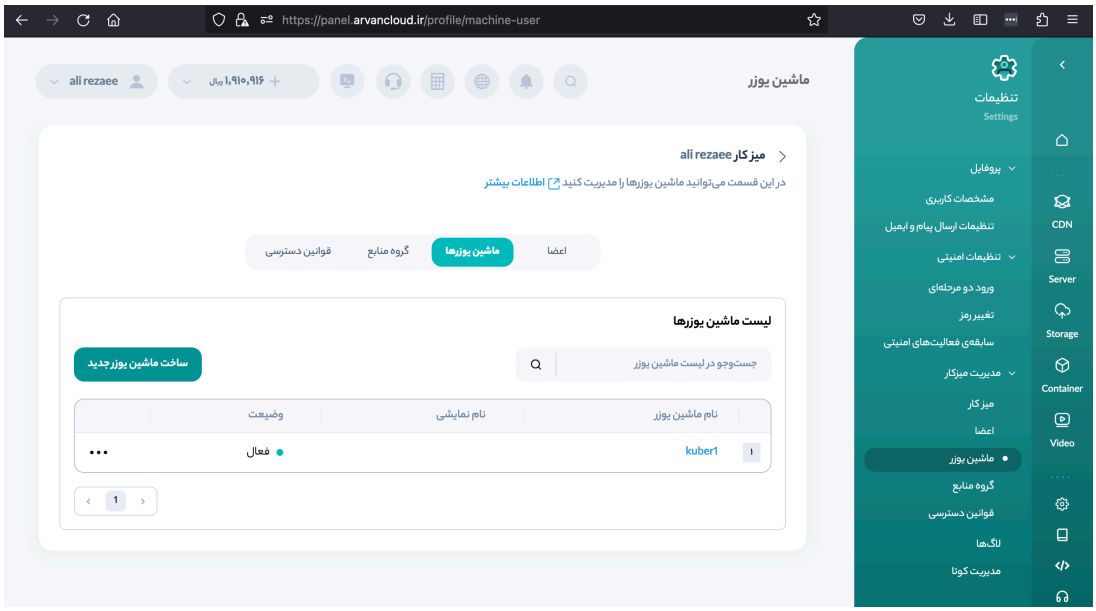
All commands in document were executed in Mac
You can access the code in private repository in git with this link:
Githubdsdfdsf

Phase 1(Terraform):

First install terraform:

```
brew tap hashicorp/tap

brew install hashicorp/tap/terraform
```

Then go to arvancloud site and create api key:



Then create folder terraform and make file called main.tf in it and write liked this in it and put your key in file like this:

main.tf

terraform > main.tf

```
terraform {
  required_providers {
    arvan = {
      source = "arvancloud/arvan"
      version = "0.6.4"
    }
  }
}
variable "ApiKey" {
    type = string
    default = "apikey 0027674a-a12b-5c83-adca-a4f8c7672711"
    sensitive = true
```

Then write command :

```
user@users-MacBook-Pro terraform-provider-arvan % terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of arvancloud/arvan from the dependency lock file
- Using previously-installed arvancloud/arvan v0.6.4

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

And see the result. It is important that before this your vpn is connected.

And then write this command :

```
user@users-MacBook-Pro terraform % terraform apply

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create
 <= read (data resources)

Terraform will perform the following actions:

  # data.arvan_iaas_abrak.get_abrak_id will be read during apply
  # (depends on a resource or a module with changes pending)
 <= data "arvan_iaas_abrak" "get_abrak_id" {
      + ha_enabled = (known after apply)
      + id         = (known after apply)
      + name       = "kuber1"
      + region     = "ir-thr-c1"
    }

  # arvan_iaas_abrak.kuber1 will be created
  + resource "arvan_iaas_abrak" "kuber1" {
      + addresses    = (known after apply)
      + disk_size    = 25
      + flavor       = "g1-4-8-0"
```

And write yes:

```
  Enter a value: yes
```

You can see that it is processing :

```
  Enter a value: yes

arvan_iaas_abrak.kuber1: Creating...
arvan_iaas_abrak.kuber1: Still creating... [10s elapsed]
```

But unfortunately, it gives me 403:

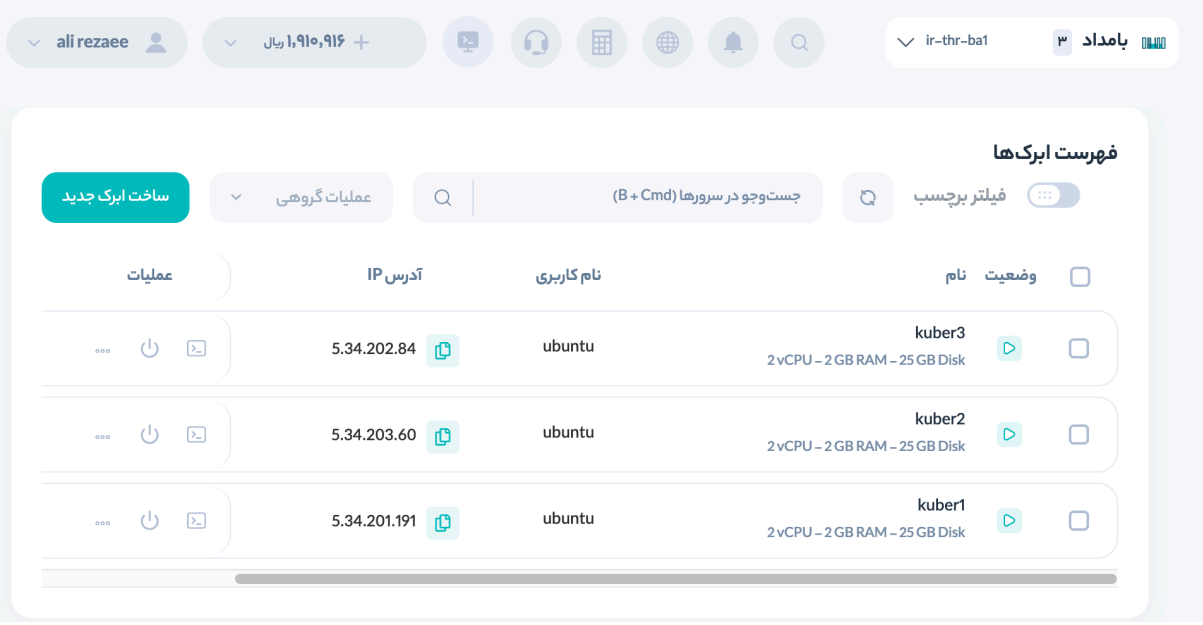
```
Error: status: 403, body: {"message":"امش دسترسی مزال ی ارب م اچنا ی ن ار اک ر ادن دی","e
rrors":[]}


  with arvan_iaas_abrak.kuber1,
  on  main.tf line 33, in resource "arvan_iaas_abrak" "kuber1":
  33: resource "arvan_iaas_abrak" "kuber1" {
```

And I change it’s flavor or run this command:

```
user@users-MacBook-Pro terraform % TF_VAR_ApiKey="apikey 0027674a-a12b-5c83-adca-a4f8c76
72711" terraform apply
```

But it didn’t work, and I create VM manually

So you can see the result I create three vms :



Phase 2(K8s cluster):

Before this step you should set Shekan on all vms

Then I download ansible play book from this link:
https://github.com/deveth0/kubernetes-cluster-ansible/tree/master

to install kuber cluster on this three vms you should first configure the inventory file and thake care that all user should be root:

kubernetes-cluster-ansible > inventory

```
[all]
kubernetes
kubernetes-node-1
kubernetes-node-1

[all:vars]
# Ubuntu uses python3 instead of python2, we need
ansible_python_interpreter=/usr/bin/python3

[kube-master]
root@5.34.202.84


[kube-node]
root@5.34.203.60
root@188.121.114.8
```

and then run kube-install-software.yml playbook by this command to install the requirements:

```
user@users-MacBook-Pro kubernetes-cluster-ansible-master % ansible-playbook -i inventory kube-install-software.yml -K
BECOME password:
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [kube-master:kube-node] ***************************************************************

TASK [Gathering Facts] *********************************************************************
ok: [root@5.34.202.84]
ok: [root@5.34.203.60]
ok: [root@5.34.201.191]

TASK [Get Kubernetes apt-key] **************************************************************
ok: [root@5.34.203.60]
ok: [root@5.34.202.84]
ok: [root@5.34.201.191]

TASK [Add Kubernetes APT repository] *******************************************************
ok: [root@5.34.203.60]
ok: [root@5.34.202.84]
ok: [root@5.34.201.191]

TASK [Install required software] ***********************************************************
ok: [root@5.34.202.84]
ok: [root@5.34.203.60]
ok: [root@5.34.201.191]

TASK [enable Docker service] ***************************************************************
ok: [root@5.34.202.84]
ok: [root@5.34.203.60]
ok: [root@5.34.201.191]

PLAY RECAP *********************************************************************************
root@5.34.201.191          : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
root@5.34.202.84           : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
root@5.34.203.60           : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

user@users-MacBook-Pro kubernetes-cluster-ansible-master %
```

Then run k kube-setup-cluster.yml playbook by this command to setup master and join the node to it:

```
⊗ user@users-MacBook-Pro kubernetes-cluster-ansible-master % ansible-playbook -i inventory kube-setup-cluster.yml -K

BECOME password:
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details

PLAY [kube-master] ************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************
ok: [root@5.34.202.84]

TASK [Install kubectl on Master] **********************************************************************************
ok: [root@5.34.202.84]

TASK [Initialize Cluster] *****************************************************************************************
ok: [root@5.34.202.84]

TASK [Create .kube directory] *************************************************************************************
ok: [root@5.34.202.84]

TASK [Copy admin.conf to the user's kube directory] ***************************************************************
ok: [root@5.34.202.84]

TASK [Setup Flannel. Use log to prevent second installation] ******************************************************
ok: [root@5.34.202.84]

TASK [Create token to join cluster] *******************************************************************************
changed: [root@5.34.202.84]

TASK [Set join command as fact] ***********************************************************************************
ok: [root@5.34.202.84]

PLAY [kube-node] **************************************************************************************************

TASK [Gathering Facts] ********************************************************************************************
ok: [root@5.34.203.60]
ok: [root@5.34.201.191]

TASK [Wait for master's port 6443] ********************************************************************************
ok: [root@5.34.203.60]
ok: [root@5.34.201.191]

TASK [Join the cluster. Use log to prevent joining twice] *********************************************************
fatal: [root@5.34.203.60]: FAILED! => {"msg": "The task includes an option with an undefined variable. The error w
as: \"hostvars['kube-master']\" is undefined. \"hostvars['kube-master']\" is undefined\n\nThe error appears to be
in '/Users/user/Desktop/sreArvan/kubernetes-cluster-ansible/kube-setup-cluster.yml': line 52, column 7, but may\nb
e elsewhere in the file depending on the exact syntax problem.\n\nThe offending line appears to be:\n\n\n    - nam
e: Join the cluster. Use log to prevent joining twice\n      ^ here\n"}
```

But it gives me this at the final step when it wants to join the node to master and I changed the inventory file but it doesn't work so I join them manually by run this command on master to get token:

```
root@kuber3:/home/ubuntu# kubeadm token create --print-join-command
kubeadm join 5.34.202.84:6443 --token bzmzyu.awlqzti3q8lgggiw --disco
```

Then run this command to each node to connect to the master:

```
root@kuber1:~# kubeadm join 5.34.202.84:6443 --token bzmzyu.awlqzti3q8lgggiw --discovery-
[preflight] Running pre-flight checks
        [WARNING Hostname]: hostname "kuber1" could not be reached
        [WARNING Hostname]: hostname "kuber1": lookup kuber1 on 178.22.122.100:53: no suc
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kub
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kub
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Now you should see this result when execute this command on master :

```
root@kuber3:/home/ubuntu# kubectl get nodes
NAME     STATUS     ROLES           AGE    VERSION
kuber1   NotReady   <none>          9s     v1.28.2
kuber2   NotReady   <none>          40m    v1.28.2
kuber3   Ready      control-plane   113m   v1.28.2
root@kuber3:/home/ubuntu#
```

Phase 3(monitor k8s with Prometheus and Grafana):

I use this link to deploy Prometheus on Kubernetes:
https://grafana.com/blog/2023/01/19/how-to-monitor-kubernetes-clusters-with-the-prometheus-operator/

first I run this to deploy Prometheus:

```
ubuntu@kuber3:~$ sudo su -
sudo: unable to resolve host kuber3: Name or service not known
root@kuber3:~# kubectl apply -f https://raw.githubusercontent.com/prometheus-operator/prometheus-operator/main/bundle.yaml
customresourcedefinition.apiextensions.k8s.io/alertmanagerconfigs.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/alertmanagers.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/podmonitors.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/probes.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/prometheusrules.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/scrapeconfigs.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/servicemonitors.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/thanosrulers.monitoring.coreos.com created
clusterrolebinding.rbac.authorization.k8s.io/prometheus-operator created
clusterrole.rbac.authorization.k8s.io/prometheus-operator created
deployment.apps/prometheus-operator created
serviceaccount/prometheus-operator created
service/prometheus-operator created
Error from server (Invalid): error when creating "https://raw.githubusercontent.com/prometheus-operator/prometheus-operator/mai
ns.k8s.io "prometheusagents.monitoring.coreos.com" is invalid: metadata.annotations: Too long: must have at most 262144 bytes
Error from server (Invalid): error when creating "https://raw.githubusercontent.com/prometheus-operator/prometheus-operator/mai
ns.k8s.io "prometheuses.monitoring.coreos.com" is invalid: metadata.annotations: Too long: must have at most 262144 bytes
```

```
root@kuber3:~# kubectl apply -f https://raw.githubusercontent.com/prometheus-operator/prometheus-operator/r
customresourcedefinition.apiextensions.k8s.io/alertmanagerconfigs.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/alertmanagers.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/podmonitors.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/probes.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/prometheusagents.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/prometheuses.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/prometheusrules.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/scrapeconfigs.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/servicemonitors.monitoring.coreos.com serverside-applied
customresourcedefinition.apiextensions.k8s.io/thanosrulers.monitoring.coreos.com serverside-applied
clusterrolebinding.rbac.authorization.k8s.io/prometheus-operator serverside-applied
clusterrole.rbac.authorization.k8s.io/prometheus-operator serverside-applied
deployment.apps/prometheus-operator serverside-applied
serviceaccount/prometheus-operator serverside-applied
service/prometheus-operator serverside-applied
```

You can see the result:

```
root@kuber3:~# kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
prometheus-operator   1/1     1            1           79s
root@kuber3:~#
```

Then you should create rbac file for it like below:

```
root@kuber3:~# vim prometheus_rbac.yaml
root@kuber3:~#
```

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prometheus
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  - nodes/metrics
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups: [""]
  resources:
  - configmaps
  verbs: ["get"]
```

And apply it :

```
serviceaccount/prometheus created
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
root@kuber3:~#
```

Then you should create instance file for it like below:

```
root@kuber3:~# vim prometheus_instance.yaml
root@kuber3:~#
```

```yaml
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  name: prometheus
spec:
  serviceAccountName: prometheus
  serviceMonitorSelector: {}
  resources:
    requests:
      memory: 300Mi
```

And apply it:

```
root@kuber3:~# kubectl apply -f prometheus_instance.yaml
prometheus.monitoring.coreos.com/prometheus created
root@kuber3:~#
```

And you can see the result:

```
root@kuber3:~# kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
prometheus-operator-6c9b57bcb8-9kg5g   1/1     Running   0          14m
prometheus-prometheus-0                2/2     Running   0          9m13s
root@kuber3:~#
```

Then you should create service file for it like below:

```
root@kuber3:~# vim service_monitor.yaml
root@kuber3:~#
```

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    name: prometheus
  name: prometheus
spec:
  endpoints:
  - interval: 30s
    targetPort: 9090
    path: /metrics
  namespaceSelector:
    any: true
  selector:
    matchLabels:
      operated-prometheus: "true"
```

And apply it:

```
root@kuber3:~# kubectl apply -f service_monitor.yaml
servicemonitor.monitoring.coreos.com/prometheus created
root@kuber3:~#
```

Then you should create expose file for it like below:

```
^Croot@kuber3:~# vim expose_prometheus.yaml
root@kuber3:~#
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: prometheus
spec:
  type: NodePort
  ports:
  - name: web
    nodePort: 30900
    port: 9090
    protocol: TCP
    targetPort: web
  selector:
    prometheus: prometheus
```

And apply it:

```
root@kuber3:~# kubectl apply -f expose_prometheus.yaml
service/prometheus created
root@kuber3:~#
```

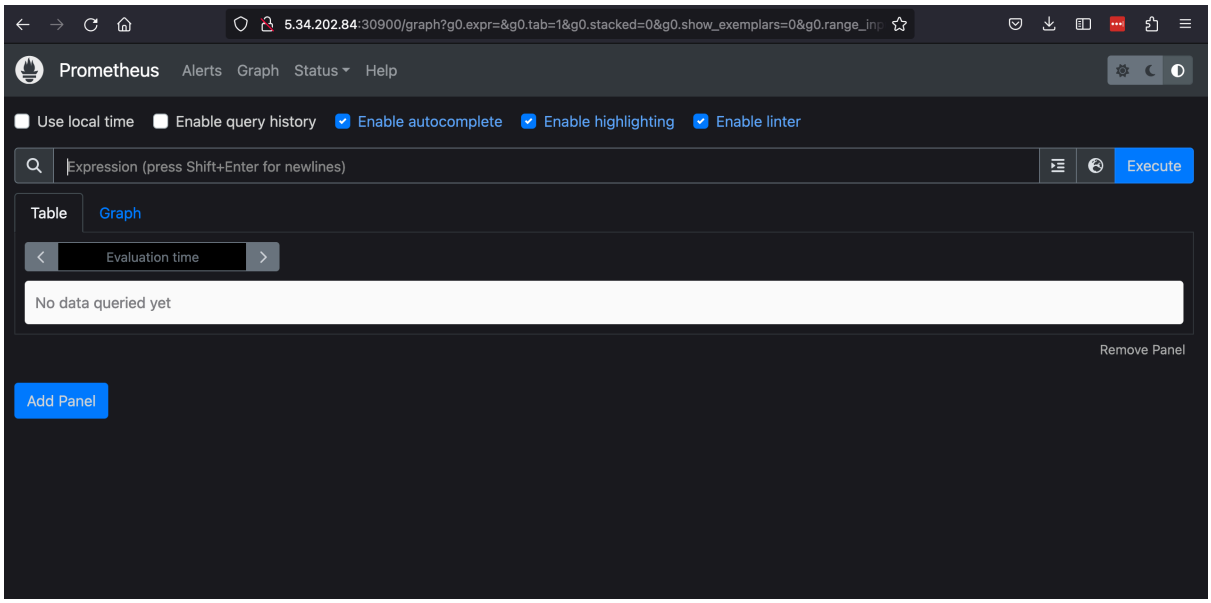
Now you can see the Prometheus panel in this link:

`http://<node_ip>:30900`.

If you want to know your Ip you can run this :

```
root@kuber3:~# kubectl get nodes -o wide
NAME     STATUS     ROLES           AGE   VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE
kuber1   Ready      <none>          15h   v1.28.2   188.121.114.8    <none>        Ubuntu 20.04.6 LT
kuber2   NotReady   <none>          16h   v1.28.2   5.34.203.60      <none>        Ubuntu 20.04.6 LT
kuber3   Ready      control-plane   17h   v1.28.2   5.34.202.84      <none>        Ubuntu 20.04.6 LT
root@kuber3:~#
```

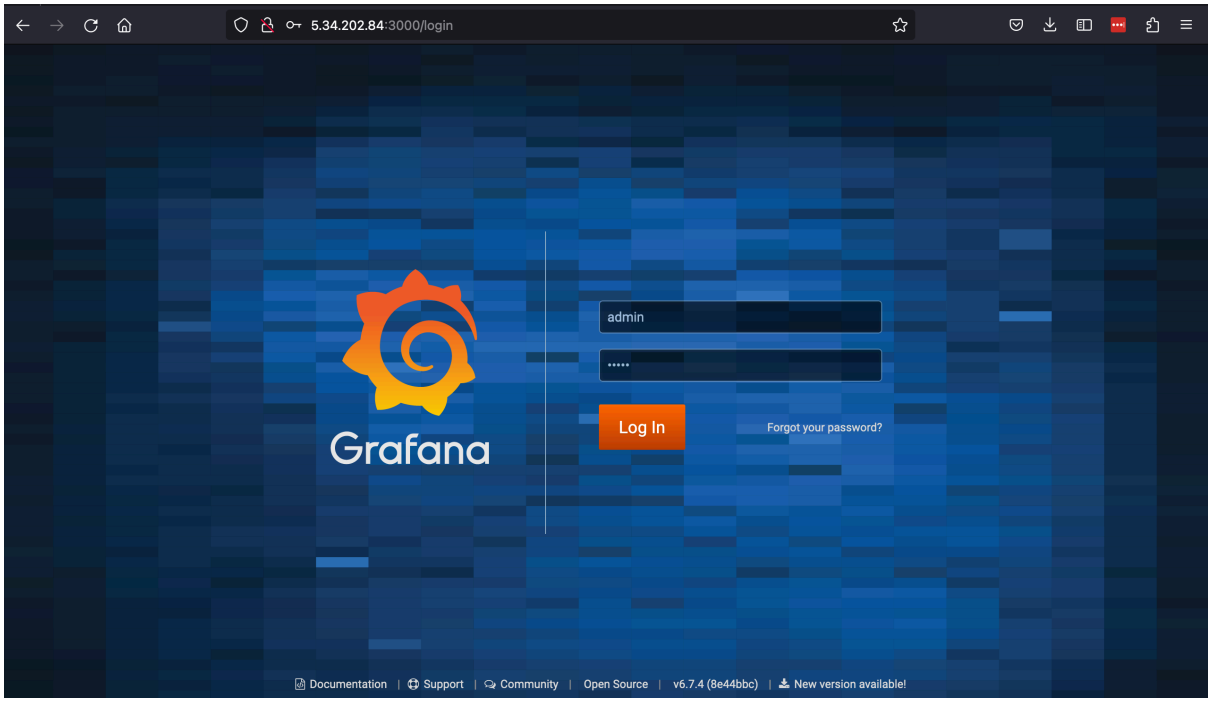
Here is Prometheus panel:



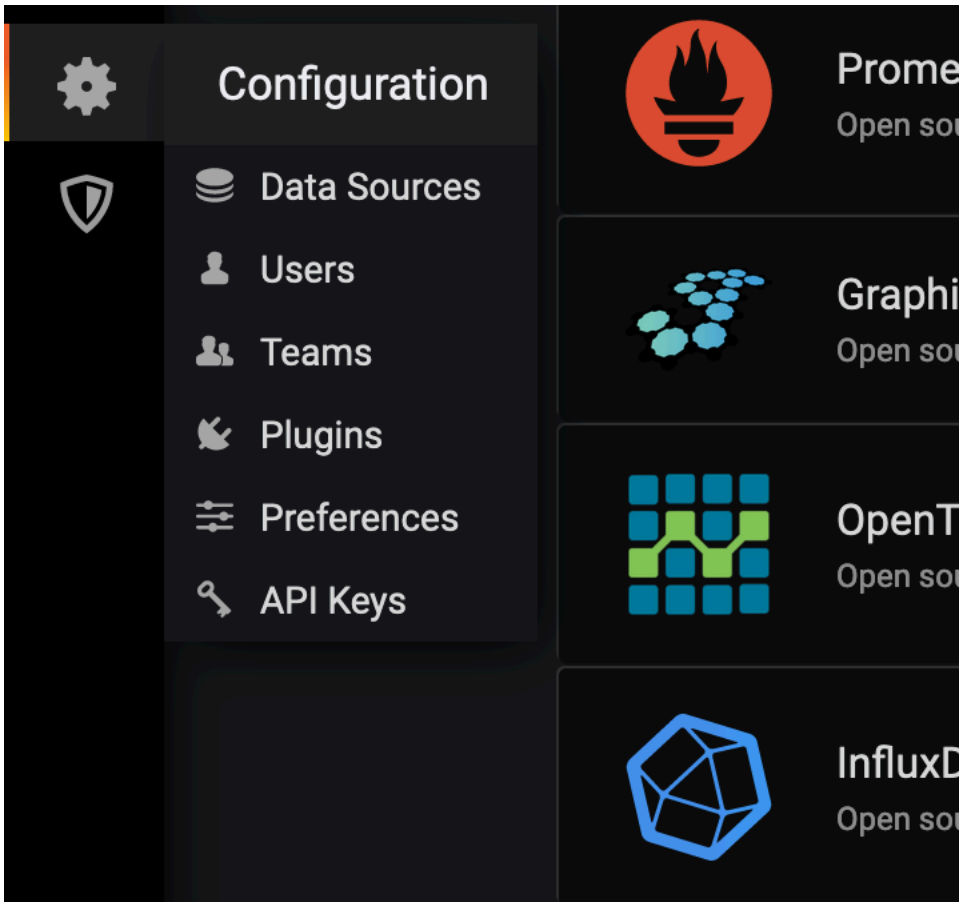
Then I install Grafana by this :

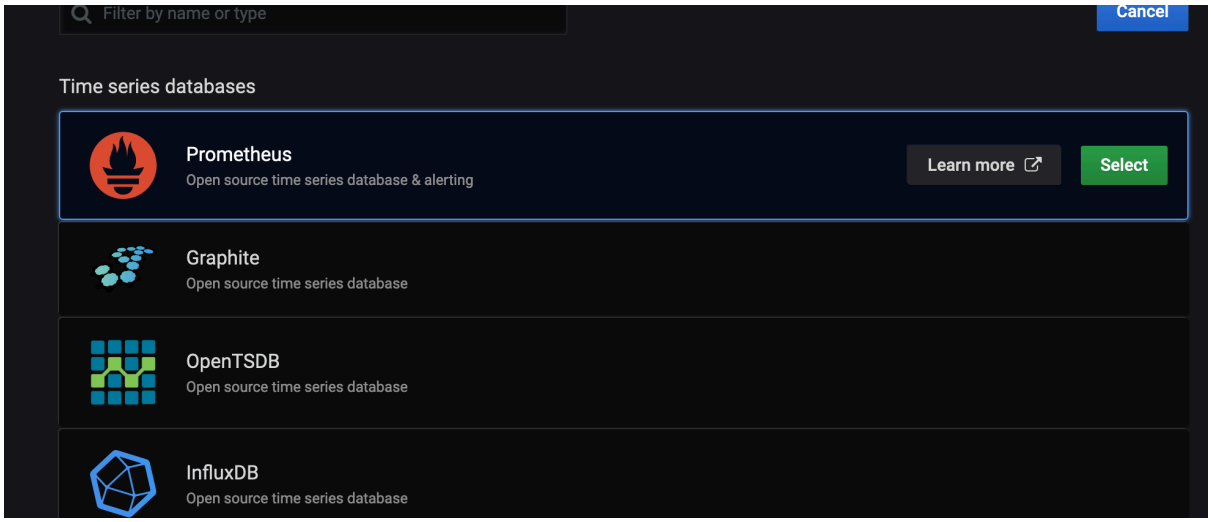
```
root@kuber3:~# sudo snap install grafana
sudo: unable to resolve host kuber3: Name or service not known
grafana 6.7.4 from Canonical✓ installed
```

And you can see the panel below:

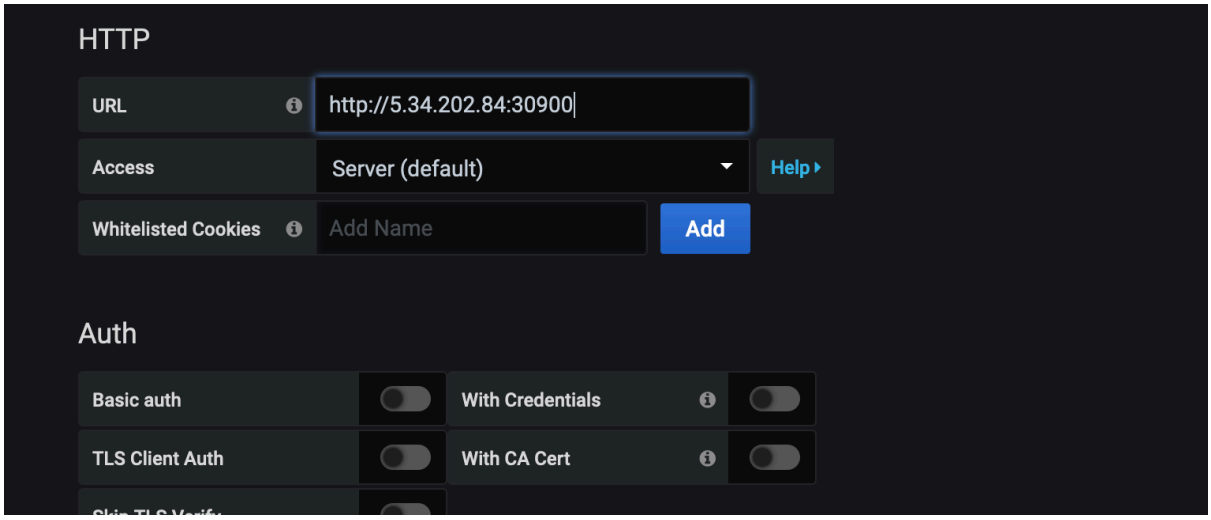


You should add Prometheus data source like this :

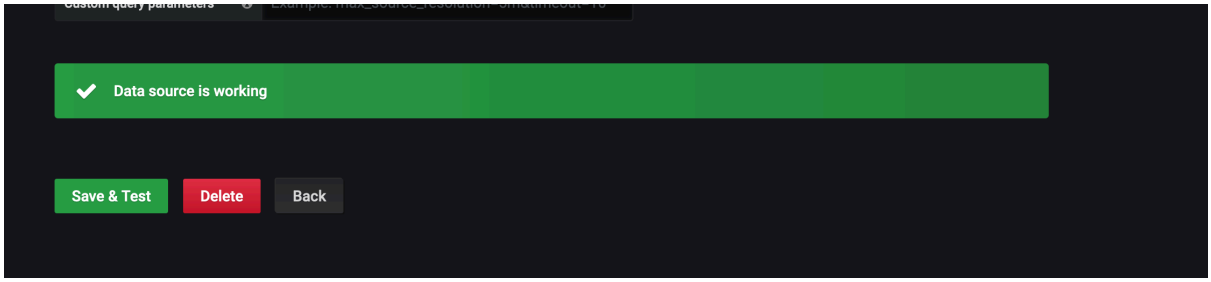




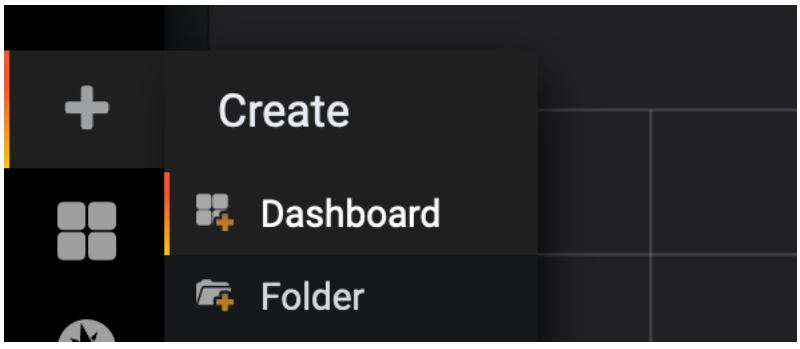
And write your node Ip in url field and port:



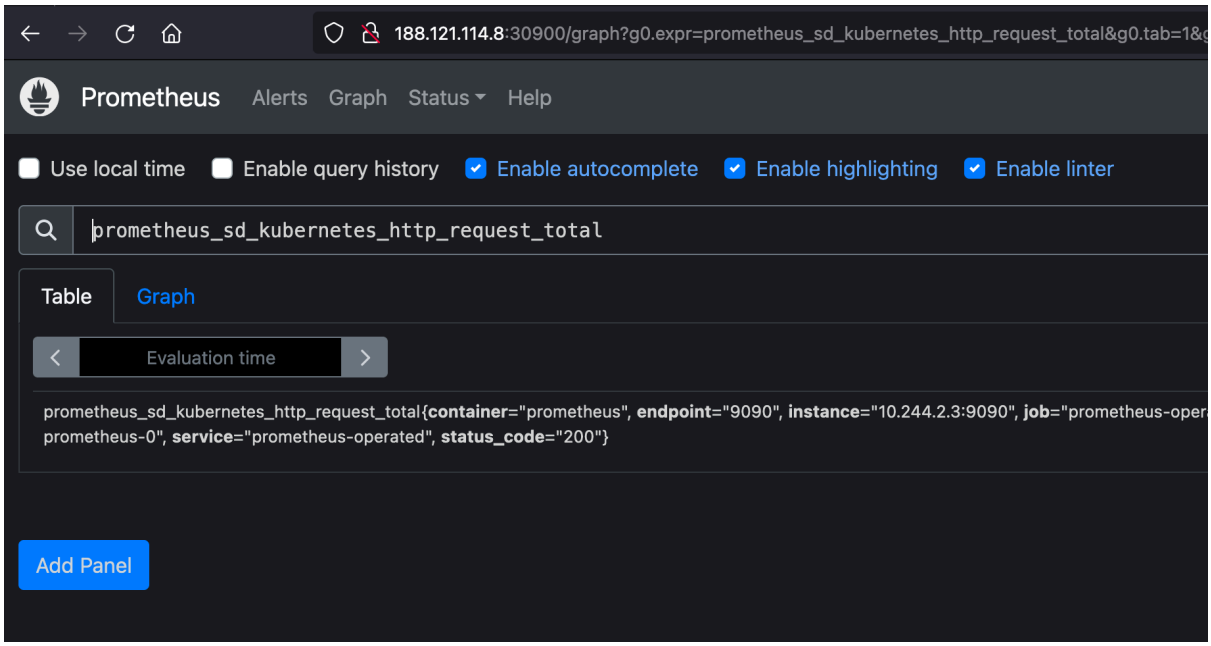
Then click on test:



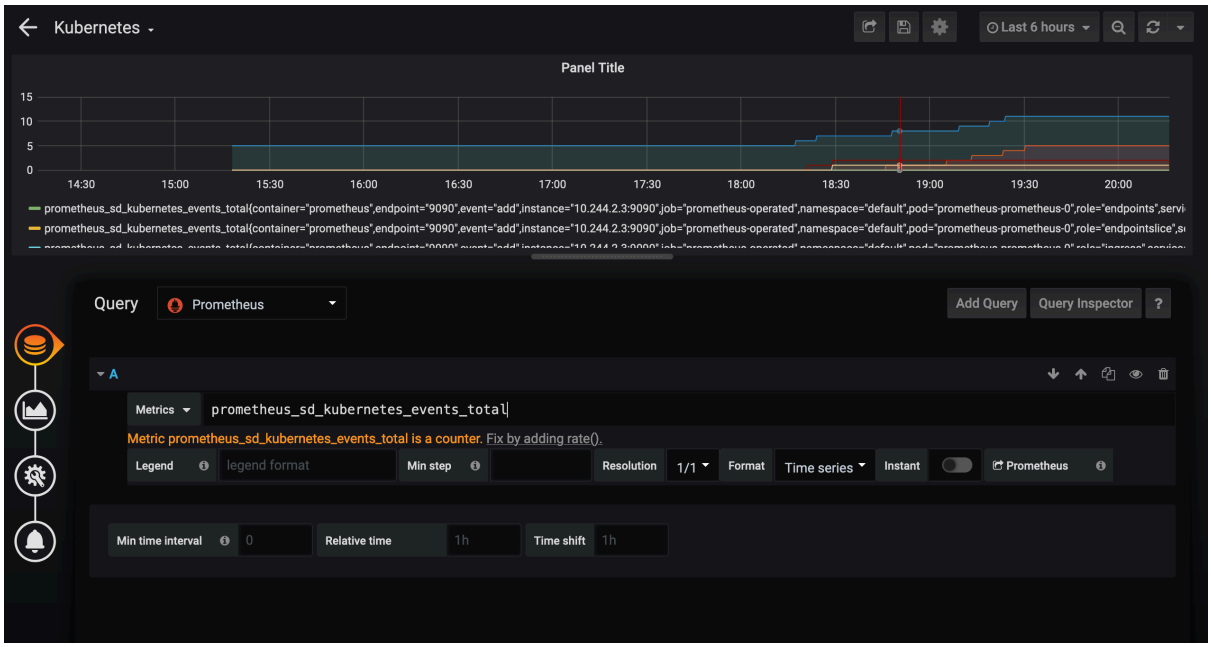
Then I create dashboard like this:



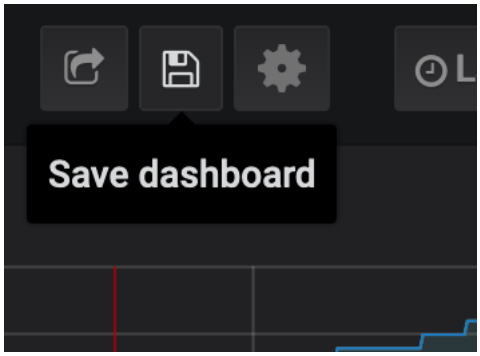
And I copy one metric from Prometheus:



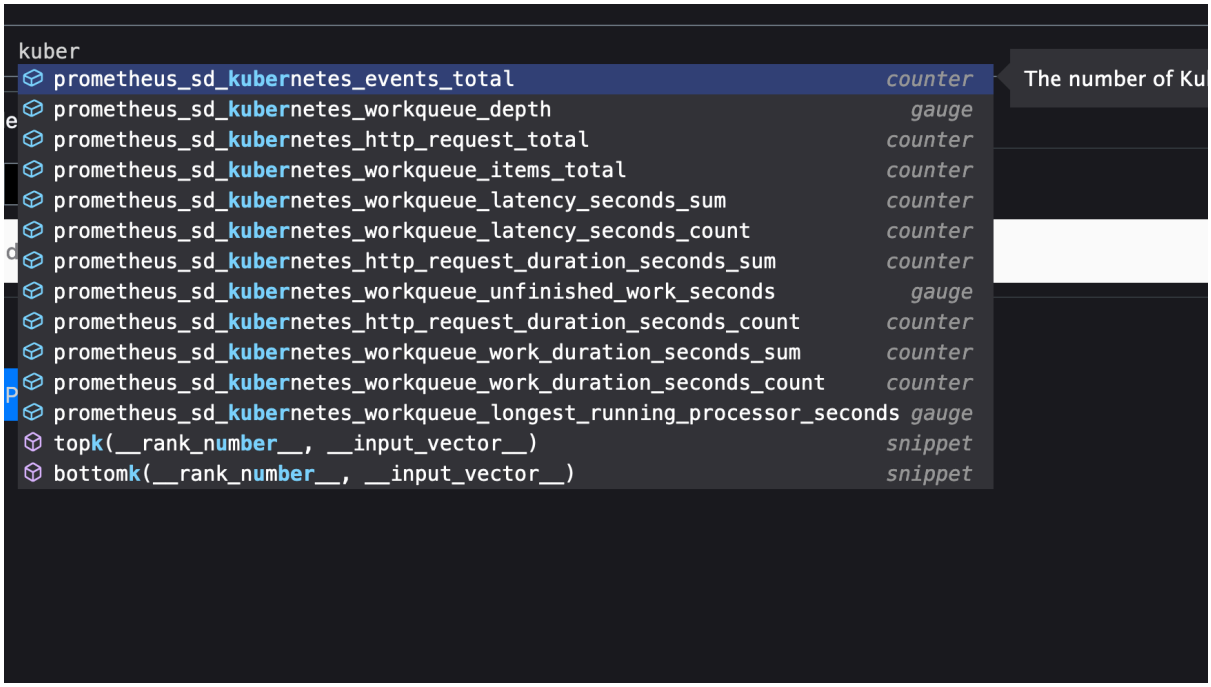
And add it to panel and choose the data source:



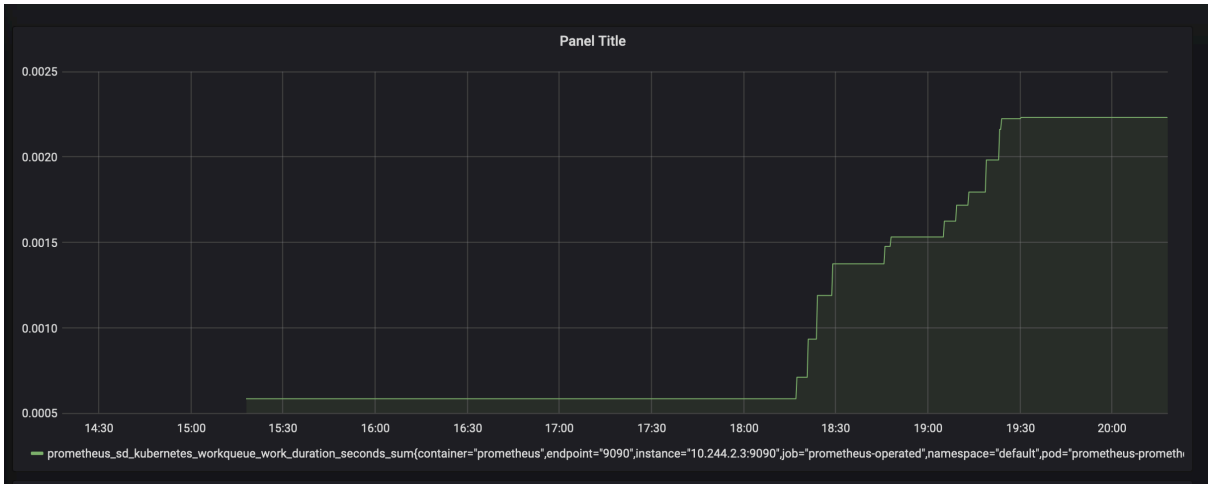
Then I save dashboard and choose name:



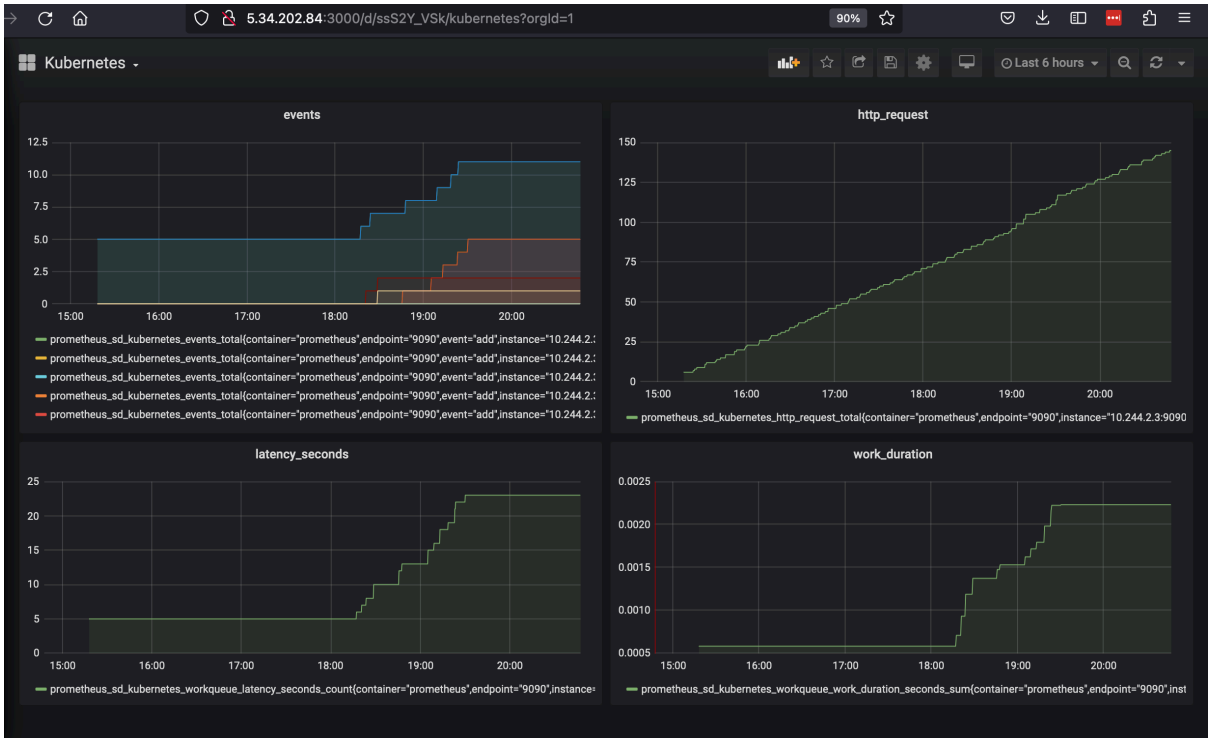
You can add other metrics as well:



For instance I add another parameter to another panel:

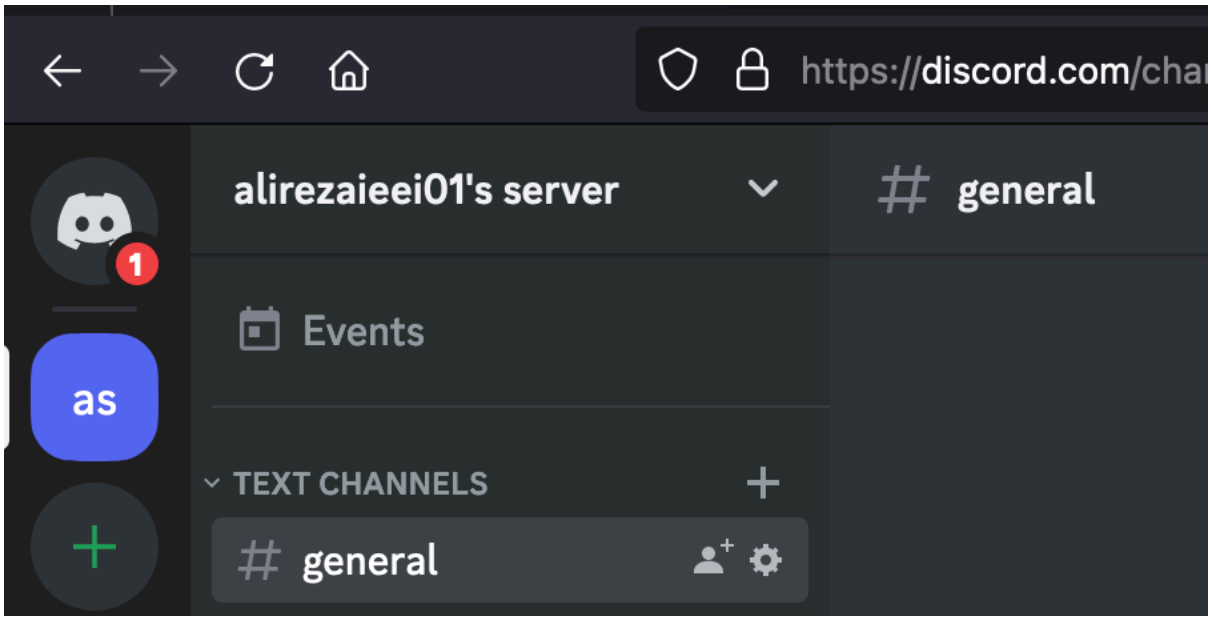


Finally their Kubernetes dashboard is prepared like this :

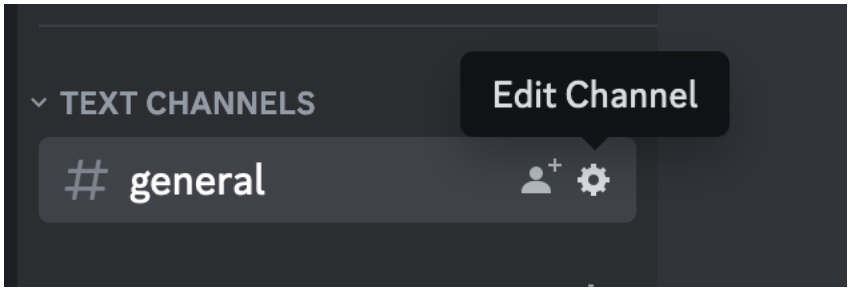


Phase 4(alerting):

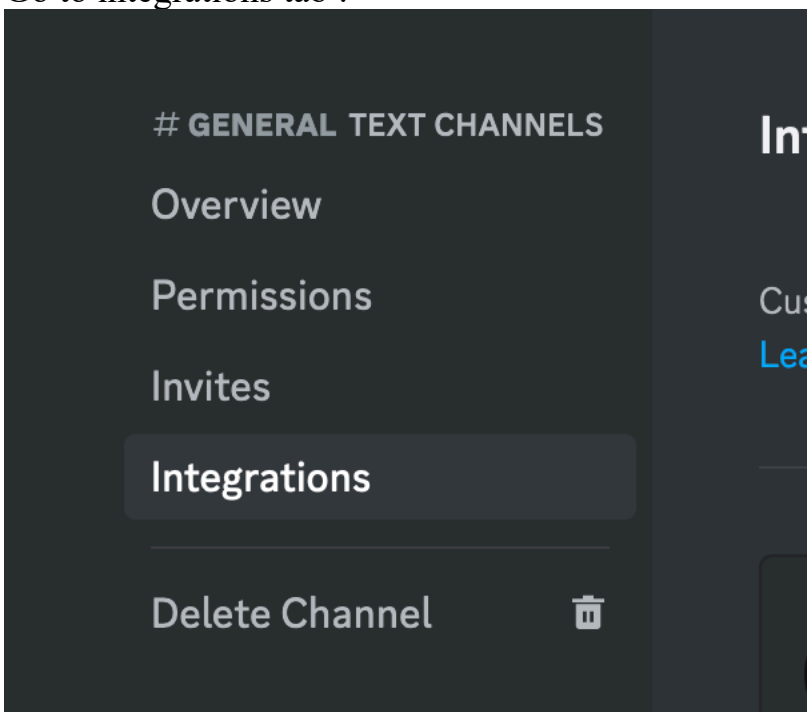
First we should create channel in discord in our own server like this:



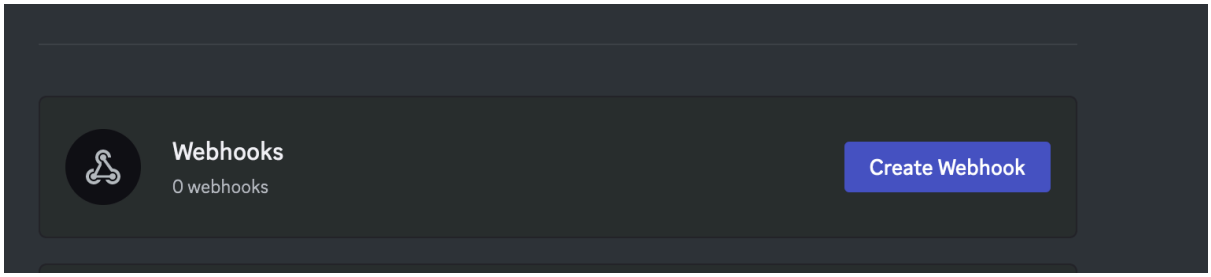
and click on edit channel:



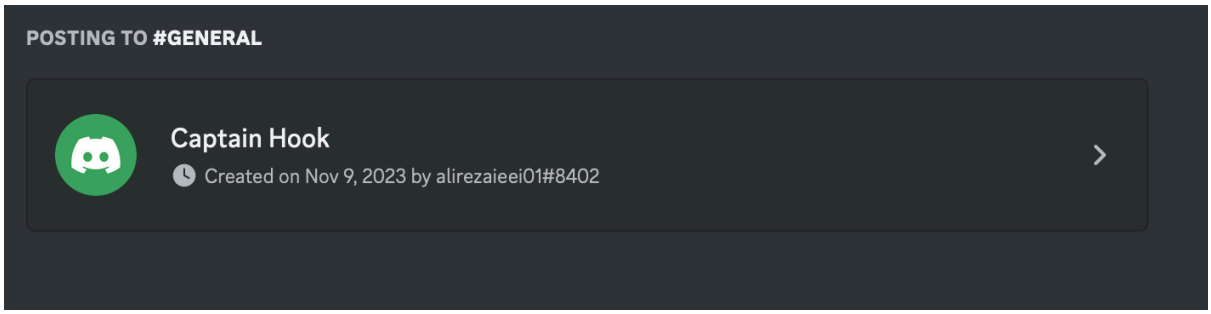
Go to integrations tab :



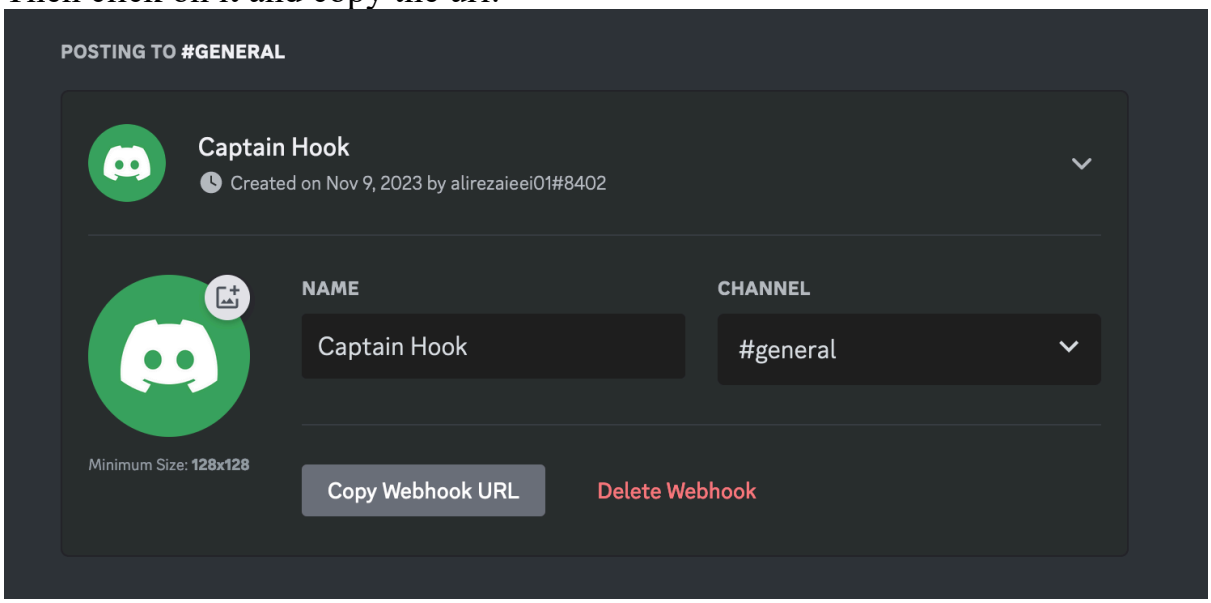
And choose create webhook:



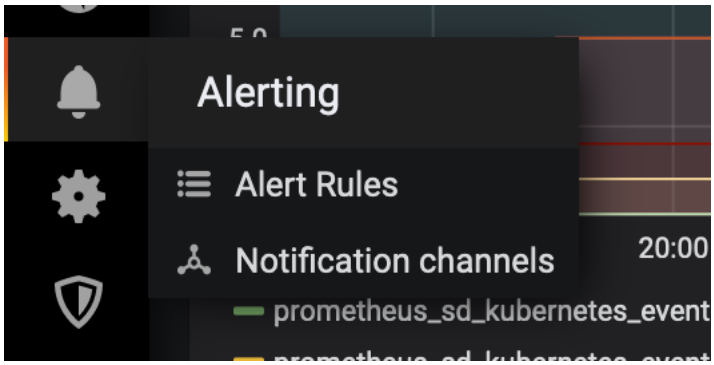
And see that is created:



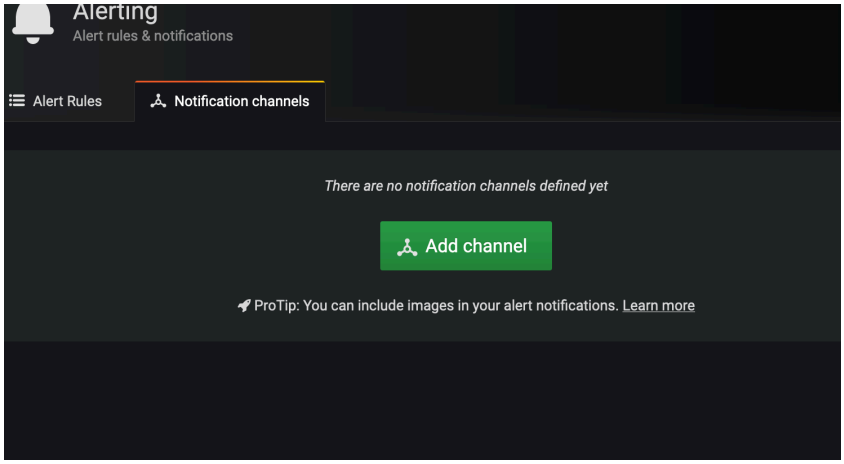
Then click on it and copy the url:



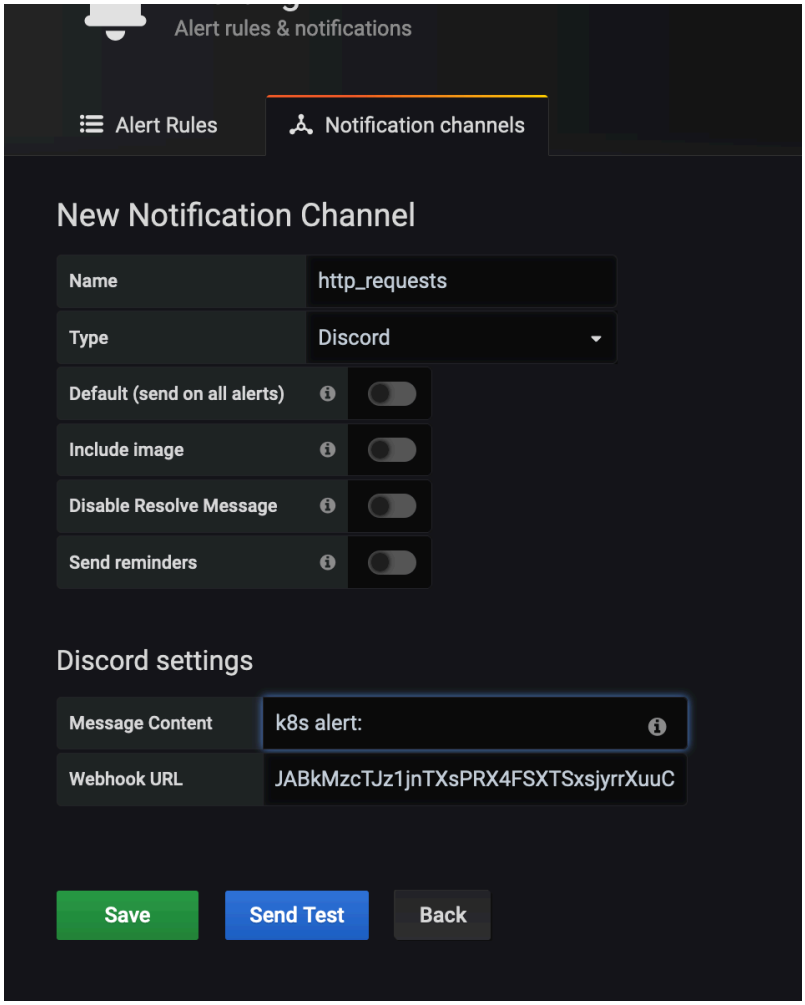
Then go to Grafana and choose notification channels:



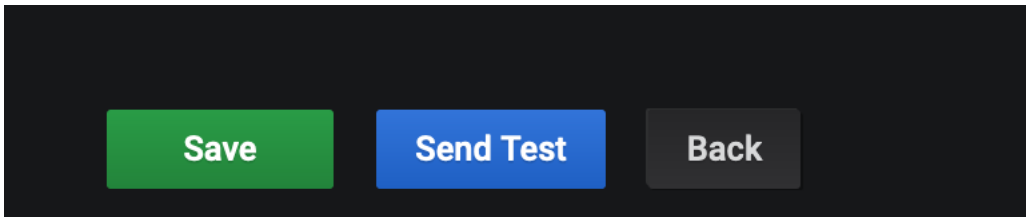
And click on add channel:



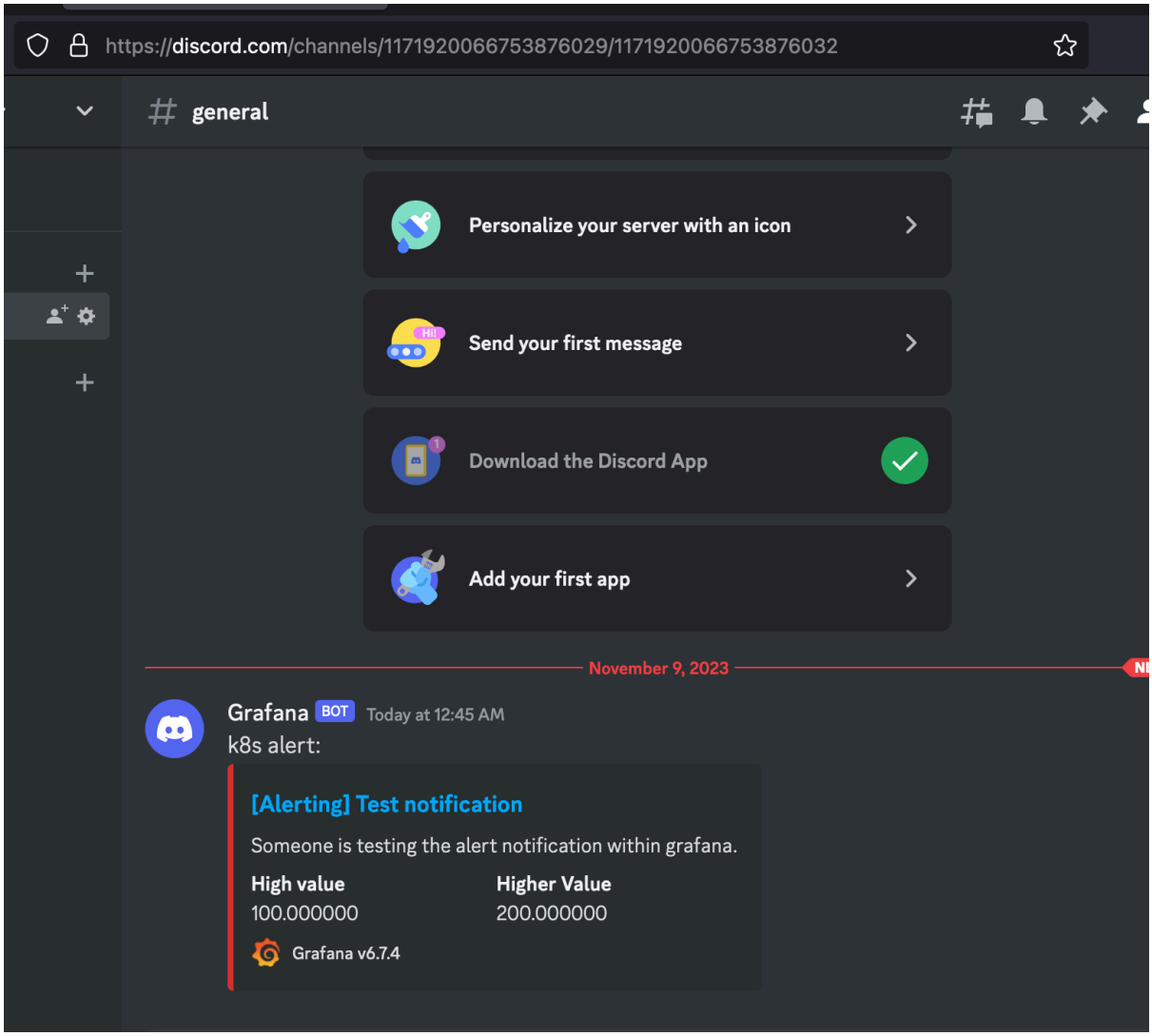
And select discord in type and paste the webhook:



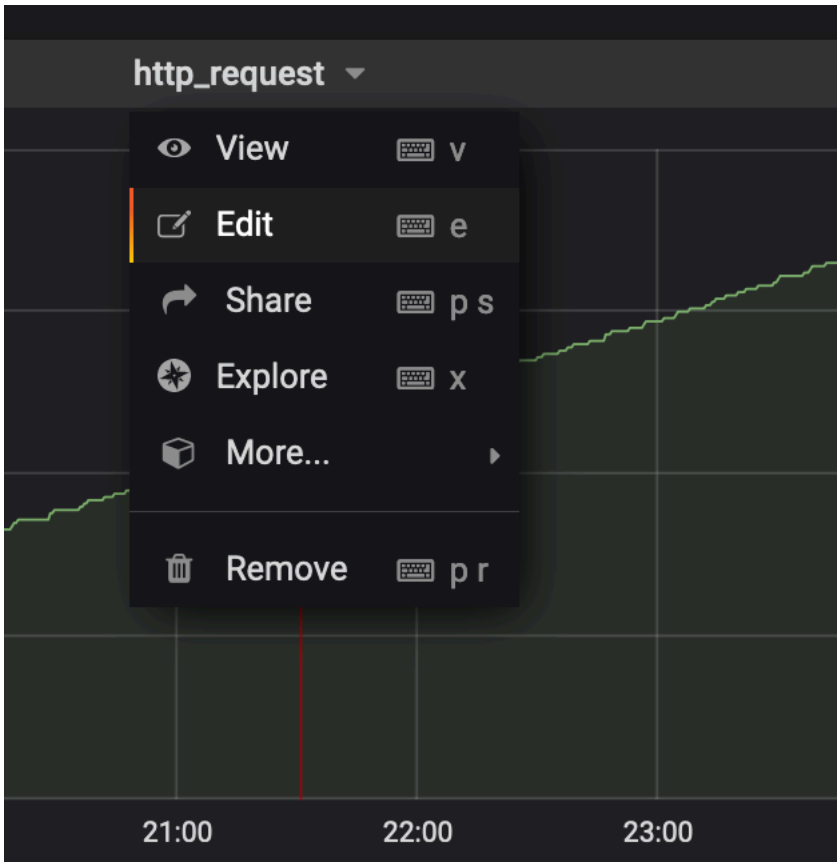
Then you can test it :



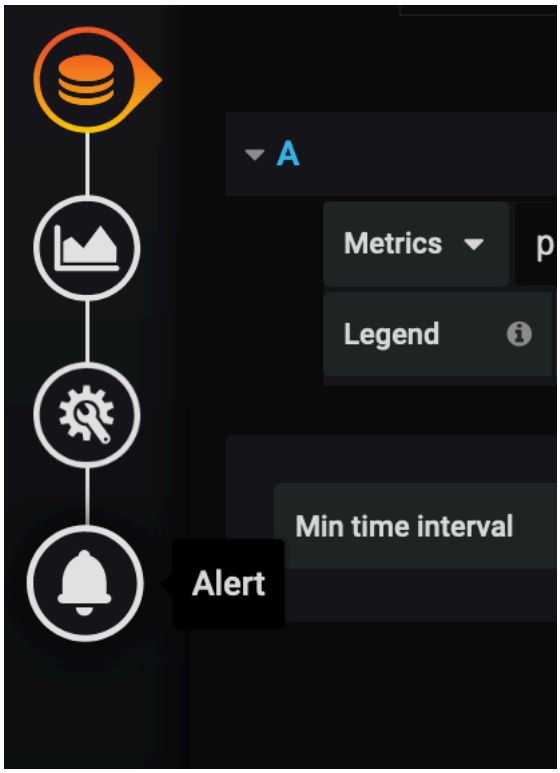
And see the result in channel:



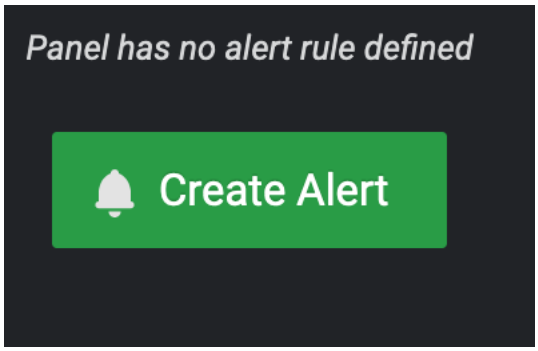
After it we should select the dashboard and click on edit:



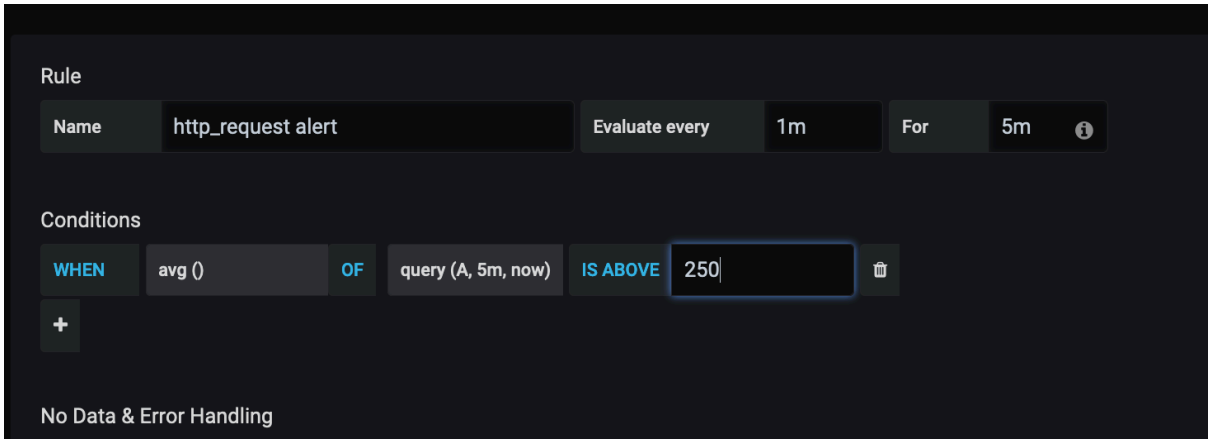
And click on alert:



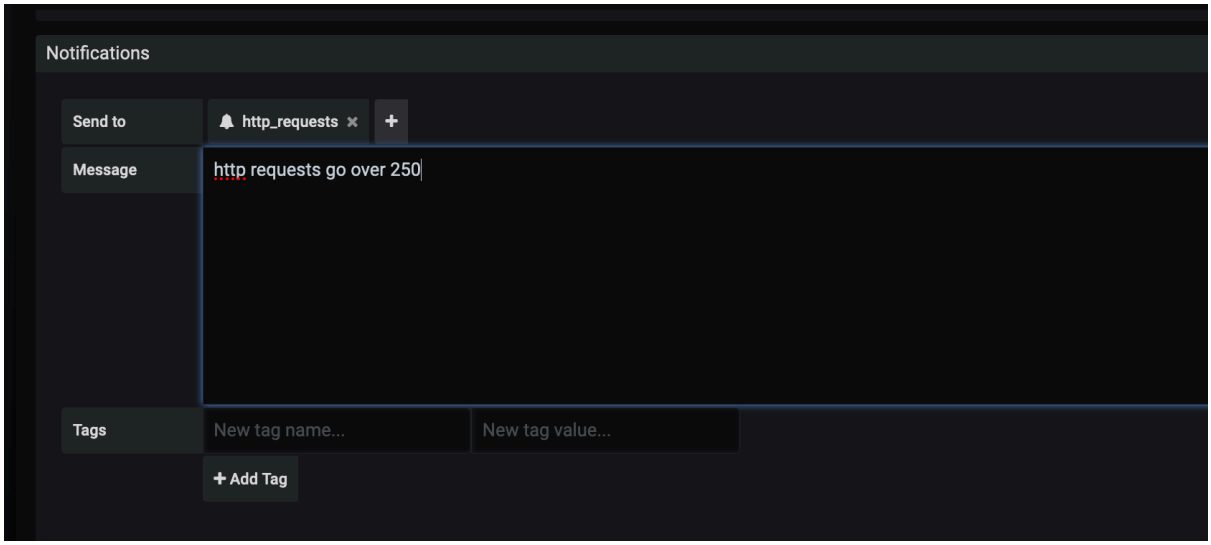
And create alert:



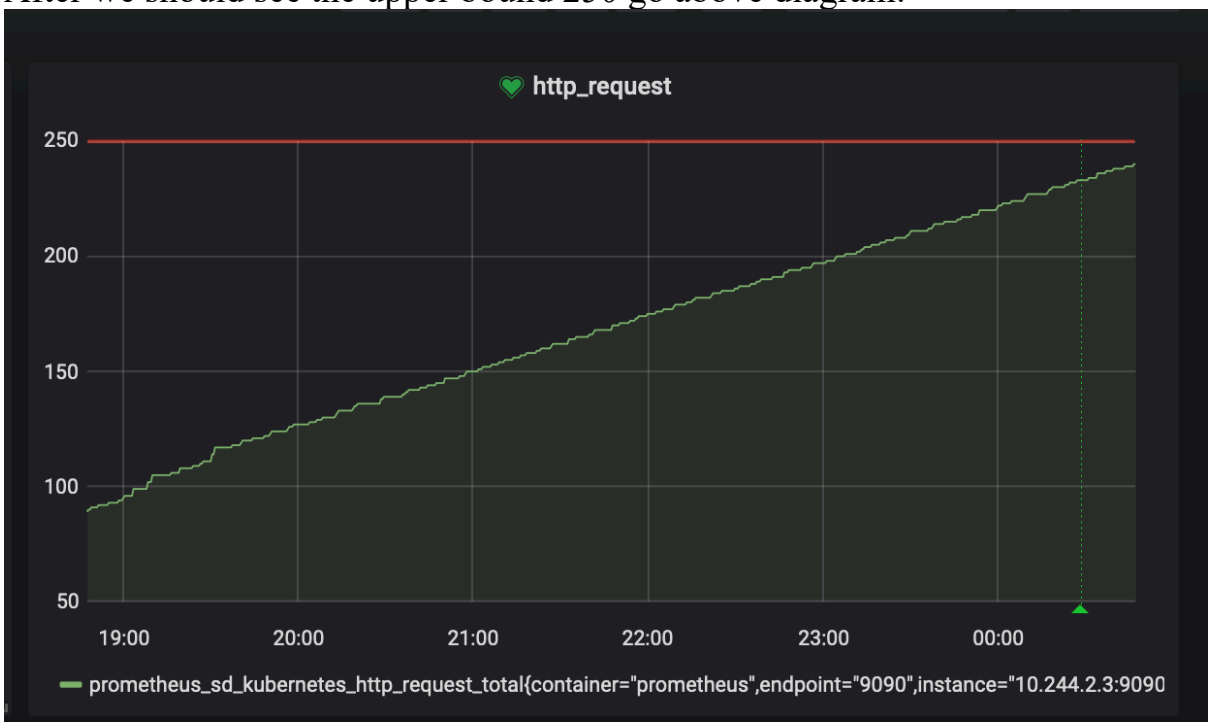
We can set condition like below that is alert when the metric go above 250:



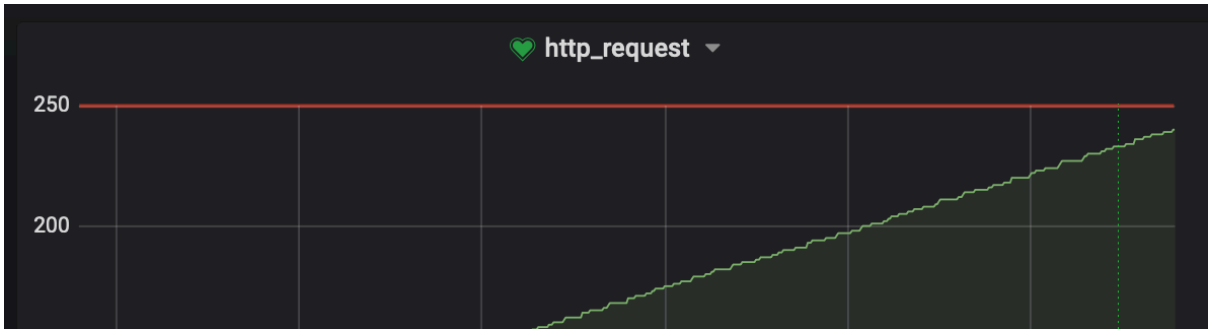
And we can set massage for it :



After we should see the upper bound 250 go above diagram:



And we can check the healthy by the color of the heart that is located on the top of the dashboard:



Phase 5(Postgres cluster):

I use this link:

https://sweetcode.io/how-to-deploy-postgresql-instance-to-kubernetes/

I create db-persistent-volume.yaml.

```
apiVersion: v1
# Kind for volume chain
kind: PersistentVolume
metadata:
  # Name the persistent chain
  name: postgresdb-persistent-volume
  # Labels for identifying PV
  labels:
    type: local
    app: postgresdb
spec:
  storageClassName: manual
  capacity:
    # PV Storage capacity
    storage: 8Gi
  # A db can write and read from volumes to multiple pods
  accessModes:
    - ReadWriteMany
  # Specify the path to persistent the volumes
  hostPath:
    path: "/data/db"
```

And I create db-volume-claim.yaml

```
apiVersion: v1
# define a resource for volume chain
kind: PersistentVolumeClaim
metadata:
  # Name the volume chain
  name: db-persistent-volume-claim
spec:
  storageClassName: manual
  accessModes:
    # Allow ReadWrite to multiple pods
    - ReadWriteMany
  # PVC requesting resources
  resources:
    requests:
      # the PVC storage
      storage: 8Gi
```

And I create db-configmap.yaml

```
apiVersion: v1
# Kind for kubernets ConfigMap
kind: ConfigMap
metadata:
  # Name your ConfigMap
  name: db-secret-credentials
  labels:
    app: postgresdb
data:
  # User DB
  POSTGRES_DB: testDB
  # Db user
  POSTGRES_USER: testUser
  # Db password
  POSTGRES_PASSWORD: testPassword
```

I create db-deployment.yaml:

```yaml
# Kubernetes API version
apiVersion: apps/v1
# Deployment object
kind: Deployment
metadata:
  # The name of the Deployment
  name: postgresdb
spec:
  # Replicas for this Deployment
  replicas: 3
  selector:
    # labels the pods
    matchLabels:
      app: postgresdb
  template:
    metadata:
      labels:
        # The label the pods created from the pod template sho
        app: postgresdb
    spec:
      containers:
        # The container name to execute pods
        - name: postgresdb
          # pull postgresimage from docker hub
          image: postgres
          ports:
            # Assign ports to expose container
            - containerPort: 5432
          envFrom:
            # Load the environment variables/PostgresSQL crede
            - configMapRef:
                # This should be the ConfigMap name created ea
                name: db-secret-credentials
          volumeMounts:
            # The volume mounts  for the container
            - mountPath: /var/lib/postgres/data
              name: db-data
```

I create db-service.yaml

```
apiVersion: v1
# Kind for service
kind: Service
metadata:
  # Name your service
  name: postgresdb
  labels:
    app: postgresdb
spec:
  # Choose how to expose your se
  type: NodePort
  ports:
    # The port number to expose
    - port: 5432
  # Pod to route service traffic
  selector:
    app: postgresdb
```

And apply all of them with this order:

```
[root@kuber3:~# kubectl apply -f db-persistent-volume.yaml
persistentvolume/postgresdb-persistent-volume created
[root@kuber3:~# kubectl apply -f db-volume-claim.yaml
persistentvolumeclaim/db-persistent-volume-claim created
[root@kuber3:~# kubectl apply -f db-configmap.yaml
configmap/db-secret-credentials created
[root@kuber3:~# kubectl apply -f db-deployment.yaml
deployment.apps/postgresdb created
[root@kuber3:~# kubectl apply -f db-service.yaml
service/postgresdb created
```

And you can see this in result:

```
root@kuber3:~# kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/postgresdb-5cf94bf6fb-55tpf         1/1     Running   0          12m
pod/postgresdb-5cf94bf6fb-9hmq9         1/1     Running   0          12m
pod/postgresdb-5cf94bf6fb-xzh6s         1/1     Running   0          12m
pod/prometheus-operator-6c9b57bcb8-9kg5g   1/1     Running   0          11h
pod/prometheus-prometheus-0             2/2     Running   0          11h

NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/kubernetes            ClusterIP   10.96.0.1        <none>        443/TCP          24h
service/postgresdb            NodePort    10.109.251.47    <none>        5432:31376/TCP   12m
service/prometheus            NodePort    10.104.145.106   <none>        9090:30900/TCP   7h33m
service/prometheus-operated   ClusterIP   None             <none>        9090/TCP         11h
service/prometheus-operator   ClusterIP   None             <none>        8080/TCP         11h

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgresdb            3/3     3            3           12m
deployment.apps/prometheus-operator   1/1     1            1           11h

NAME                                             DESIRED   CURRENT   READY   AGE
replicaset.apps/postgresdb-5cf94bf6fb            3         3         3       12m
replicaset.apps/prometheus-operator-6c9b57bcb8   1         1         1       11h

NAME                                    READY   AGE
statefulset.apps/prometheus-prometheus   1/1     11h
root@kuber3:~#
```

And you can connect to it with this command by copy one pod id:

```
root@kuber3:~# kubectl exec -it postgresdb-5b9bf77c46-6xhx2 -- psql -h localhost -U testUser --password -p 5432 testDB
Error from server (NotFound): pods "postgresdb-5b9bf77c46-6xhx2" not found
root@kuber3:~# kubectl exec -it postgresdb-5cf94bf6fb-55tpf -- psql -h localhost -U testUser --password -p 5432 testDB
Password:
```

The password is : POSTGRES_PASSWORD

And see the result :

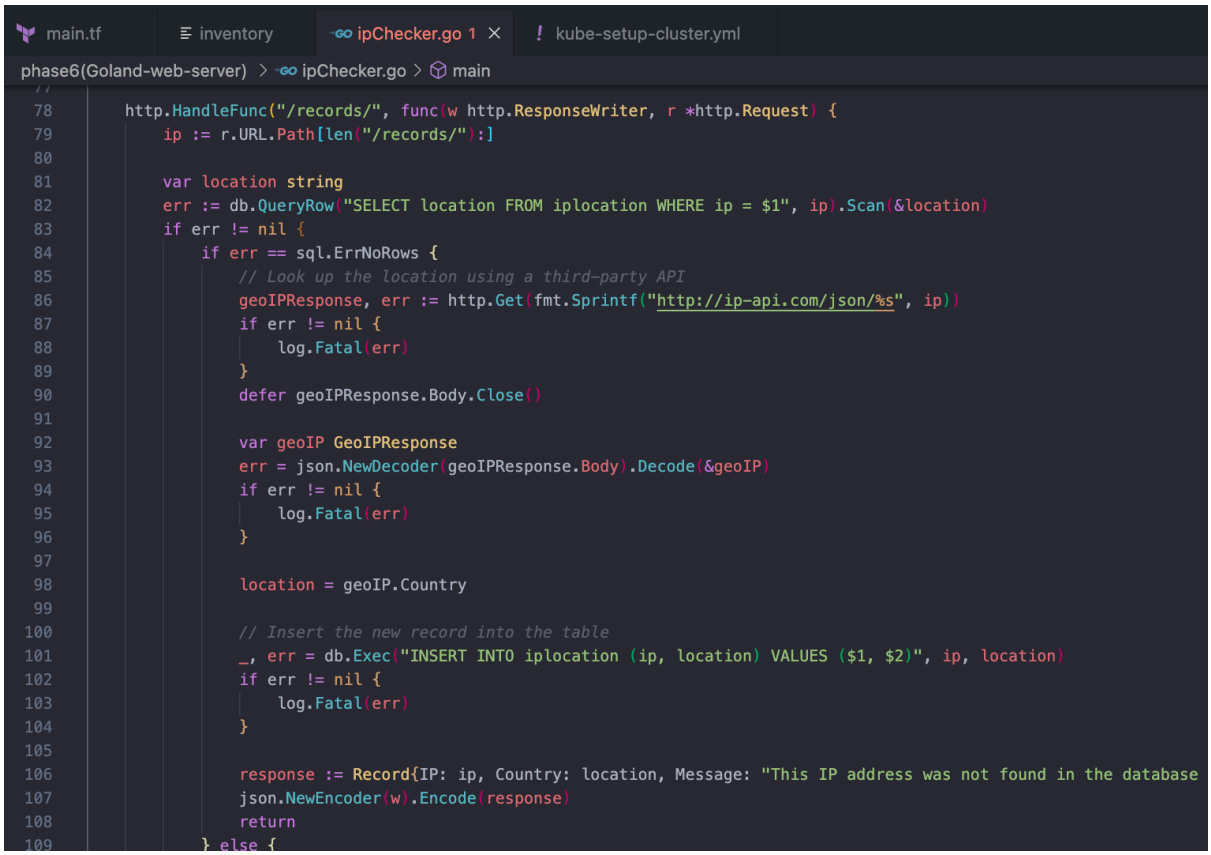
```
root@kuber3:~# kubectl exec -it postgresdb-5cf94bf6fb-55tpf -- psql -h localhost -U testUser --password -p 5432 testDB
Password:
psql (16.0 (Debian 16.0-1.pgdg120+1))
Type "help" for help.

testDB=#
```

Phase 6(Golang web api):

In this code first the server is connected to pgdb by reading the os variable، and then if the table does not exist create the table “iplocation” with columns: ip and location. After this I implement two api that first one return the number of ip that is exist in our table because I want to use it for monitoring and the second one return the country where the ip is from، but first check that it is exist in our database or not and send the massage it exist. Lets see the part of code that you can see all of it in phase 6:

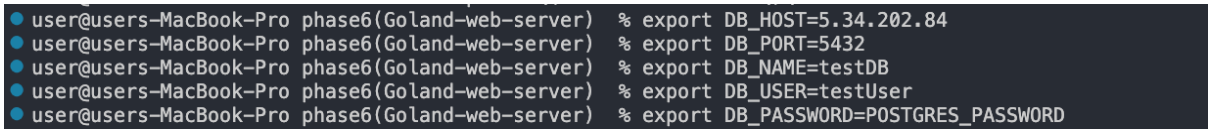
I use this website to give the location of ip :

https://members.ip-api.com



How can we run it:

Firstly، we should set the os variable like this :



Then we should create the go.mod and go.sum and get the pq library:

```
user@users-MacBook-Pro phase6(Goland-web-server)  % go mod init ipchecker
go: creating new go.mod: module ipchecker
go: to add module requirements and sums:
        go mod tidy
user@users-MacBook-Pro phase6(Goland-web-server)  % go get github.com/lib/pq
go: added github.com/lib/pq v1.10.9
```

And we can run it now:

```
user@users-MacBook-Pro phase6(Goland-web-server)  % go run ipChecker.go
2023/11/10 12:59:10 read tcp 192.168.149.66:50103->5.34.202.84:5432: read: connection reset by peer
exit status 1
```

The problem is for the wrong port that we can see the right port mapping by the this command that is 31376:

```
root@kuber3:~# kubectl get services
NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1        <none>        443/TCP          2d11h
postgresdb            NodePort    10.109.251.47    <none>        5432:31376/TCP   35h
prometheus            NodePort    10.104.145.106   <none>        9090:30900/TCP   42h
prometheus-operated   ClusterIP   None             <none>        9090/TCP         46h
prometheus-operator   ClusterIP   None             <none>        8080/TCP         46h
root@kuber3:~#
```

Then I should change the os variable again:

```
user@users-MacBook-Pro phase6(Goland-web-server)  % export DB_PORT=31376
```

Now I see this :

```
user@users-MacBook-Pro phase6(Goland-web-server)  % go run ipChecker.go
2023/11/10 13:39:03 pq: password authentication failed for user "testUser"
exit status 1
```

So I connect to pq that is located in pod by this command :

```
root@kuber3:~# kubectl exec -it postgresdb-5b9bf77c46-6xhx2 -- psql -h localhost -U testUser --password -p 5432 testDB
Error from server (NotFound): pods "postgresdb-5b9bf77c46-6xhx2" not found
root@kuber3:~# kubectl exec -it postgresdb-5cf94bf6fb-55tpf -- psql -h localhost -U testUser --password -p 5432 testDB
Password:
```

The password is: POSTGRES_PASSWORD

```
root@kuber3:~# kubectl exec -it postgresdb-5cf94bf6fb-55tpf -- psql -h localhost -U testUser --password -p 5432 testDB
Password:
psql (16.0 (Debian 16.0-1.pgdg120+1))
Type "help" for help.

testDB=#
```

And create new user:

```
testDB=# create user postgres with superuser password 'postgres';
CREATE ROLE
testDB=#
```

And change the var again:

```
user@users-MacBook-Pro phase6(Goland-web-server)  % export DB_PASSWORD=postgres
user@users-MacBook-Pro phase6(Goland-web-server)  % export DB_USER=postgres
```

Then it return this:

```
user@users-MacBook-Pro phase6(Goland-web-server)  % go run ipChecker.go
2023/11/10 13:02:34 pq: SSL is not enabled on the server
exit status 1
user@users-MacBook-Pro phase6(Goland-web-server)  %
```

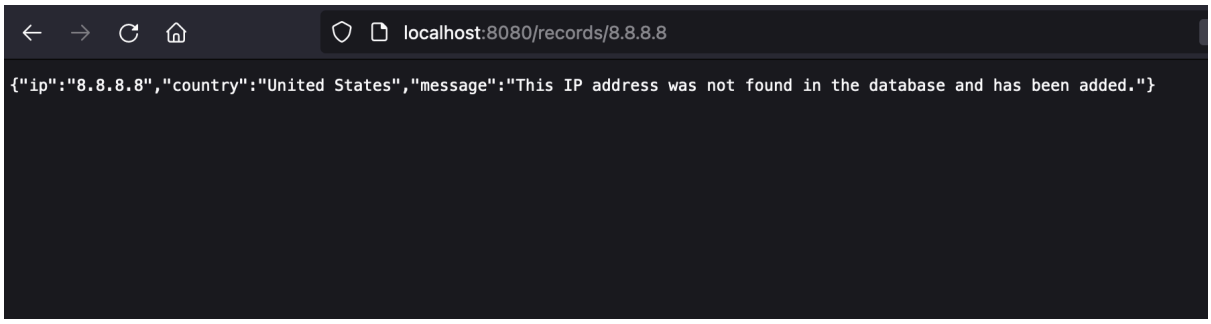
So I can disable ssl like this in code:

```
db, err := sql.Open("postgres", dbURI+"?sslmode=disable")
```

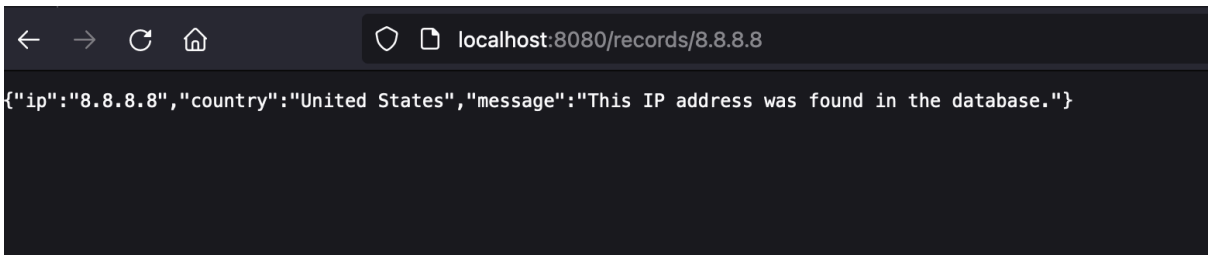
Now you can run it:

```
user@users-MacBook-Pro phase6(Goland-web-server)  % go run ipChecker.go
```

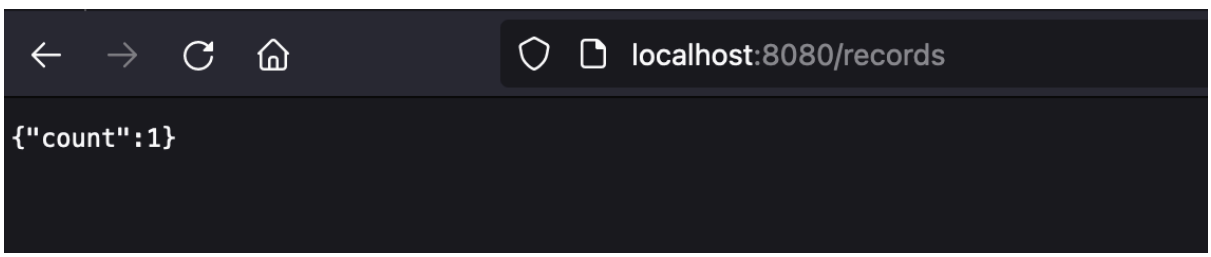
And see the result if the ip doesn't exist:



And see the result if the ip exist:



And the number of entries that we will use it for monitoring in future:

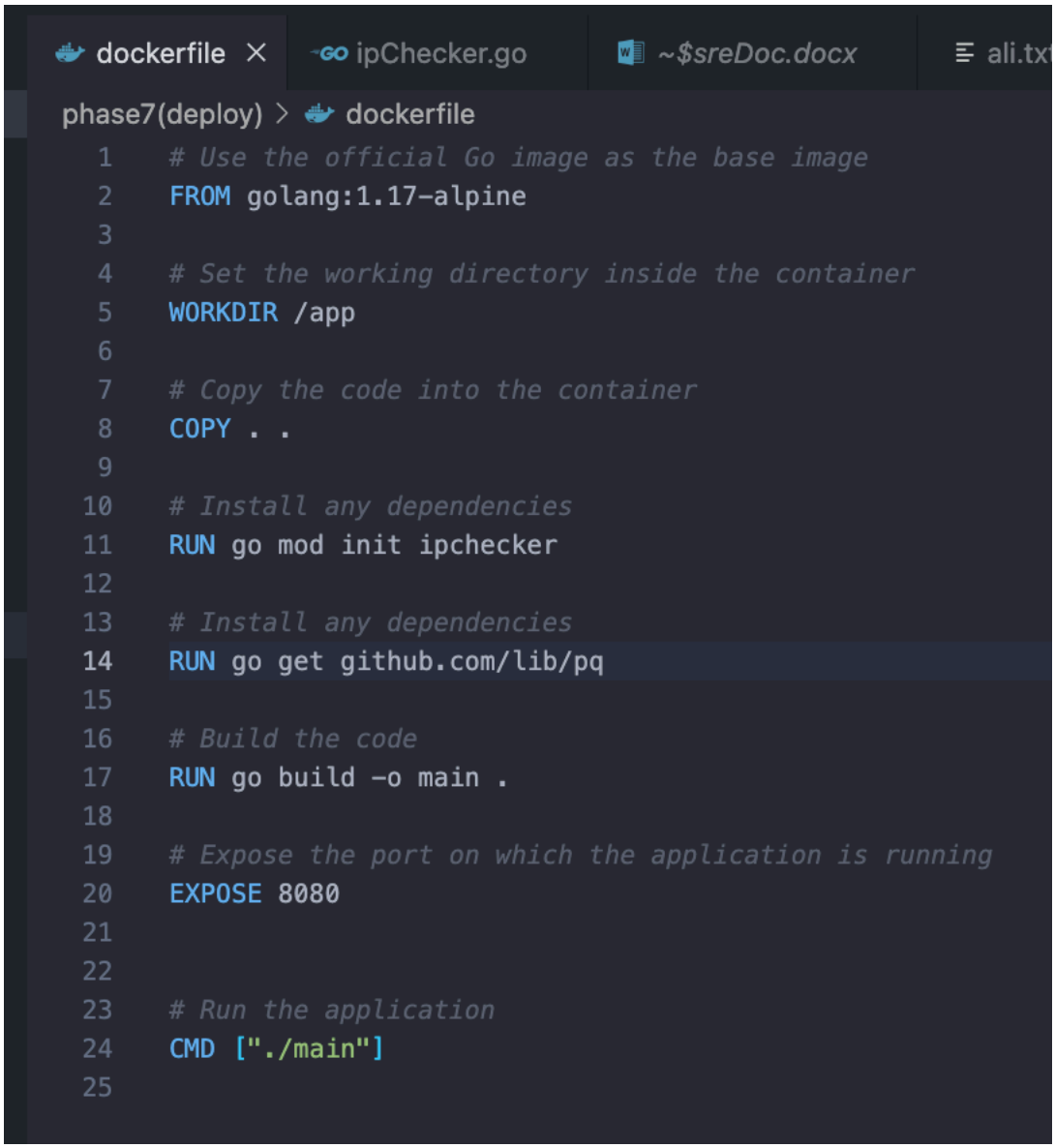


Phase 7(deploy):

First you should set shekan on your pc and creat dockerhub account like this:

https://hub.docker.com/repositories/alirezaieei

then create docker file for your server like this:



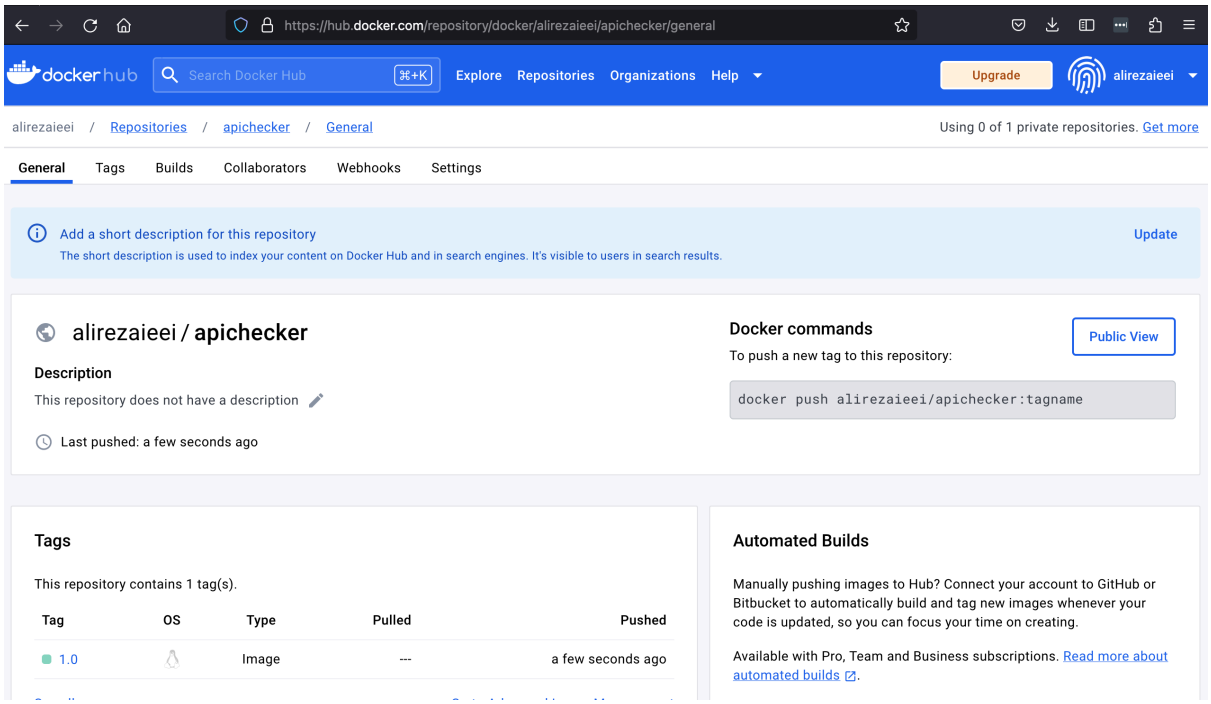
And build it :

```
user@users-MacBook-Pro phase7(deploy) % docker build -t alirezaieei/apichecker:1.0 .
[+] Building 10.5s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                0.0s
 => => transferring dockerfile: 37B                                                 0.0s
 => [internal] load .dockerignore                                                   0.0s
 => => transferring context: 2B                                                     0.0s
 => [internal] load metadata for docker.io/library/golang:1.17-alpine               1.0s
 => [1/6] FROM docker.io/library/golang:1.17-alpine@sha256:99ddec1bbfd6d6bca3f9804c02363daee8c8524dae50d  0.0s
 => [internal] load build context                                                   0.0s
 => => transferring context: 507B                                                   0.0s
 => CACHED [2/6] WORKDIR /app                                                       0.0s
 => [3/6] COPY . .                                                                  0.0s
 => [4/6] RUN go mod init ipchecker                                                 0.3s
 => [5/6] RUN go get github.com/lib/pq                                              7.3s
 => [6/6] RUN go build -o main .                                                    1.3s
 => exporting to image                                                              0.2s
 => => exporting layers                                                             0.2s
 => => writing image sha256:bc8c97cb967eaecdf5f3d378461e91b460cca53048ab98853ec944f1b28bc529  0.0s
 => => naming to docker.io/alirezaieei/apichecker:1.0                               0.0s
user@users-MacBook-Pro phase7(deploy) %
```

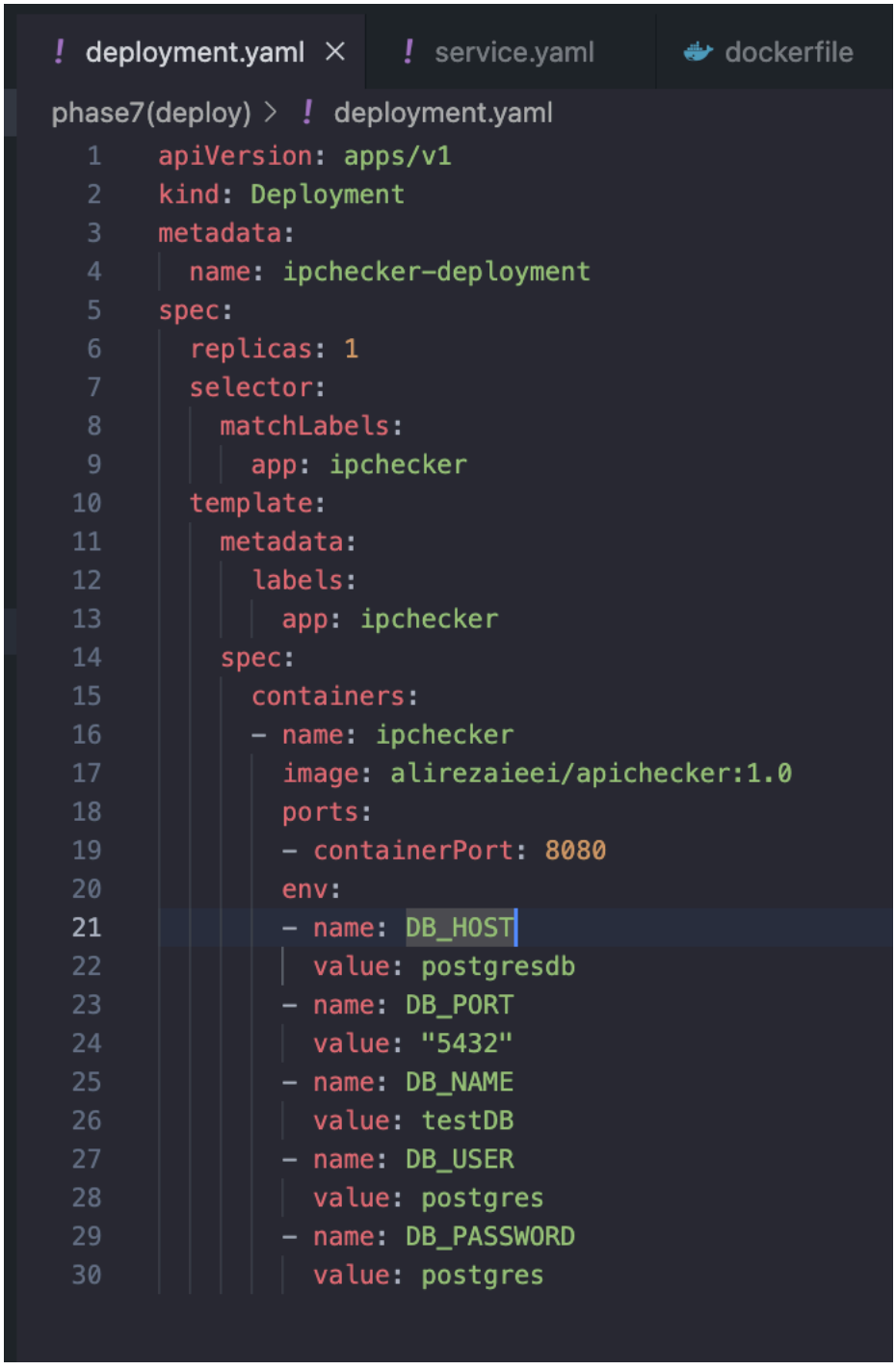
And push it on dockerhub:

```
user@users-MacBook-Pro phase7(deploy) % docker image push alirezaieei/apichecker:1.0
The push refers to repository [docker.io/alirezaieei/apichecker]
7748b6426fa7: Layer already exists
b1d99ecea5e0: Layer already exists
3e9436f8d89f: Layer already exists
c4ad3b140634: Layer already exists
b8de6377084f: Layer already exists
fed9565c145c: Layer already exists
60af18e613ad: Layer already exists
925e9633fd86: Layer already exists
7b35f2def65d: Pushed
ec34fcc1d526: Layer already exists
1.0: digest: sha256:622f5c24406ee4eb6bd343cca4b874901ba38c1f6f2391f30996ba1b0217ffb0 size: 2408
user@users-MacBook-Pro phase7(deploy) %
```

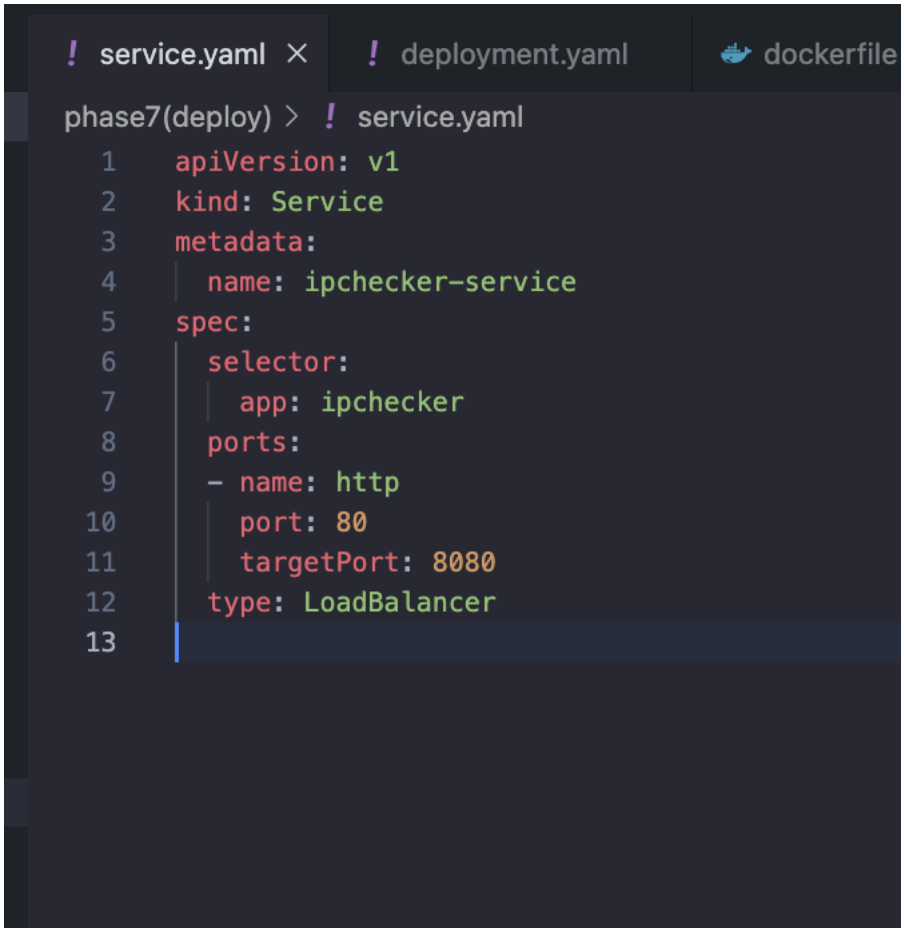
And see the result:



Then create deployment file:



And service file :



And apply them in server by kubectl :

```
root@kuber3:~# vim deployment.yaml
root@kuber3:~# vim service.yaml
root@kuber3:~# kubectl apply -f deployment.yaml
deployment.apps/ipchecker-deployment created
root@kuber3:~# kubectl apply -f service.yaml
service/ipchecker-service created
```

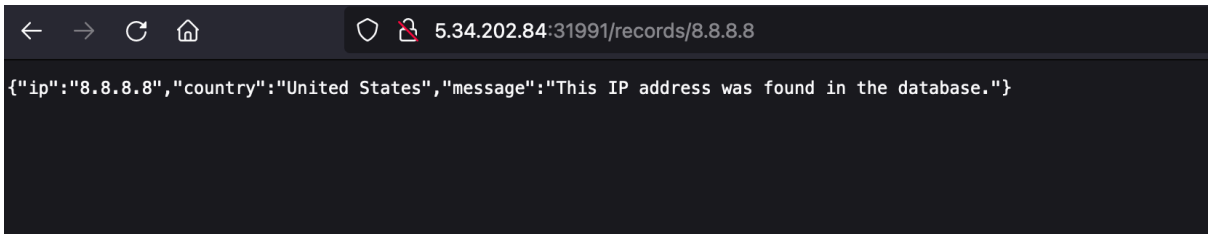
Then check the pod status:

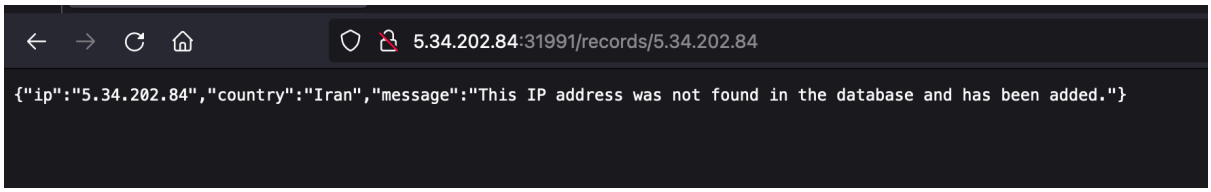
```
root@kuber3:~# kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
ipchecker-deployment-79b5445f67-4brbc 1/1     Running   0          5s
postgresdb-5cf94bf6fb-55tpf           1/1     Running   0          3d11h
postgresdb-5cf94bf6fb-9hmq9           1/1     Running   0          3d11h
postgresdb-5cf94bf6fb-xzh6s           1/1     Running   0          3d11h
prometheus-operator-6c9b57bcb8-9kg5g  1/1     Running   0          3d22h
prometheus-prometheus-0               2/2     Running   0          3d22h
root@kuber3:~#
```

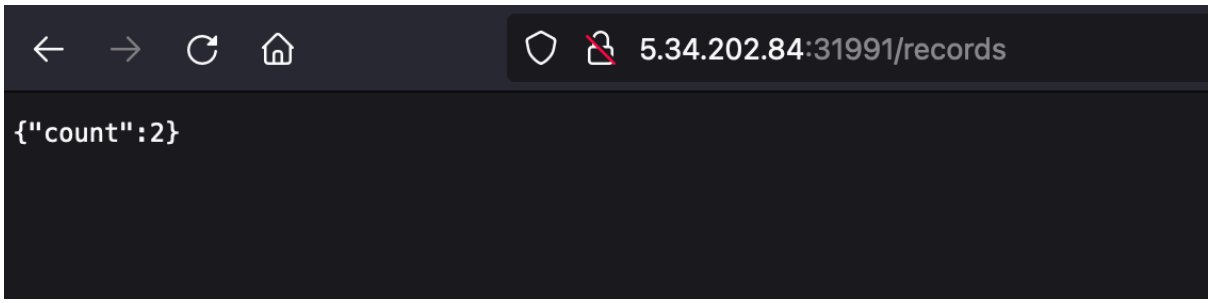
Now see the port maping in your service:

```
PORT(S)
80:31991/TCP
```

Finally it is deployed in Kubernetes cluster and you can see the result:





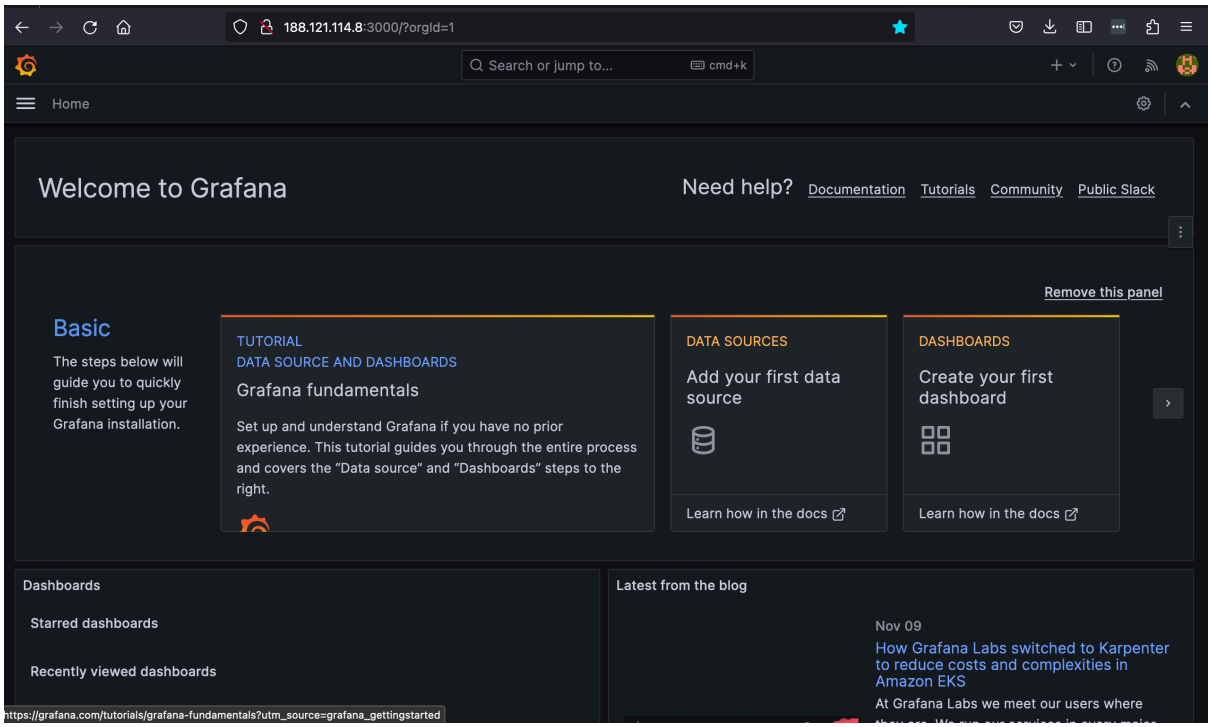


Phase 8(service-monitoring):

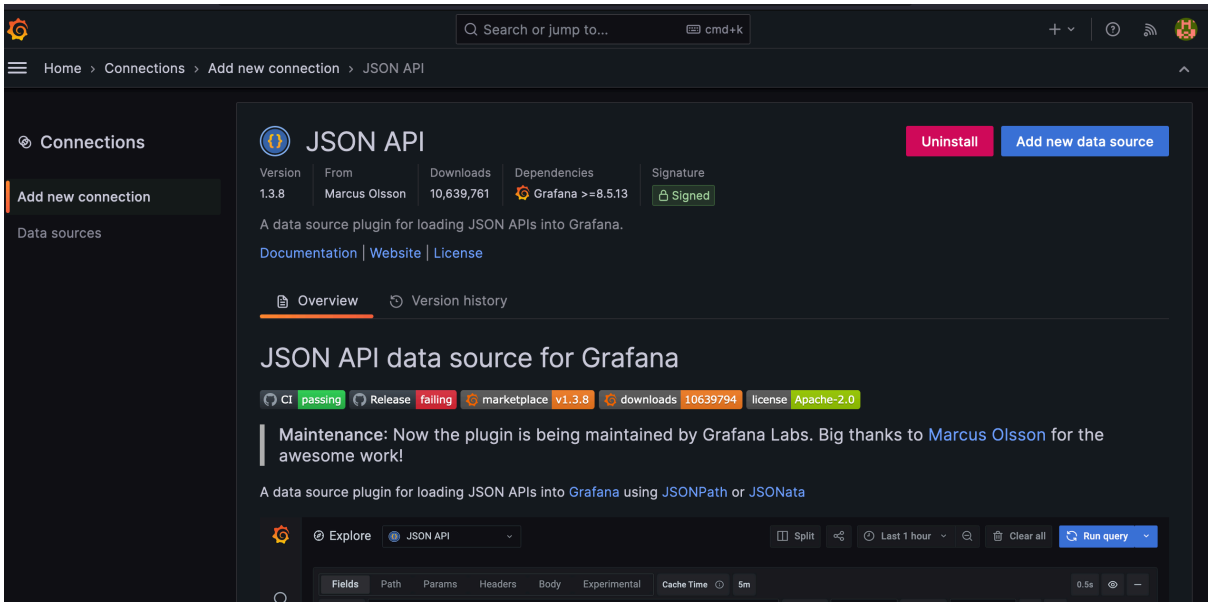
My Grafana that was installed by snap didn't have http source and it cannot possible to add it، so I install Grafana by apt get in another server by this link:

https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/

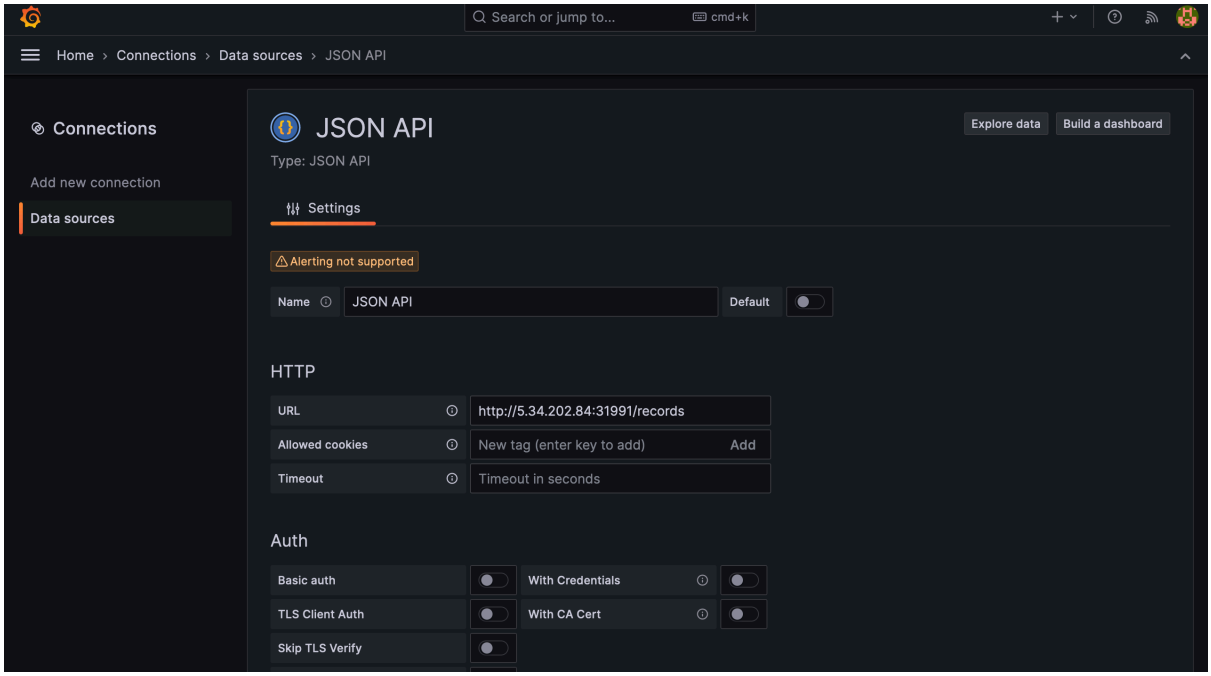
here is the new grafan:



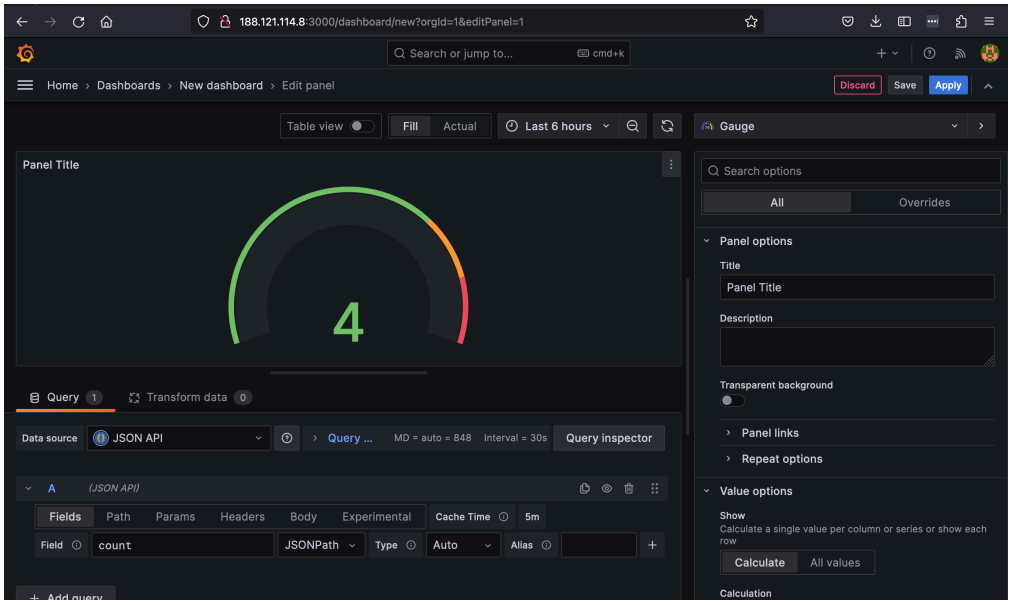
And now we should install this data source:



And add it:



And create dashboard for it :



Now their service monitoring is ready: