaws.com
Port:                     service  80/TCP
TargetPort:               3000/TCP
NodePort:                 service  30662/TCP
Endpoints:                100.106.26.194:3000
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                Age   From                Message
  ----    ------                ----  ----                -------
  Normal  Type                  17s   service-controller  ClusterIP -> LoadBalancer
  Normal  EnsuringLoadBalancer  17s   service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer   15s   service-controller  Ensured load balancer
[ec2-user@ip-12-0-14-210 ~]$
```

Internal LoadBalncer is created.

http://internal-a793b03552de911eaa63c0a8ffb89ebc-279173293.ap-south-1.elb.amazonaws.com

In AWS Console

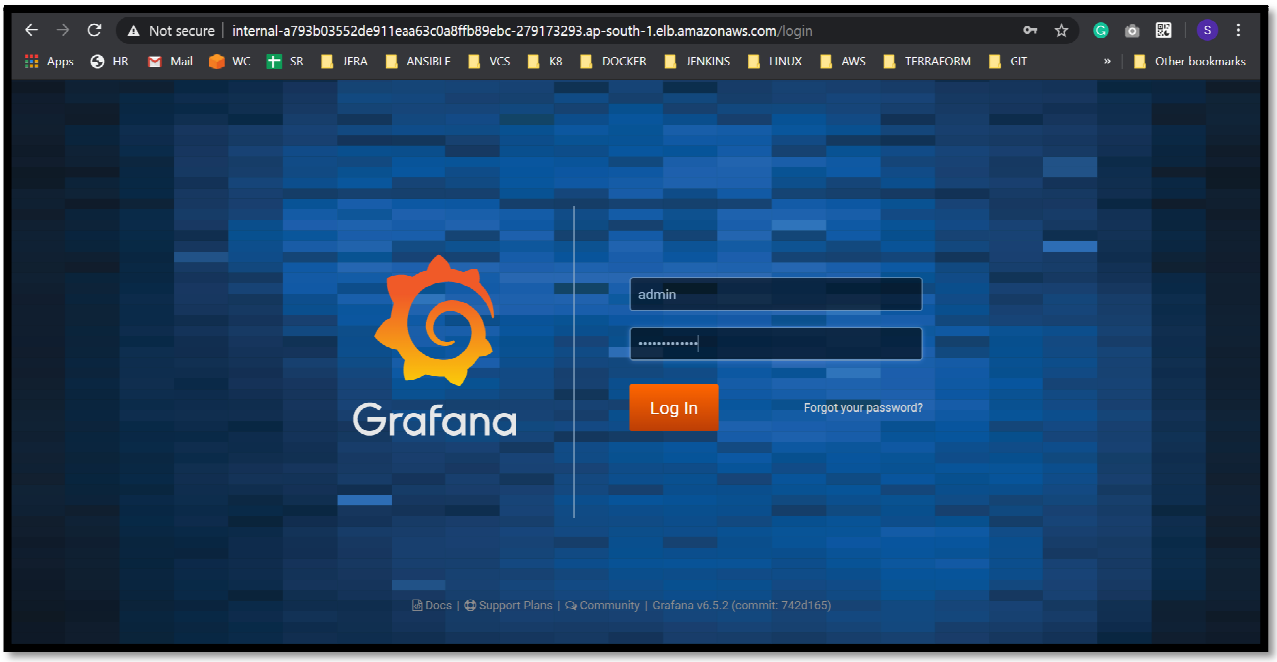


To access a Grafana dashboard url in browser

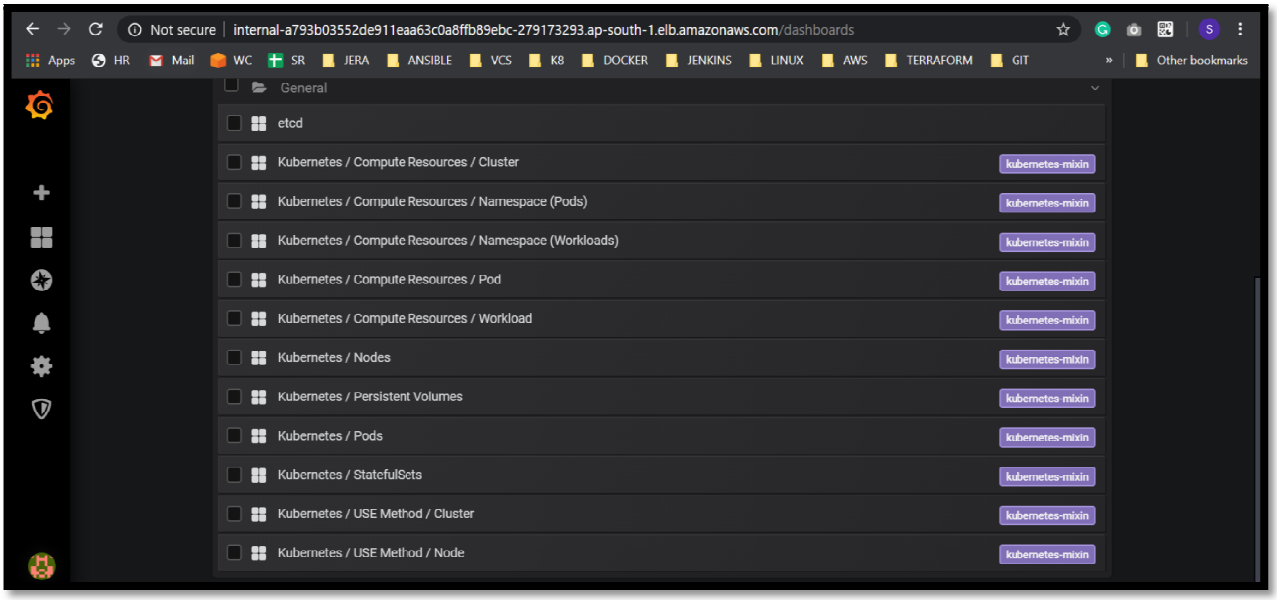
http://internal-a793b03552de911eaa63c0a8ffb89ebc-279173293.ap-south-1.elb.amazonaws.com

Default username: admin

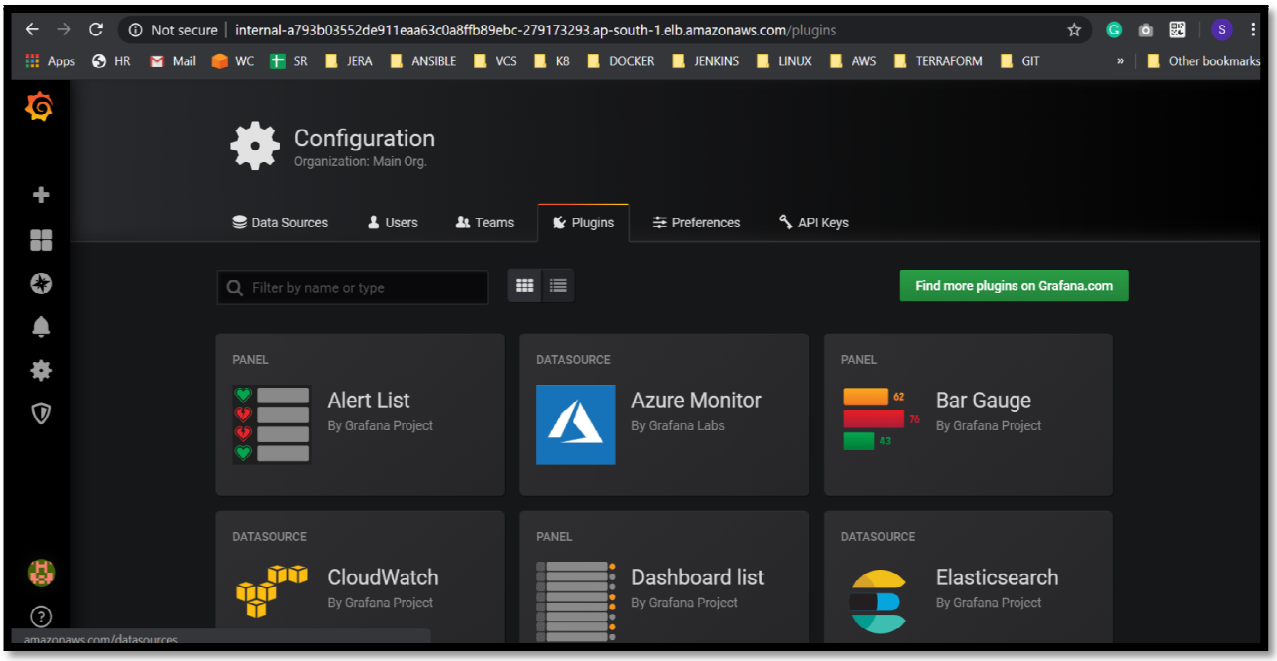
Default password: prom-operator



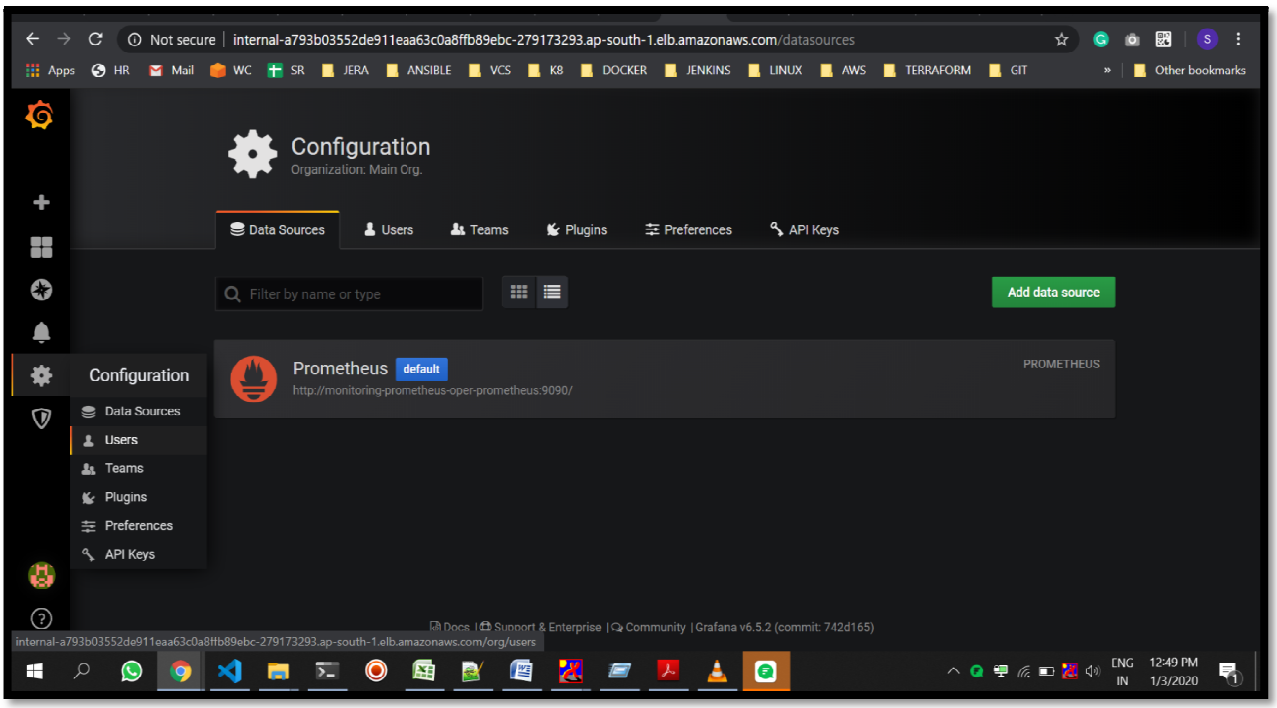
The entire predefined dashboard for kubernetes.



Install the plugins in Grafana

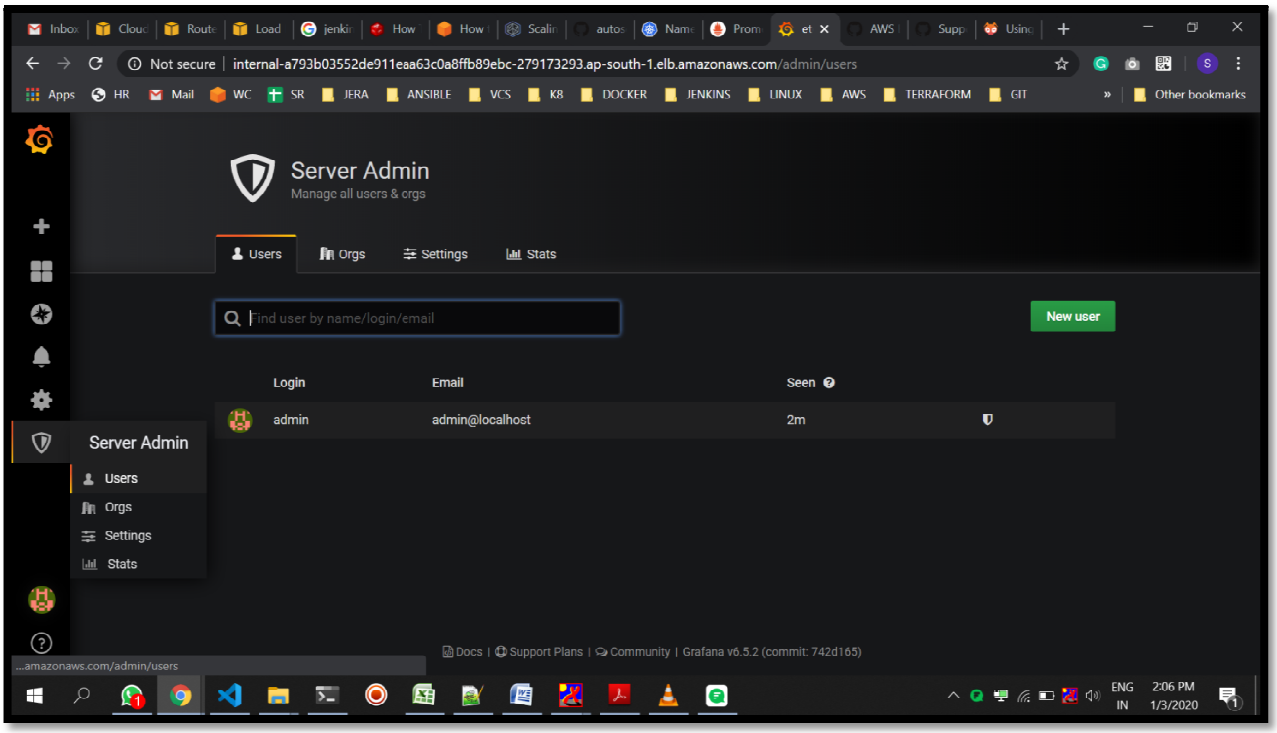


In below picture Prometheus already available, we have installed before Grafana dashboard details collecting from Prometheus.

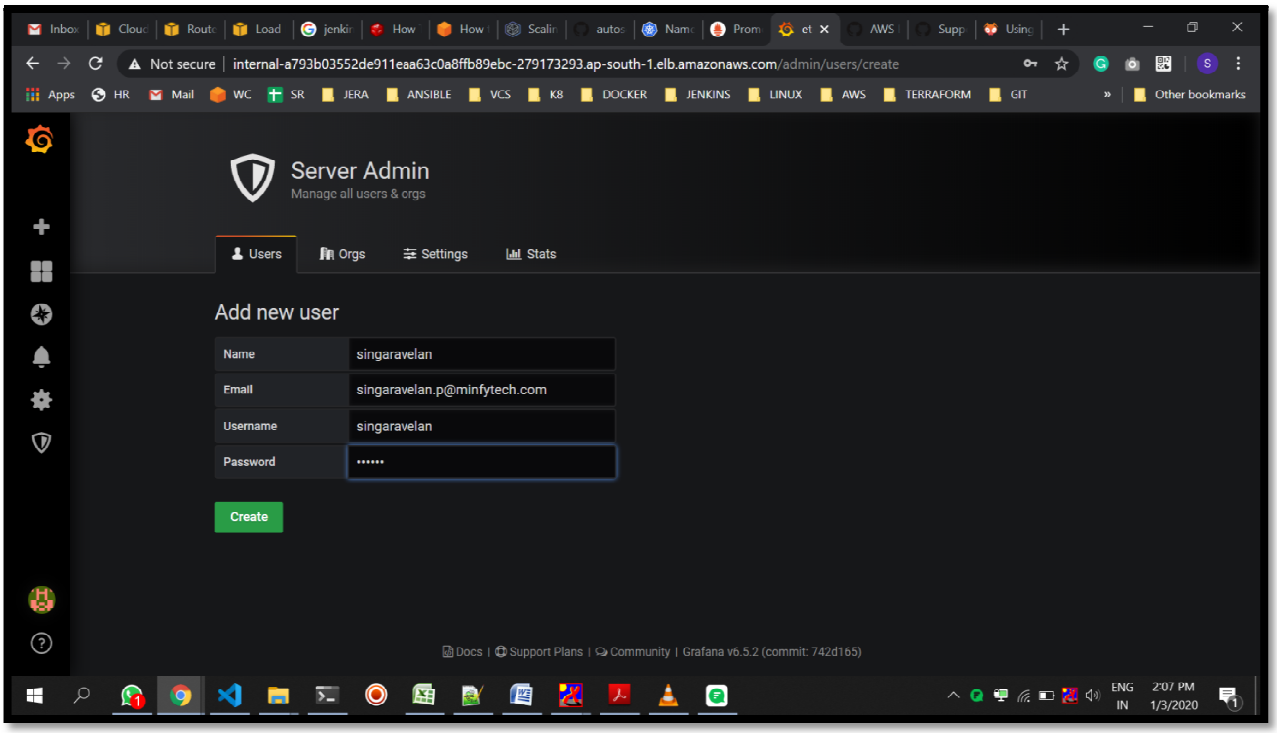


User creation for Grafana dashboard

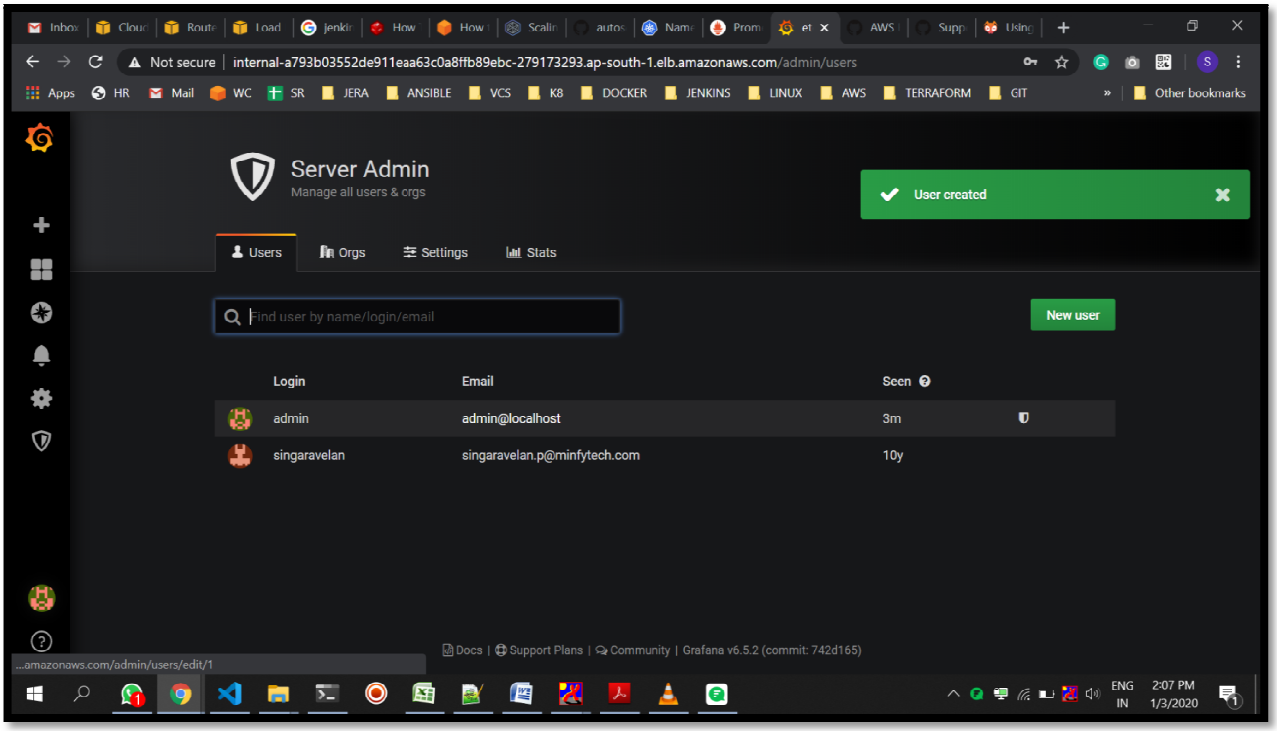
Go to users -> click New user



Fill the user details



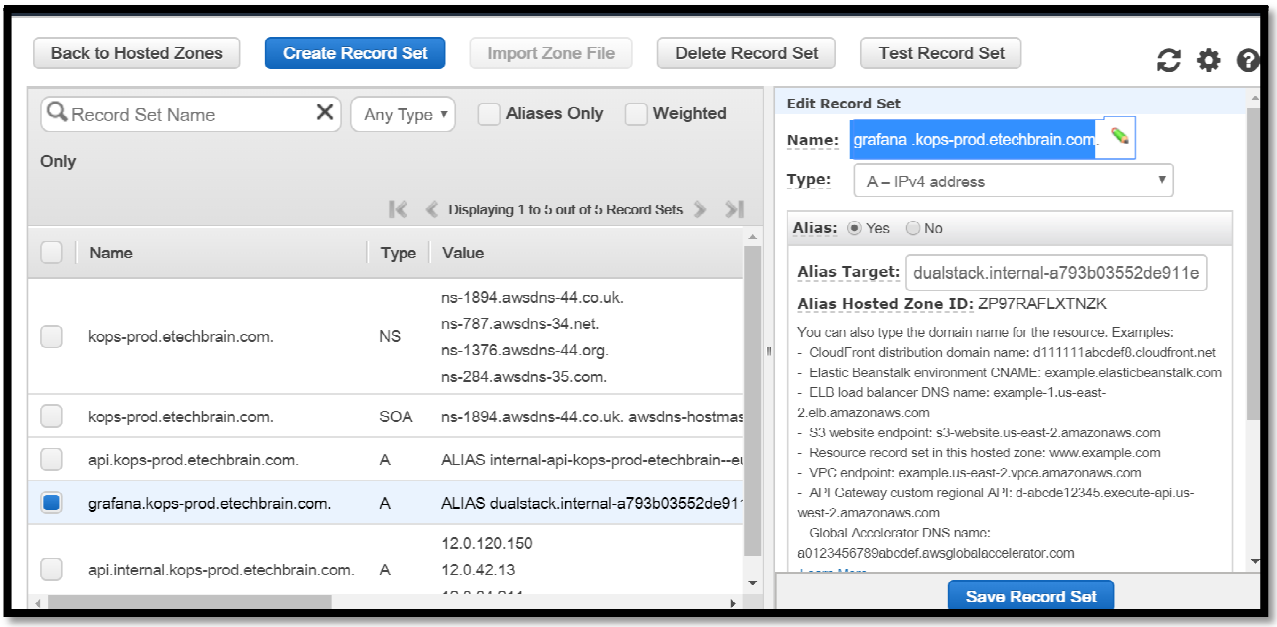
User creation successful



Grafana dashboard assess via domain name

Domain name is : Grafana.kops-prod.etechbrain.com

URL: http://grafana.kops-prod.etechbrain.com

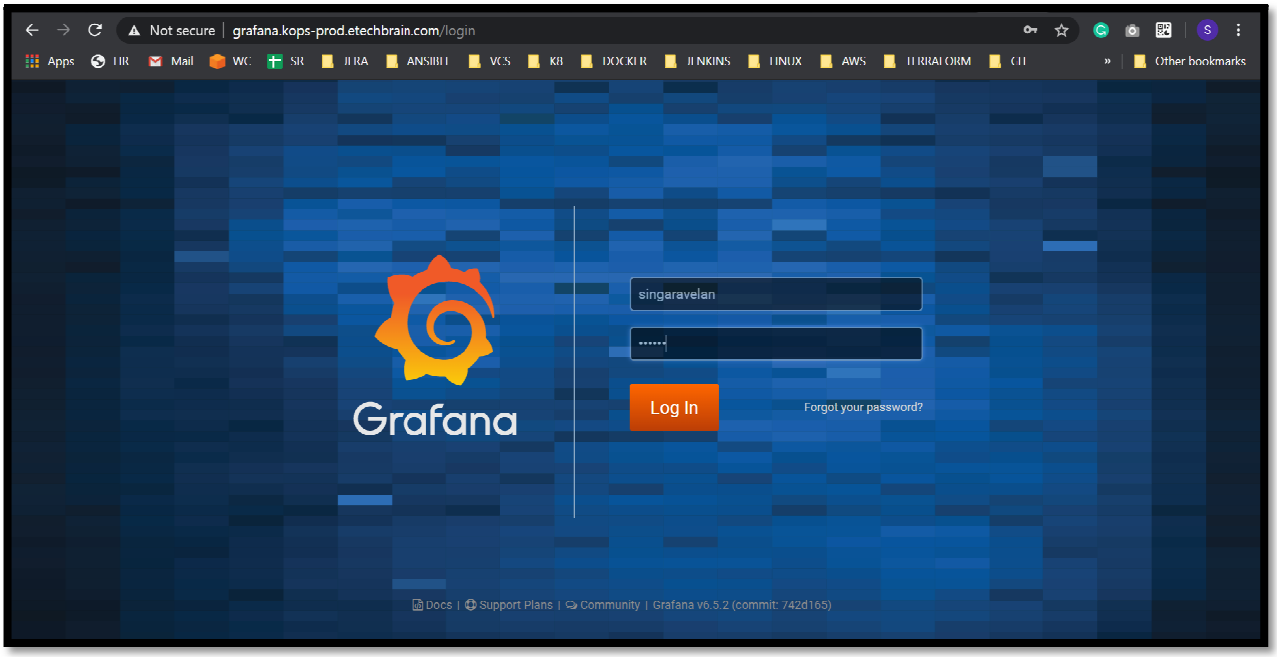


We have mapped to LoadBalncer url

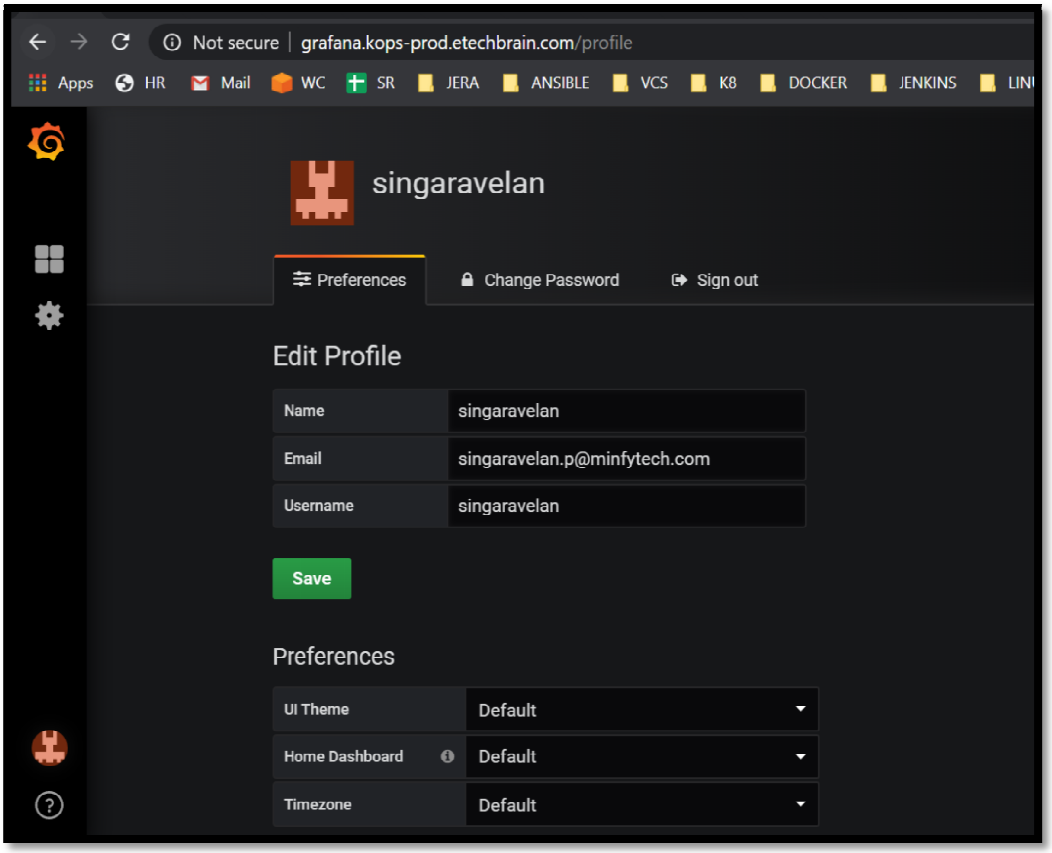
To login via domain URL with newly created user

User Name: singaravelan

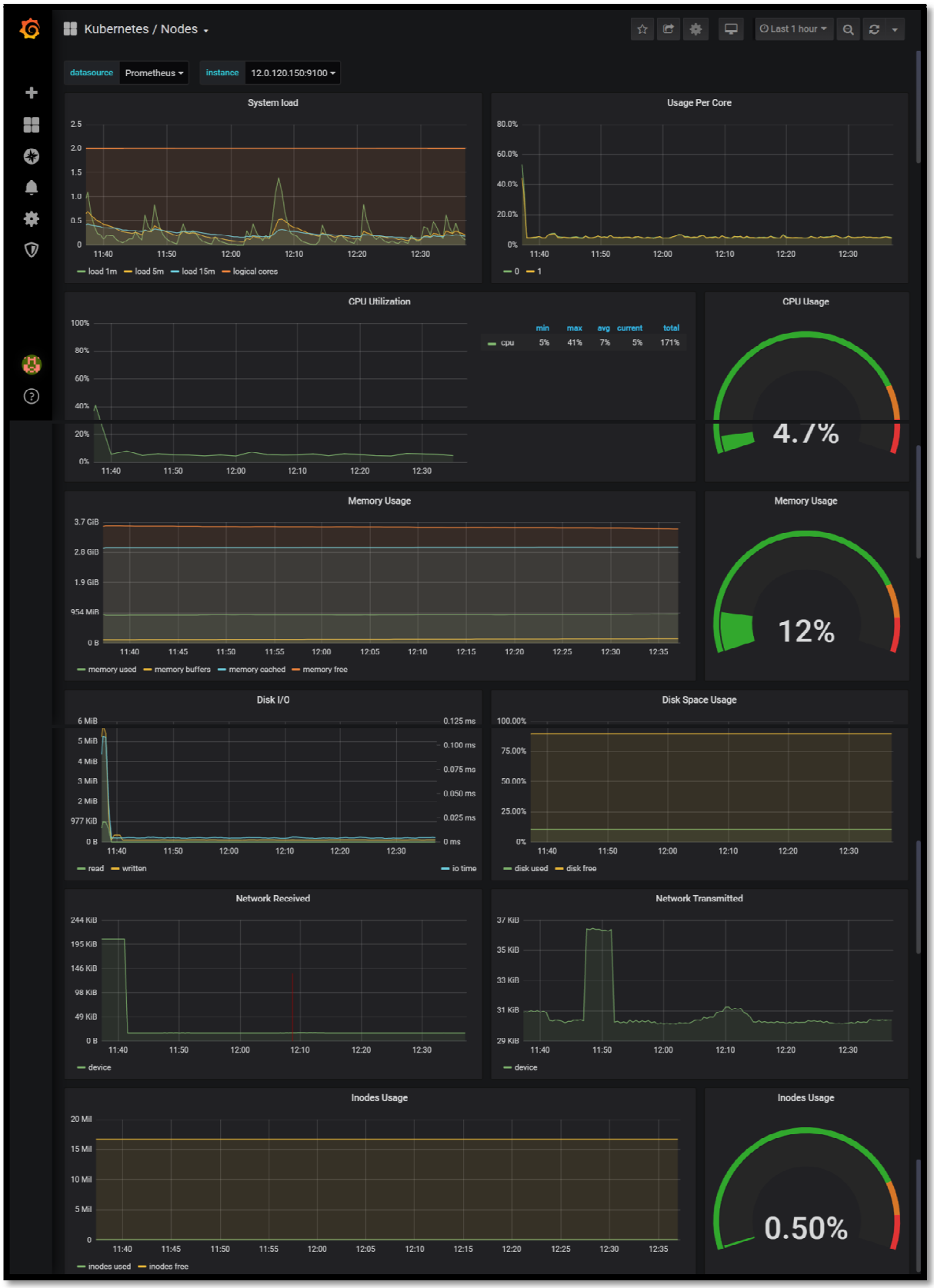
Password : redhat



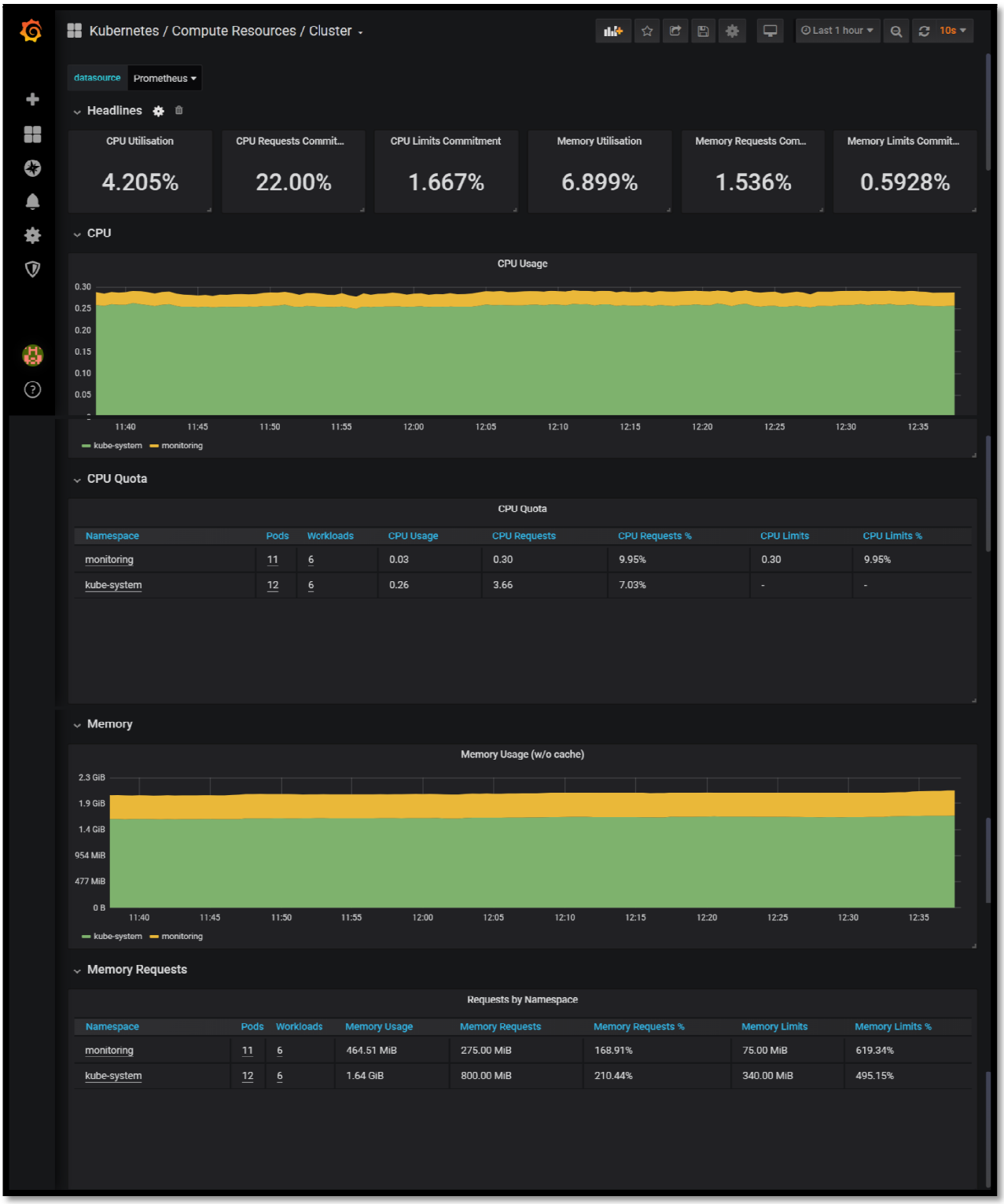
Singaravelan user profile page



Kubernetes nodes view in Grafana

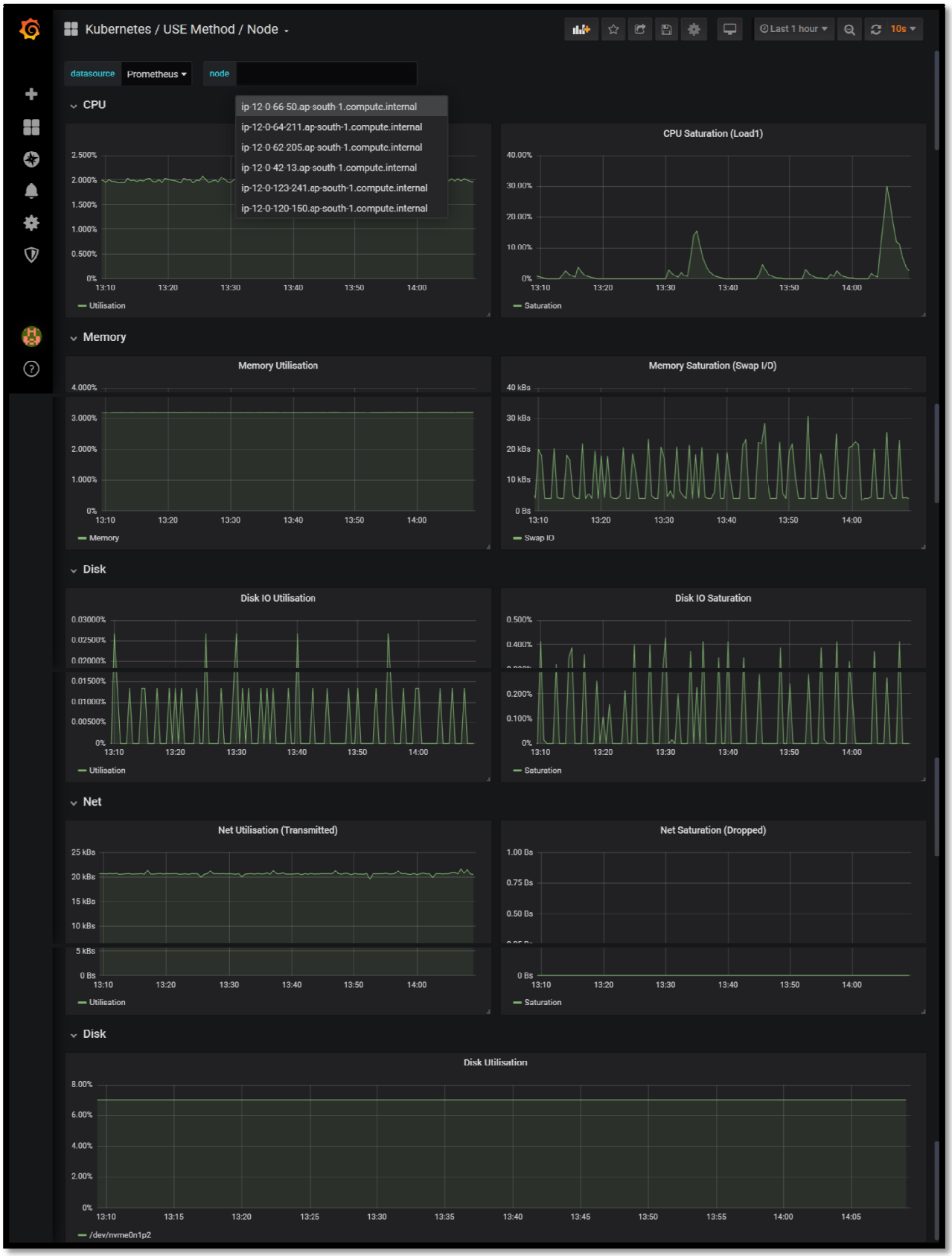


Kubernetes Compute Resource Cluster view in Grafana



Kubernetes Pods view in Grafana

Kubernets USE Method Node view for Grafana



Kubernetes Persistent Volume view for Grafana