

- 1, Creating EKS cluster with 2 worker nodes in your own VPC (see helm-project-instruction.pdf)
- 2, Deploying ChatApp application using helm chart VPC (see helm-project-instruction.pdf)

### 3, Installing Kubernetes Metrics Server

Run command:

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
```

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl get deployment metrics-server -n kube-system
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           100s
```

Check the resource utilization:

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl top node
NAME                           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
ip-10-0-1-106.us-west-2.compute.internal   40m        4%    533Mi        35%
ip-10-0-2-182.us-west-2.compute.internal   53m        5%    1009Mi       67%
```

- 4, Deploying Prometheus on EKS Kubernetes Cluster

Step 1: Create Prometheus namespace:

Run command: kubectl create namespace Prometheus

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl create namespace prometheus
namespace/prometheus created
```

Step 2: install

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> helm install prometheus prometheus-community/prometheus --namespace prometheus
--set alertmanager.persistentVolume.storageClass="gp2" --set server.persistentVolume.storageClass="gp2"
NAME: prometheus
LAST DEPLOYED: Sun Sep  4 15:51:39 2022
NAMESPACE: prometheus
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.prometheus.svc.cluster.local

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

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

Get the Alertmanager URL by running these commands in the same shell:
export POD_NAME=$(kubectl get pods --namespace prometheus -l "app=prometheus,component=alertmanager" -o jsonpath="{.items[0].metadata.name}")
kubectl --namespace prometheus port-forward $POD_NAME 9093
#####
##### WARNING: Pod Security Policy has been moved to a global property. #####
##### use .Values.podSecurityPolicy.enabled with pod-based #####
##### annotations #####
##### (e.g. .Values.nodeExporter.podSecurityPolicy.annotations) #####
#####

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

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

Activate Windows

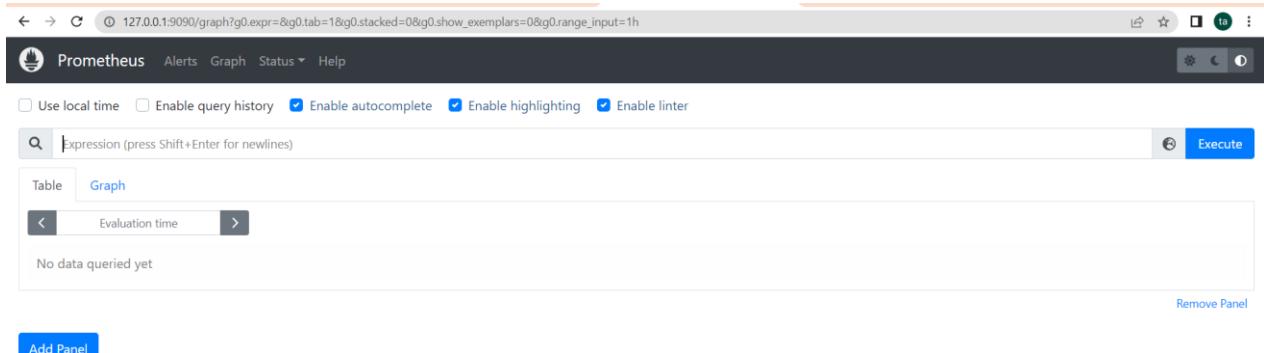
[Get Started](#) | [Private Windows](#)

Check if Prometheus components deployed as expected

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl get pod -n prometheus
  NAME           READY   STATUS    RESTARTS   AGE
  prometheus-alertmanager-669c87db47-jjsjm   2/2     Running   0          12m
  prometheus-kube-state-metrics-77ddf69b4-lmw9l   1/1     Running   0          12m
  prometheus-node-exporter-ljl2r      1/1     Running   0          12m
  prometheus-node-exporter-w764t      1/1     Running   0          12m
  prometheus-pushgateway-5c989b8998-tmjf7   1/1     Running   0          12m
  prometheus-server-57f4946cdb-hpnjn✓   2/2     Running   0          12m
```

Get pod forward:

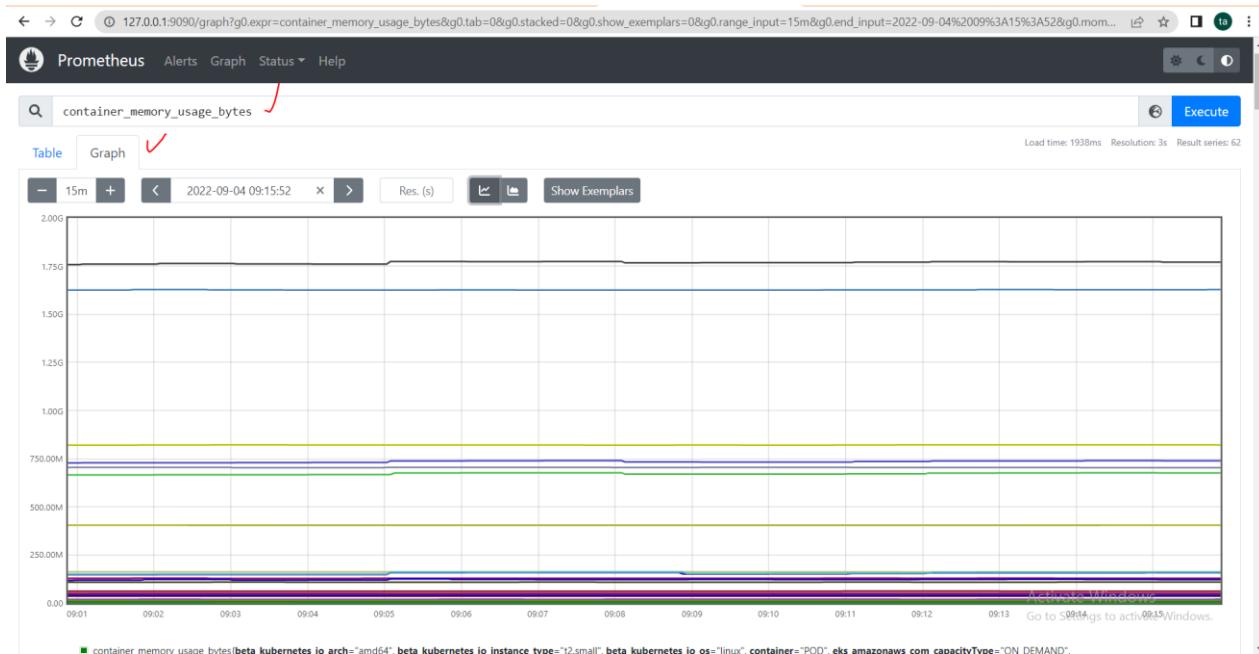
```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl --namespace prometheus port-forward prometheus-server-57f4946cbb-hpnjn 9090
Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::]:9090 -> 9090
Handling connection for 9090
Handling connection for 9090
Handling connection for 9090
```



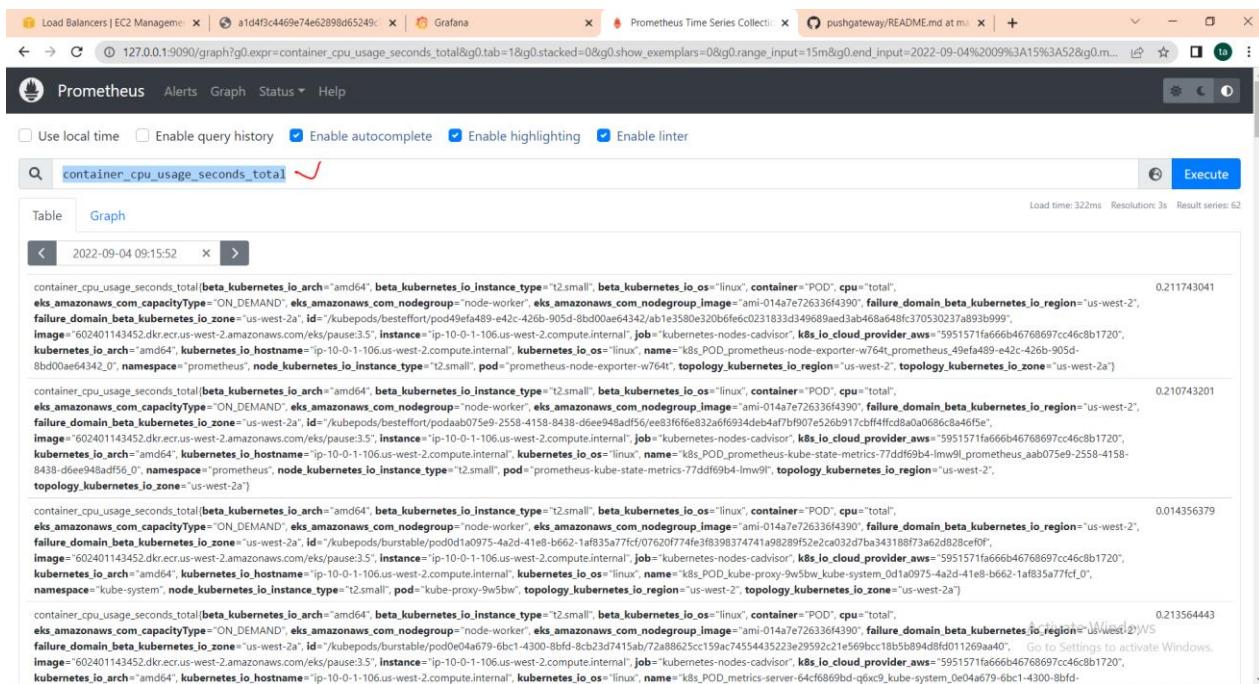
- 5, Monitoring EKS cluster using Prometheus

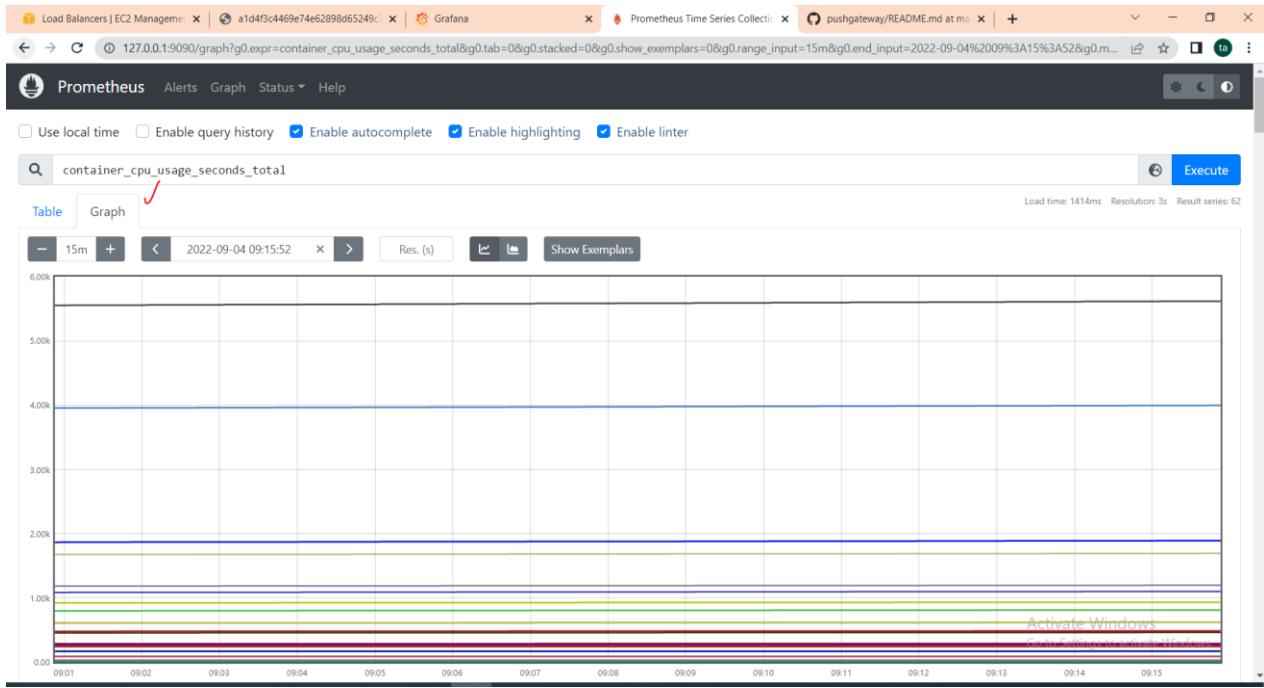
6. Querying some metrics of EKS cluster, such as: CPU, memory, network latency, disk utilization

- query container\_memory\_usage\_bytes:

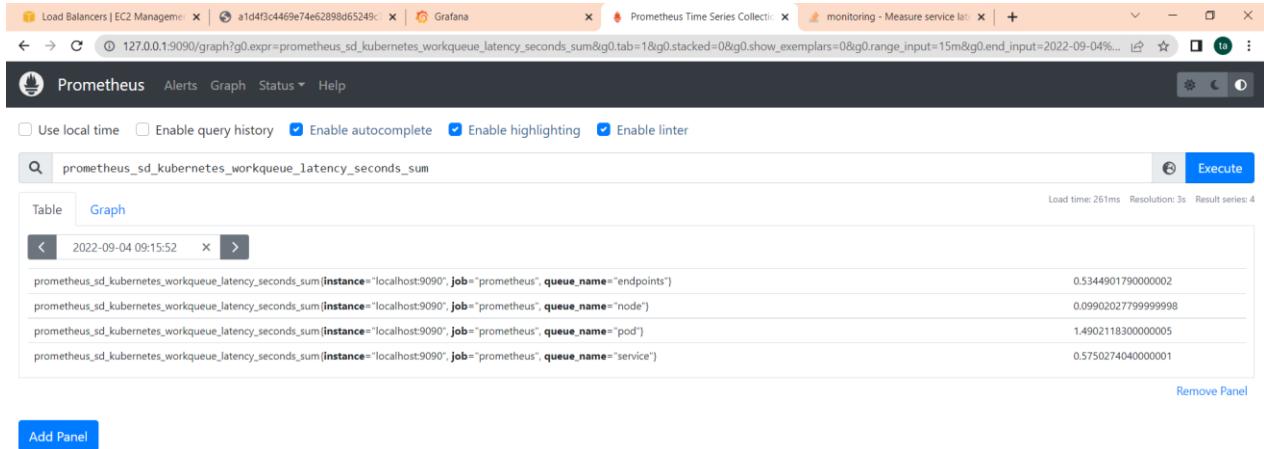


- query : "container\_cpu\_usage\_seconds\_total"

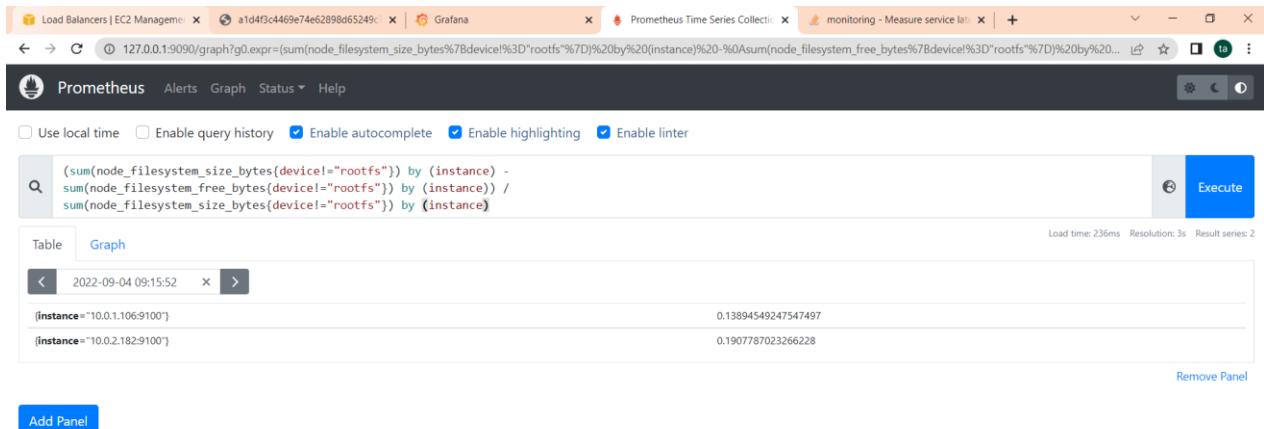




- query: "network latency"



- query: "disk utilization"



- 7, Showing result ChatApp working and log from Prometheus

The screenshot shows the Prometheus Targets page with the following targets listed:

- kubernetes-apiservers (2/2 up)
- kubernetes-nodes (2/2 up)
- kubernetes-nodes-cadvisor (2/2 up)
- kubernetes-service-endpoints (5/5 up)
- prometheus (1/1 up)
- prometheus-pushgateway (1/1 up)

Check: "apiserver\_request\_total"

Metric	Count
apiserver_request_total{code="0", component="apiserver", group="admissionregistration.k8s.io", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="mutatingwebhookconfigurations", scope="cluster", verb="WATCH", version="v1"}	496
apiserver_request_total{code="0", component="apiserver", group="admissionregistration.k8s.io", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="validatingwebhookconfigurations", scope="cluster", verb="WATCH", version="v1"}	496
apiserver_request_total{code="0", component="apiserver", group="admissionregistration.k8s.io", instance="10.0.2.32:443", job="kubernetes-apiservers", resource="mutatingwebhookconfigurations", scope="cluster", verb="WATCH", version="v1"}	250
apiserver_request_total{code="0", component="apiserver", group="admissionregistration.k8s.io", instance="10.0.2.32:443", job="kubernetes-apiservers", resource="validatingwebhookconfigurations", scope="cluster", verb="WATCH", version="v1"}	249
apiserver_request_total{code="0", component="apiserver", group="apiextensions.k8s.io", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="customresourcedefinitions", scope="cluster", verb="WATCH", version="v1"}	478
apiserver_request_total{code="0", component="apiserver", group="apiextensions.k8s.io", instance="10.0.2.32:443", job="kubernetes-apiservers", resource="customresourcedefinitions", scope="cluster", verb="WATCH", version="v1"}	246
apiserver_request_total{code="0", component="apiserver", group="apiregistration.k8s.io", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="apiservices", scope="cluster", verb="WATCH", version="v1"}	485
apiserver_request_total{code="0", component="apiserver", group="apiregistration.k8s.io", instance="10.0.2.32:443", job="kubernetes-apiservers", resource="apiservices", scope="cluster", verb="WATCH", version="v1"}	242
apiserver_request_total{code="0", component="apiserver", group="apps", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="controllerrevisions", scope="cluster", verb="WATCH", version="v1"}	236
apiserver_request_total{code="0", component="apiserver", group="apps", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="daemonsets", scope="cluster", verb="WATCH", version="v1"}	254
apiserver_request_total{code="0", component="apiserver", group="apps", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="deployments", scope="cluster", verb="WATCH", version="v1"}	253
apiserver_request_total{code="0", component="apiserver", group="apps", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="deployments", scope="namespace", verb="WATCH", version="v1"}	246
apiserver_request_total{code="0", component="apiserver", group="apps", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="replicasets", scope="cluster", verb="WATCH", version="v1"}	499
apiserver_request_total{code="0", component="apiserver", group="apps", instance="10.0.1.240:443", job="kubernetes-apiservers", resource="statefulsets", scope="cluster", verb="WATCH", version="v1"}	499

## Monitoring with grafana:

### Step 1:

Create namespace Grafana:

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl create namespace grafana
```

### Step 2:

Add repo grafana:

The screenshot shows the ArtifactHub website with the search bar set to "grafana". The results page for "grafana" is displayed, showing the "Grafana Helm Chart". Key details include:

- Chart Details:** The chart is maintained by the "Grafana" organization and is a "Helmchart". It has 167 stars and 4 production users.
- Chart Components:** Includes "INSTALL", "TEMPLATES", "DEFAULT VALUES", and "CHANGELOG" buttons.
- Chart Versions:** Available versions are 9.1.1, 6.36.2 (2 Sep. 2022), and 6.36.1 (2 Sep. 2022). A note says "CHART VERSIONS available on Windows".
- Get Repo Info:** Shows the command to add the repository: `helm repo add grafana https://grafana.github.io/helm-charts` and `helm repo update`.
- Installing the Chart:** A link to the helm chart's documentation.

### Step 3: Customize file values.yaml in folder grafana repo:

#### Step 1: open file values.yaml

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> helm repo add grafana https://grafana.github.io/helm-charts
"grafana" has been added to your repositories
```

The screenshot shows the Visual Studio Code interface with the "values.yaml" file open in the editor. The code content is as follows:

```
rbac:
  create: true
  ## Use an existing ClusterRole/Role (depending on rbac.namespaced false/true)
  # useExistingRole: name-of-some-(cluster)role
  pspEnabled: true
  pspUseAppArmor: true
  namespaced: false
  extraRoleRules: []
    # - apiGroups: []
    #   resources: []
    #   verbs: []
  extraClusterRoleRules: []
    # - apiGroups: []
    #   resources: []
    #   verbs: []
  serviceAccount:
    create: true
    name:
      nameTest:
        ## Service account annotations. Can be templated.
        # annotations:
        #   eks.amazonaws.com/role-arn: arn:aws:iam::123456789000:role/iam-role-name-here
        autoMount: true
  replicas: 1
  ## Create a headless service for the deployment
  headlessService: false
  ## Create HorizontalPodAutoscaler object for deployment type
  # autoscaling:
  #   enabled: false
  #   minReplicas: 1
  #   maxReplicas: 10
  #   metrics:
  #     - type: Resource
```

## Step 2:

### Configure service type: LoadBalancer

```
grafana > ! values.yaml > {} rbac
153 ## ref: http://kubernetes.io/docs/user-guide/services/
154 ##
155 < service:
156   enabled: true ✓
157   type: LoadBalancer ✓
158   port: 80 ✓
159   targetPort: 3000 ✓
160   # targetPort: 4181 To be used with a proxy extraContainer
161   ## Service annotations. Can be templated.
162   annotations: {}
163   labels: {}
164   portName: service
165   # Adds the appProtocol field to the service. This allows to work with istio protocol selection. Ex: "http" or "tcp"
166   appProtocol: ""
167
168 < serviceMonitor:
169   ## If true, a ServiceMonitor CRD is created for a prometheus operator
170   ## https://github.com/coreos/prometheus-operator
171   ##
172   enabled: false
173   path: /metrics
174   # namespace: monitoring (defaults to use the namespace this chart is deployed to)
175   labels: {}
176   interval: 1m
177   scheme: http
178   tlsConfig: {}
179   scrapeTimeout: 30s
180   relabelings: []
181
182   extraExposePorts: []
183   # - name: keycloak
184   #   port: 8080
185   #   targetPort: 8080
186   #   type: ClusterIP
187
188   # overrides pod.spec.hostAliases in the grafana deployment's pods
189   hostAliases: []
190
```

## Step 3:

### Configure user, password for admin

```
# Administrator credentials when not using an existing secret (see below)
adminUser: admin ✓
adminPassword: admin ✓

# Use an existing secret for the admin user.
admin:
  ## Name of the secret. Can be templated.
  existingSecret: ""
  userKey: admin-user
  passwordKey: admin-password
```

## Step 4: Install grafana:

### Run command:

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> helm install grafana ./grafana
W0904 14:53:08.802834 15088 warnings.go:70] policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
W0904 14:53:22.994229 15088 warnings.go:70] policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
W0904 14:53:23.006135 15088 warnings.go:70] policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
NAME: grafana
LAST DEPLOYED: Sun Sep 4 14:53:06 2022
NAMESPACE: grafana
STATUS: deployed
1. Get your 'admin' user password by running:
  kubectl get secret --namespace grafana grafana -o jsonpath='{.data.admin-password}' | base64 --decode ; echo

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

3. Login with the password from step 1 and the username: admin
#####
##### WARNING: Persistence is disabled!!! You will lose your data when #####
#####           the Grafana pod is terminated. #####
#####
```

## Check service grafana:

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl get svc --namespace grafana -w
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP
grafana   LoadBalancer  172.20.37.27  afa92bd7680a9487cbafc0f48d952de5-893729063.us-west-2.elb.amazonaws.com
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl get pod,svc -n grafana
NAME                READY   STATUS    RESTARTS   AGE
pod/grafana-fdd75664c-6vx28  1/1     Running   0          115s
NAME              TYPE      CLUSTER-IP      EXTERNAL-IP
service/grafana  LoadBalancer  172.20.37.27  afa92bd7680a9487cbafc0f48d952de5-893729063.us-west-2.elb.amazonaws.com  ✓ PORT(S)  80:32696/TCP  116s
```

## Check load balancer:

The screenshot shows the AWS EC2 Load Balancers console. On the left, there's a sidebar with various navigation options like New EC2 Experience, Services, Events, Tags, Limits, Instances, Images, Elastic Block Store, Network & Security, and Feedback. The main area displays a table of load balancers. One row is highlighted with a red arrow pointing to the 'Status' column, which shows '2 of 2 instances in service'. Another row is unselected. The table includes columns for Name, DNS name, VPC ID, Availability Zones, Type, Created At, and Monitoring.

## Step 4:

The screenshot shows a browser window with the URL 'Not secure | afa92bd7680a9487cbafc0f48d952de5-893729063.us-west-2.elb.amazonaws.com/login'. The page displays the Grafana 'Welcome to Grafana' interface. There are two input fields for 'New password' and 'Confirm new password', both containing the value 'admin'. A red arrow points to each of these fields. Below the fields is a 'Submit' button and a 'Skip' link. At the bottom of the page, there's a footer with links to Documentation, Support, Community, Open Source, and a note about a new version available.

## Step 5: Add data source:

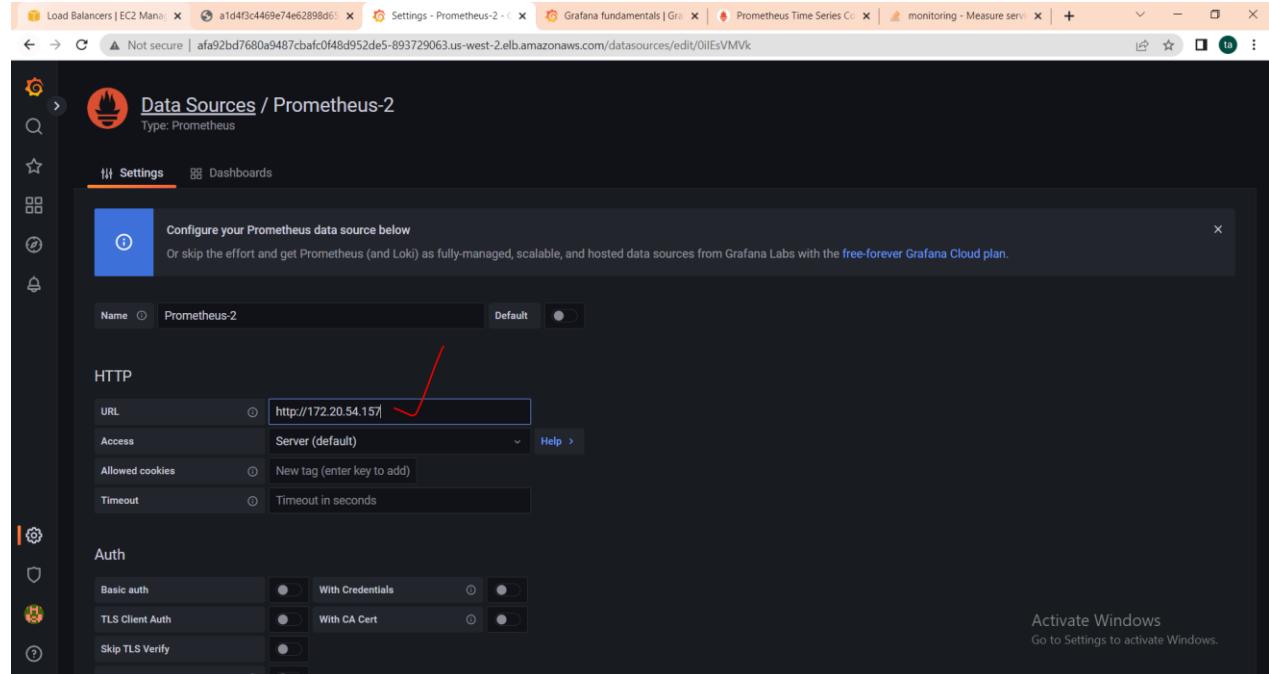
The screenshot shows the Grafana home page with a dark theme. On the left, there's a sidebar with icons for general navigation, search, and dashboard management. The main content area has several panels: one titled 'Basic' with instructions for setting up Grafana; a 'TUTORIAL' panel for 'DATA SOURCE AND DASHBOARDS' with a 'Grafana fundamentals' section; a 'COMPLETE' panel with a 'J' icon and the text 'Add your first data source'; and a 'DASHBOARDS' panel with a 'Create your first dashboard' button. Below these are sections for 'Starred dashboards' and 'Recently viewed dashboards'. A 'Latest from the blog' section features a post by Sep 02 about reducing MTTR with Grafana, Loki, and Tempo.

The screenshot shows the 'Add data source' configuration page. The top bar includes tabs for 'Load Balancers | EC2 Manager', 'a1d4f3c4469e74e52898d65', 'Add data source - Grafana', 'Grafana fundamentals | Gra...', 'Prometheus Time Series Co...', and 'monitoring - Measure serv...'. The main area is titled 'Add data source' with the sub-instruction 'Choose a data source type'. It features a search bar and a 'Time series databases' section with five options: Prometheus (selected, marked with a red checkmark), Graphite, InfluxDB, and OpenTSDB. Below this is a 'Logging & document databases' section with a single option, Loki. A 'Cancel' button is located in the top right corner. A watermark for 'Activate Windows' is visible in the bottom right.

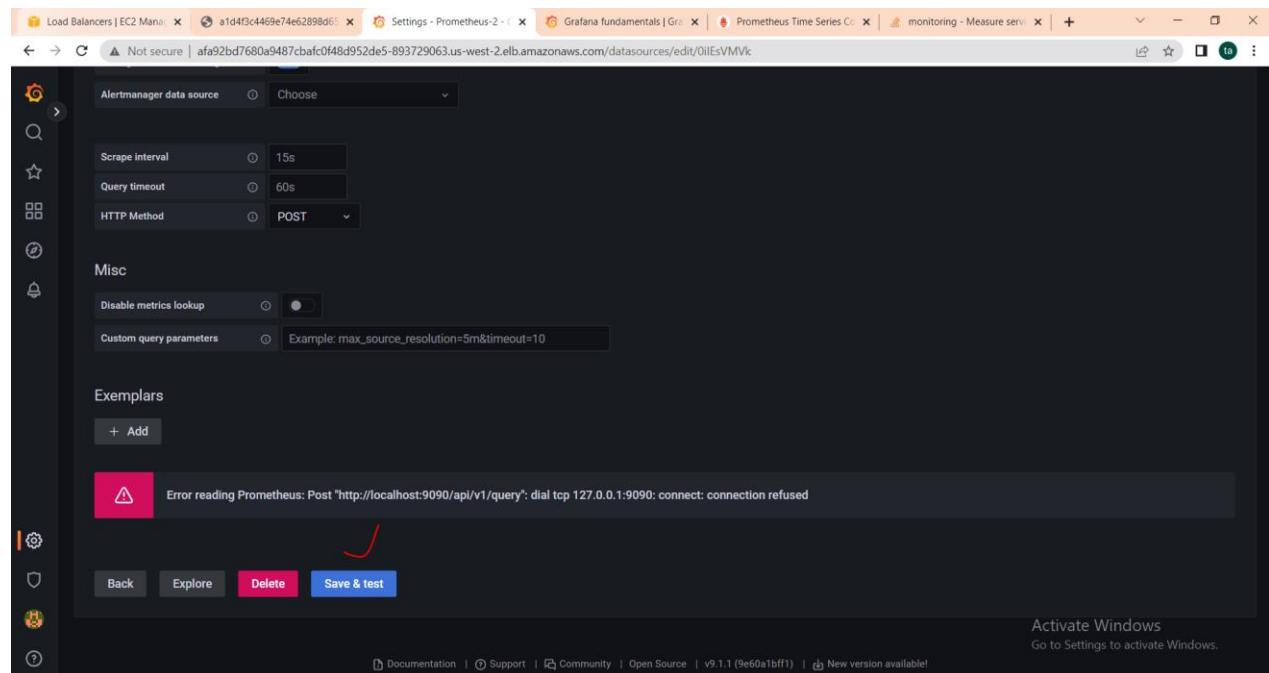
## Check Prometheus service:

```
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> kubectl get svc -n prometheus
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
prometheus-alertmanager  ClusterIP  172.20.248.223 <none>     80/TCP    76m
prometheus-kube-state-metrics  ClusterIP  172.20.252.181 <none>     8080/TCP  76m
prometheus-node-exporter       ClusterIP  172.20.197.90  <none>     9100/TCP  76m
prometheus-pushgateway        ClusterIP  172.20.147.7  <none>     9091/TCP  76m
prometheus-server             ClusterIP  172.20.54.157 ✓<none>     80/TCP    76m
PS C:\Users\DELL\Desktop\Renoleaf-DevOpsBootCamp\W13\Day3\helm_project\charts> |
```

Enter URL



The screenshot shows the 'Data Sources / Prometheus-2' configuration page in Grafana. The 'Settings' tab is selected. In the 'HTTP' section, the 'URL' field contains 'http://172.20.54.157'. A red arrow points from the text above to this URL field.



The screenshot shows the same 'Data Sources / Prometheus-2' configuration page. A prominent red warning icon with an exclamation mark is displayed above the error message. The error message itself reads: 'Error reading Prometheus: Post "http://localhost:9090/api/v1/query": dial tcp 127.0.0.1:9090: connect: connection refused'. Below the error message, there are three buttons: 'Back', 'Explore', and 'Delete' (highlighted with a red arrow), followed by a 'Save & test' button.

The screenshot shows the 'Exemplars' section of the Grafana Data Sources configuration. A green checkmark icon with a red checkmark is highlighted with a red arrow. Below it, a message says 'Data source is working'. At the bottom, there are buttons for 'Back', 'Explore', 'Delete', and 'Save & test'. A watermark at the bottom right reads 'Activate Windows Go to Settings to activate Windows'.

Query:

Go to tab explore:

The screenshot shows the 'Explore' tab for the 'chat-app' data source. It includes fields for 'HTTP' (URL: http://172.20.54.157, Access: Server (default), Allowed cookies: New tag (enter key to add), Timeout: Timeout in seconds) and 'Auth' (Basic auth, TLS Client Auth, Skip TLS Verify, Forward OAuth Identity). A 'Custom HTTP Headers' section is also present. The status bar at the bottom indicates 'Activate Windows Go to Settings to activate Windows'.

The screenshot shows the 'Explore' interface for Prometheus. A raw query is entered in the text area: `orgId=1&left=%7B"datasource":"qMx50VG4k","queries":%5B%7B"refId":"A"%7D%5D;"range...`

The interface includes sections for 'Metric' (Select metric, Choose, =, Choose, +), 'Labels' (Choose, =, Choose, +), 'Operations' (+ Add operations), and 'Raw query' (Options: Legend: Auto, Format: Time series, Step: auto, Type: Both, Exemplars: false). Buttons at the bottom include '+ Add query', 'Query history', and 'Inspector'.

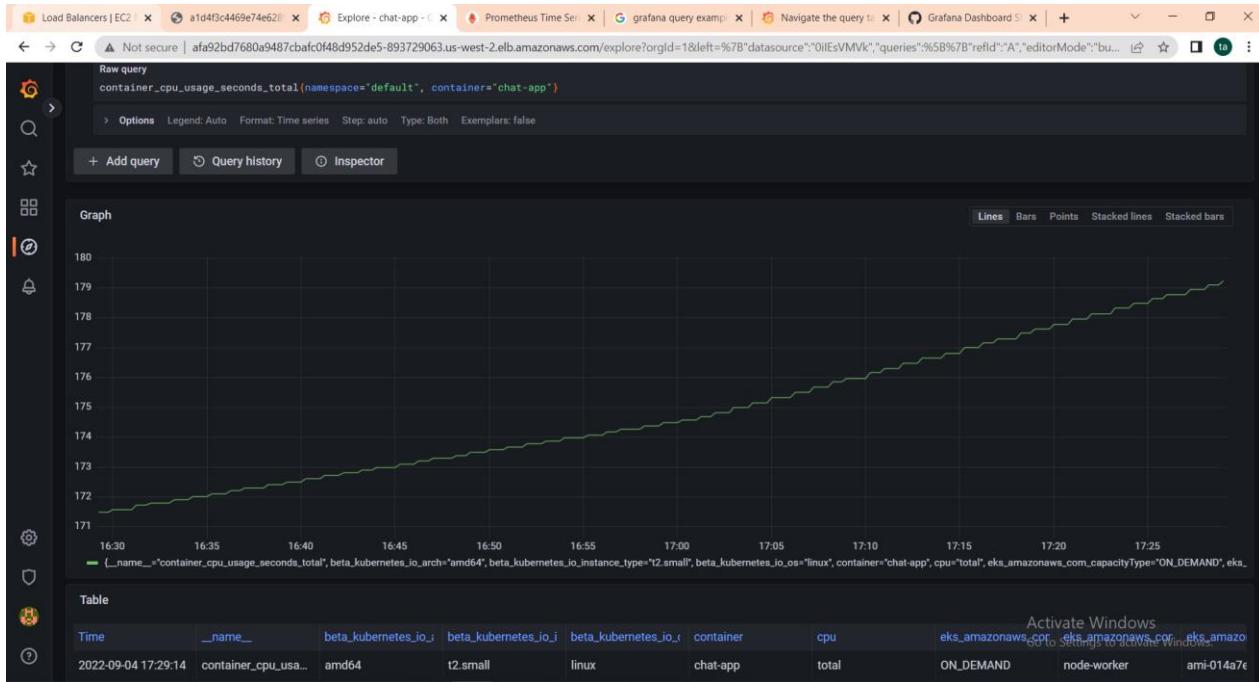
## Select chat-app project to query:

The screenshot shows the Grafana Explore interface. At the top, there are several tabs: 'Load Balancers | EC2', 'a1d4f3c4469e74e62b...', 'Explore - Prometheus', 'Prometheus Time Series', 'grafana query example', 'Navigate the query bar', 'Grafana Dashboard', and '+'. Below the tabs, the 'Explore' tab is active, and the 'chat-app' project is selected. The left sidebar shows 'Prometheus (chat-app)' under 'Prometheus (default)'. The main area has a 'Metric' section with a dropdown set to 'Grafana'. Buttons for '+ Operations', '+ Add query', 'Query history', and 'Inspector' are visible at the bottom.

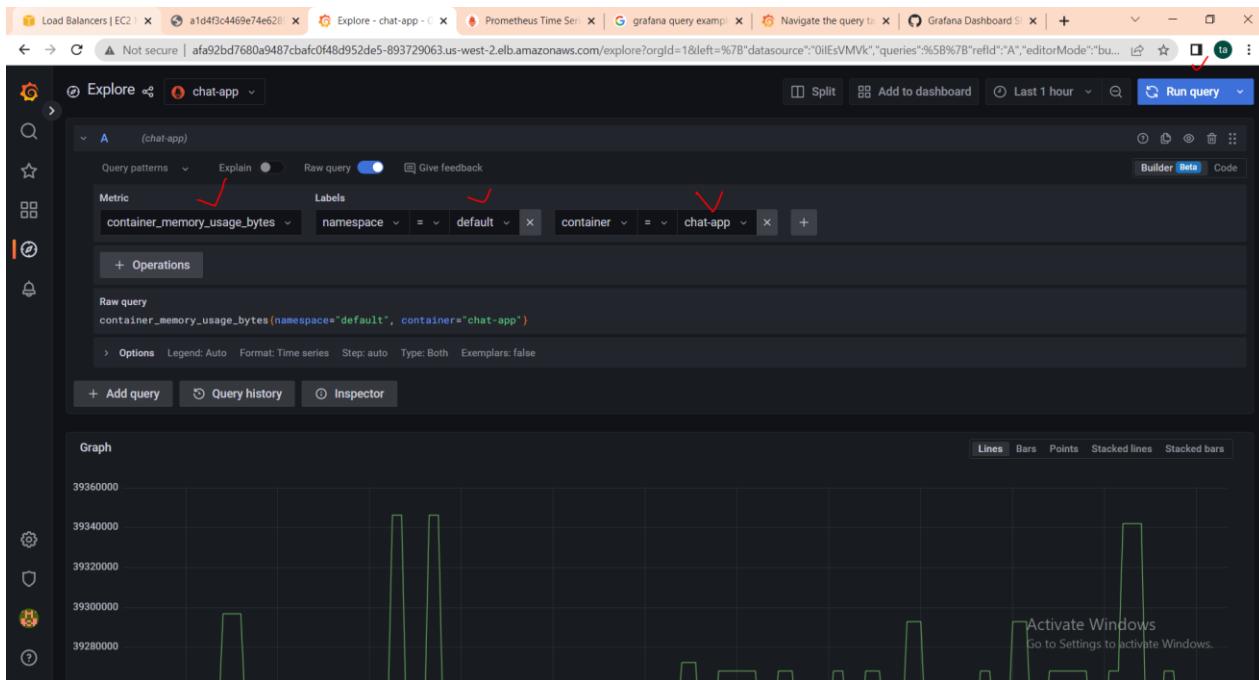
## Query: CPU

The screenshot shows the Grafana Explore interface with a highlighted query. The query is: `container_cpu_usage_seconds_total(namespace="default", container="chat-app")`. The 'Run query' button is highlighted with a red circle. The interface includes a 'Metric' section with dropdowns for 'container\_cpu\_usage\_seconds\_total', 'namespace', 'default', 'container', and 'chat-app'. A 'Labels' section is also present. The 'Raw query' field contains the same code. The bottom navigation buttons are '+ Add query', 'Query history', and 'Inspector'.

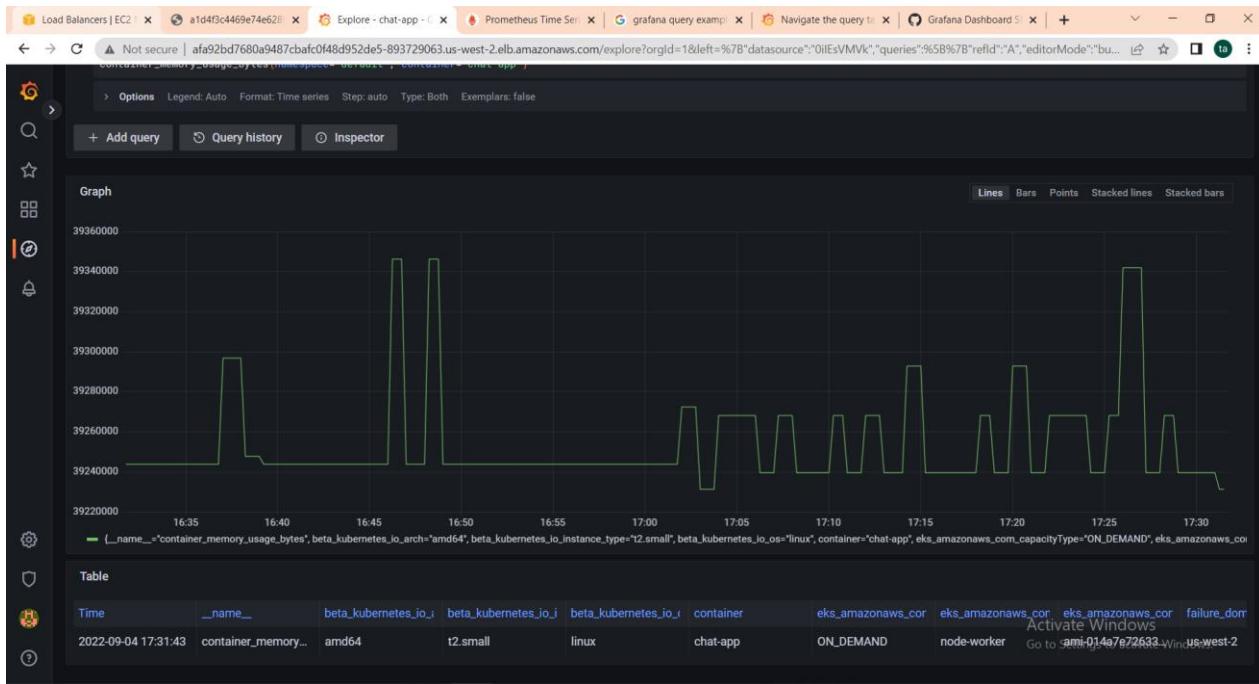
## Result:



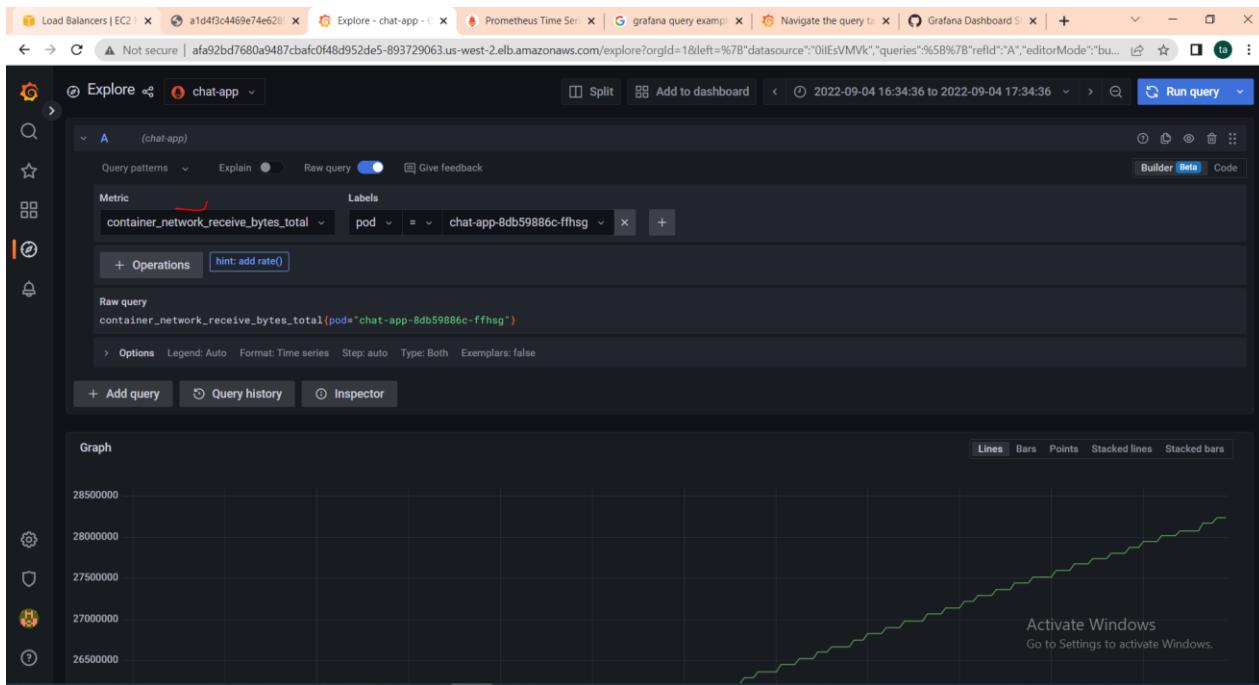
## -Memory:



## Result:



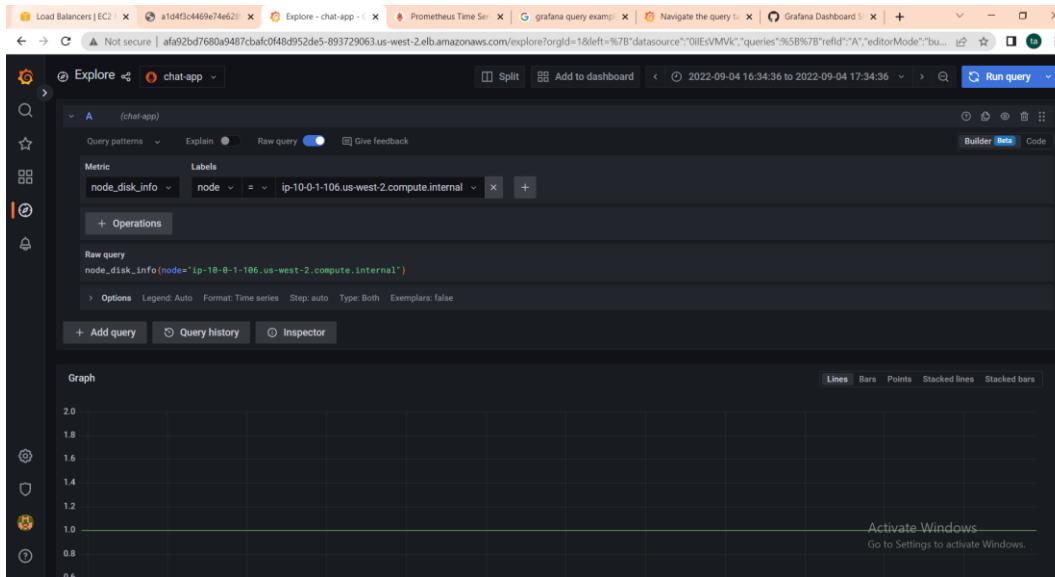
## - Network-latency:



## Result:



## - Disk utilization:



## Result:

