Kong Konnect Enterprise
Kong Gateway 2.4
Ingress Controller 1.3 -
D2IQ Certification
6/14/2021

Kong HQ

150 Spear Street, Suite 1600

San Francisco, CA

United States

Executive Summary

This installation report describes a basic Kong Konnect Enterprise Ingress Controller deployment on a Konvoy Cluster.

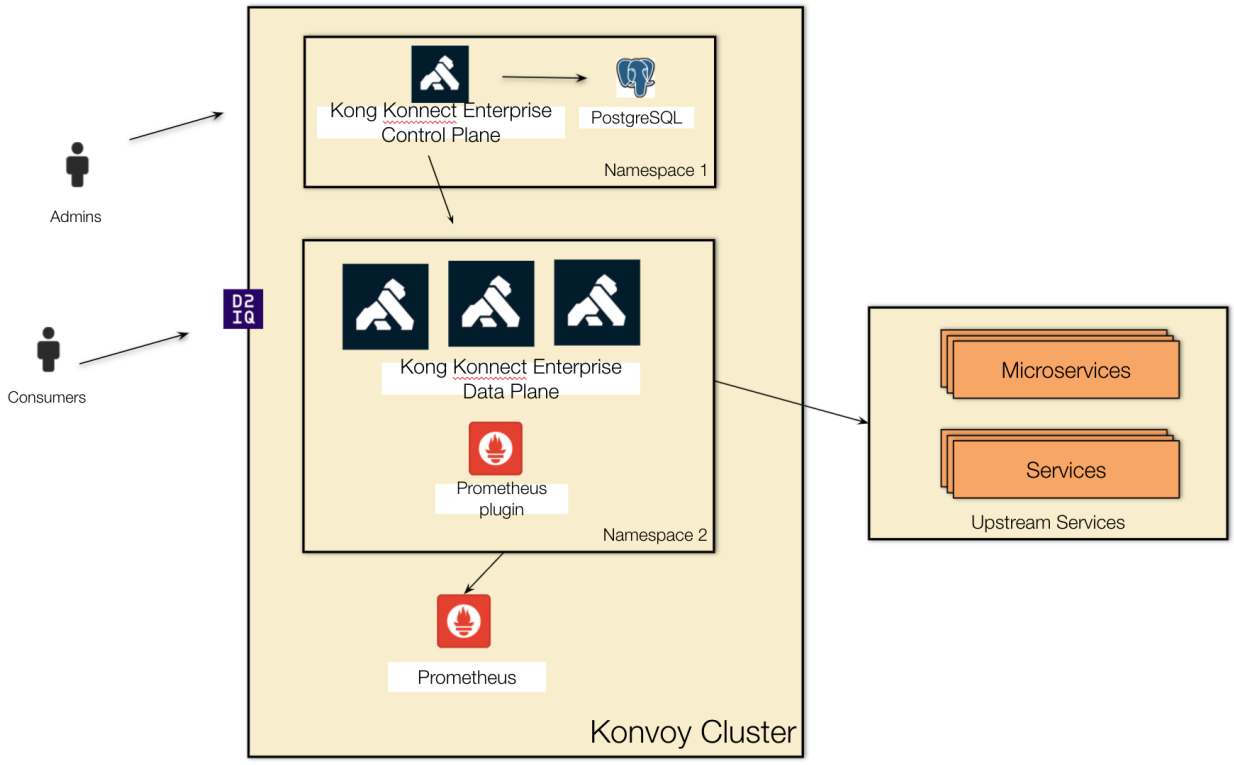
The Konvoy Cluster runs in AWS, eu-central-1 (Frankfurt) region.

Kong Konnect Control Plane and Data Plane run on different Konvoy namespaces. In order to go elastic, the Data Plane takes advantage of important capabilities provided by the Kubernetes platform like HPA.

Fortio was used as the load generator tool and runs on a separate EC2 instance.

The deployment initially uses Kong Konnect version 2.2 and it is upgraded to 2.3 and 2.4 version while being consumed by the load generator.

The figure below illustrates the Architecture:



Konvoy Cluster

Prepare the EC2/Ubuntu 20 VM

If you're going to manage your Konvoy Cluster from a VM follow this section. You can install only Fortio if you are using MacOS.

```
ssh -i "acqua-frankfurt.pem"
ubuntu@ec2-18-195-169-65.eu-central-1.compute.amazonaws.com

sudo apt-get -y update
sudo apt -y install httpie
sudo apt -y install jq
sudo apt -y install awscli

sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
```

Kubectl

```
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list

sudo apt-get update
sudo apt-get install -y kubectl
kubectl version
```

Helm

```
curl https://baltocdn.com/helm/signing.asc | sudo apt-key add -

sudo apt-get install apt-transport-https --yes

echo "deb https://baltocdn.com/helm/stable/debian/ all main" | sudo tee
/etc/apt/sources.list.d/helm-stable-debian.list

sudo apt-get update
sudo apt-get install helm
helm version
```

Docker

```
sudo apt install -y docker.io
sudo systemctl enable docker.service
sudo systemctl start docker.service
sudo docker info
```

Fortio

```
sudo add-apt-repository ppa:longsleep/golang-backports
sudo apt update
sudo apt install golang-go

go get fortio.org/fortio
```

(To run Fortio go to go/bin directory, "/root/go/bin")

Konvoy Installer

Ubuntu

```
wget https://downloads.mesosphere.io/konvoy/konvoy_v1.7.3_linux.tar.bz2
tar xvf *
```

MacOS

```
wget https://downloads.mesosphere.io/konvoy/konvoy_v1.7.3_darwin.tar.bz2
tar xvf *
```

Installing Konvoy

```
aws configure
eu-central-1


konvoy init --cluster-name k4k8s
```

Change `cluster.yaml` file with new:

- AWS Region: eu-central-1
- Availability Zones: eu-central-1a
- AMI Images: ami-095e1a4d3d632d215 (CentOS 7.8.2003 x86_64) for worker, control-plane and bastion.

In case the private key has not been inserted in .ssh directory:

```
ssh-add k4k8s-ssh.pem
ssh-add -L
ssh-add -D


konvoy up --cluster-name k4k8s

…….
Kubernetes cluster and addons deployed successfully!

Run `./konvoy apply kubeconfig` to update kubectl credentials.

Run `./konvoy check` to verify that the cluster has reached a steady state
and all deployments have finished.

Navigate to the URL below to access various services running in the
cluster.

https://a3327e66f1f6b499994cd6a8632b0328-324595487.eu-central-1.elb.amazon
aws.com/ops/landing
And login using the credentials below.
  Username: zealous_lamport
  Password:
lXD12s4vxwFxfbHNStQ3LBooJAemULm0BBEo3DLGa5vGG1TzDqUQS8ID1FVEJIwD

If the cluster was recently created, the dashboard and services may take a
few minutes to be accessible.
```

Konvoy Version

```
$ ./konvoy version
{
    "Version": "v1.7.3",
    "BuildDate": "Thu Jun 10 07:14:09 UTC 2021"
}
```

Update kubeconfig

```
//rm -r /root/.kube

$ konvoy apply kubeconfig
…….
Kubeconfig activated successfully!


$ kubectl config get-contexts
CURRENT   NAME                 CLUSTER   AUTHINFO       NAMESPACE
*         k4k8s-admin@k4k8s    k4k8s     k4k8s-admin
```

Deleting the Cluster

If you want to delete the Cluster run:

```
konvoy down
```

Kommander

https://a3327e66f1f6b499994cd6a8632b0328-324595487.eu-central-1.elb.amazonaws.com/ops/portal/kommander/ui

```
And login using the credentials below.
  Username: zealous_lamport
  Password:
lXD12s4vxwFxfbHNStQ3LBooJAemULm0BBEo3DLGa5vGG1TzDqUQS8ID1FVEJIwD
```

Sample App Installation

Deploy Sample App

```
kubectl create namespace kong-app

cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: sample
  namespace: kong-app
  labels:
    app: sample
spec:
  type: ClusterIP
  ports:
  - port: 5000
    name: http
  selector:
    app: sample
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample
  namespace: kong-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sample
  template:
    metadata:
      labels:
        app: sample
        version: v1
    spec:
      containers:
      - name: sample
        image: claudioacquaviva/sampleapp
```

```
        ports:
        - containerPort: 5000
EOF
```

```
$ kubectl get services -n kong-app
NAME     TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
sample   ClusterIP   10.0.11.226   <none>        5000/TCP   2s

$ kubectl get pod -n kong-app
NAME                      READY   STATUS    RESTARTS   AGE
sample-76db6bb547-wsvbt   1/1     Running   0          31s
```

```
$ kubectl port-forward service/sample -n kong-app 5000
```

```
$ http localhost:5000/hello
HTTP/1.0 200 OK
Content-Length: 45
Content-Type: text/html; charset=utf-8
Date: Tue, 15 Jun 2021 13:21:25 GMT
Server: Werkzeug/1.0.1 Python/3.7.4

Hello World, Kong: 2021-06-15 13:21:25.806664
```

Deleting the application

```
kubectl delete service sample -n kong-app
kubectl delete deployment sample -n kong-app
kubectl delete namespace kong-app
```

Kong Konnect Enterprise 2.2 + Ingress Controller 1.2.0 installation

```
$ helm version
version.BuildInfo{Version:"v3.6.1",
GitCommit:"61d8e8c4a6f95540c15c6a65f36a6dd0a45e7a2f",
GitTreeState:"dirty", GoVersion:"go1.16.5"}


helm repo add kong https://charts.konghq.com
helm repo update
helm repo list

$ helm show chart kong/kong
apiVersion: v1
appVersion: "2.4"
dependencies:
- condition: postgresql.enabled
  name: postgresql
  repository: https://charts.bitnami.com/bitnami
  version: 8.6.8
description: The Cloud-Native Ingress and API-management
home: https://konghq.com/
icon:
https://s3.amazonaws.com/downloads.kong/universe/assets/icon-kong-inc-larg
e.png
maintainers:
- email: harry@konghq.com
  name: hbagdi
- email: traines@konghq.com
  name: rainest
name: kong
version: 2.1.0
```

First of all, create a "kong" namespace

```
kubectl create namespace kong
```

Create a secret with your license file

```
kubectl create secret generic kong-enterprise-license -n kong
--from-file=./license
```

Generating Private Key and Digital Certificate

```
openssl req -new -x509 -nodes -newkey ec:<(openssl ecparam -name
secp384r1) \
  -keyout ./cluster.key -out ./cluster.crt \
  -days 1095 -subj "/CN=kong_clustering"

kubectl create secret tls kong-cluster-cert --cert=./cluster.crt
--key=./cluster.key -n kong
```

Control Plane

```
helm install kong kong/kong -n kong \
--set ingressController.enabled=true \
--set ingressController.installCRDs=false \
--set
ingressController.image.repository=kong/kubernetes-ingress-controller \
--set ingressController.image.tag=1.2.0 \
--set image.repository=kong/kong-gateway \
--set image.tag=2.2.1.3-alpine \
--set env.database=postgres \
--set env.role=control_plane \
--set env.cluster_cert=/etc/secrets/kong-cluster-cert/tls.crt \
--set env.cluster_cert_key=/etc/secrets/kong-cluster-cert/tls.key \
--set cluster.enabled=true \
--set cluster.tls.enabled=true \
--set cluster.tls.servicePort=8005 \
--set cluster.tls.containerPort=8005 \
--set clustertelemetry.enabled=true \
--set clustertelemetry.tls.enabled=true \
--set clustertelemetry.tls.servicePort=8006 \
--set clustertelemetry.tls.containerPort=8006 \
--set proxy.enabled=true \
--set admin.enabled=true \
--set admin.http.enabled=true \
--set admin.type=LoadBalancer \
--set admin.labels.enable-metrics=true \
--set enterprise.enabled=true \
```

```
--set enterprise.license_secret=kong-enterprise-license \
--set enterprise.portal.enabled=false \
--set enterprise.rbac.enabled=false \
--set enterprise.smtp.enabled=false \
--set manager.enabled=true \
--set manager.type=LoadBalancer \
--set secretVolumes[0]=kong-cluster-cert \
--set postgresql.enabled=true \
--set postgresql.postgresqlUsername=kong \
--set postgresql.postgresqlDatabase=kong \
--set postgresql.postgresqlPassword=kong
```

Data Plane

```
kubectl create namespace kong-dp

kubectl create secret generic kong-enterprise-license -n kong-dp
--from-file=./license

kubectl create secret tls kong-cluster-cert --cert=./cluster.crt
--key=./cluster.key -n kong-dp


helm install kong-dp kong/kong -n kong-dp \
--set ingressController.enabled=false \
--set image.repository=kong/kong-gateway \
--set image.tag=2.2.1.3-alpine \
--set env.database=off \
--set env.role=data_plane \
--set env.cluster_cert=/etc/secrets/kong-cluster-cert/tls.crt \
--set env.cluster_cert_key=/etc/secrets/kong-cluster-cert/tls.key \
--set
env.lua_ssl_trusted_certificate=/etc/secrets/kong-cluster-cert/tls.crt \
--set
env.cluster_control_plane=kong-kong-cluster.kong.svc.cluster.local:8005 \
--set
env.cluster_telemetry_endpoint=kong-kong-clustertelemetry.kong.svc.cluster
.local:8006 \
--set proxy.enabled=true \
--set proxy.type=LoadBalancer \
```

```
--set enterprise.enabled=true \
--set enterprise.license_secret=kong-enterprise-license \
--set enterprise.portal.enabled=false \
--set enterprise.rbac.enabled=false \
--set enterprise.smtp.enabled=false \
--set manager.enabled=false \
--set portal.enabled=false \
--set portalapi.enabled=false \
--set env.status_listen=0.0.0.0:8100 \
--set secretVolumes[0]=kong-cluster-cert
```

Checking the Installation

```
$ kubectl get deployment -n kong
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
kong-kong   1/1     1            1           2m3s

$ kubectl get deployment -n kong-dp
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
kong-dp-kong   1/1     1            1           36s


$ kubectl get pod -n kong
NAME                              READY   STATUS      RESTARTS   AGE
kong-kong-d865f7f4-g8cxl          2/2     Running     0          2m26s
kong-kong-init-migrations-fmgfl   0/1     Completed   0          2m26s
kong-postgresql-0                 1/1     Running     0          2m26s

$ kubectl get pod -n kong-dp
NAME                            READY   STATUS    RESTARTS   AGE
kong-dp-kong-f6c6fd568-p7w2r   1/1     Running   0          61s



$ kubectl get service -n kong
NAME                         TYPE           CLUSTER-IP    EXTERNAL-IP
PORT(S)                         AGE
kong-kong-admin              LoadBalancer   10.0.4.111
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com
8001:31889/TCP,8444:30857/TCP   106s
kong-kong-cluster            ClusterIP      10.0.41.177   <none>
8005/TCP                        106s
kong-kong-clustertelemetry   ClusterIP      10.0.55.196   <none>
8006/TCP                        106s
kong-kong-manager            LoadBalancer   10.0.4.37
ac47d907d972b4f648bdb496f3b508e9-1933016071.eu-central-1.elb.amazonaws.com
8002:30219/TCP,8445:30537/TCP   106s
kong-kong-portal             NodePort       10.0.3.175    <none>
8003:30328/TCP,8446:31468/TCP   106s
```

```
kong-kong-portalapi          NodePort       10.0.23.193   <none>
8004:32019/TCP,8447:30673/TCP   106s
kong-kong-proxy              LoadBalancer   10.0.26.106
ab27be50edb204d4290648d128fddb30-1912033636.eu-central-1.elb.amazonaws.com
80:31290/TCP,443:30591/TCP      106s
kong-postgresql              ClusterIP      10.0.0.8      <none>
5432/TCP                        106s
kong-postgresql-headless     ClusterIP      None          <none>
5432/TCP                        106s

$ kubectl get service -n kong-dp
NAME                 TYPE           CLUSTER-IP   EXTERNAL-IP
PORT(S)                      AGE
kong-dp-kong-proxy   LoadBalancer   10.0.48.95
af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.com
80:32025/TCP,443:31228/TCP   22s
```

Configuring Kong Manager Service

```
$ kubectl get service -n kong kong-kong-admin
--output=jsonpath='{.status.loadBalancer.ingress[0].hostname}'
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com

kubectl patch deployment -n kong kong-kong -p "{\"spec\": { \"template\" :
{ \"spec\" : {\"containers\":[{\"name\":\"proxy\",\"env\": [{ \"name\" :
\"KONG_ADMIN_API_URI\", \"value\":
\"a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.co
m:8001\" }]}]}}}}"
```

Checking the Admin

Use the Load Balancer created during the deployment

```
$ http
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com:
8001 | jq .version
"2.2.1.3-enterprise-edition"
```

Checking the Proxy

Use the Load Balancer created during the deployment

```
kubectl get service -n kong-dp kong-dp-kong-proxy
--output=jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

```
af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.com
```

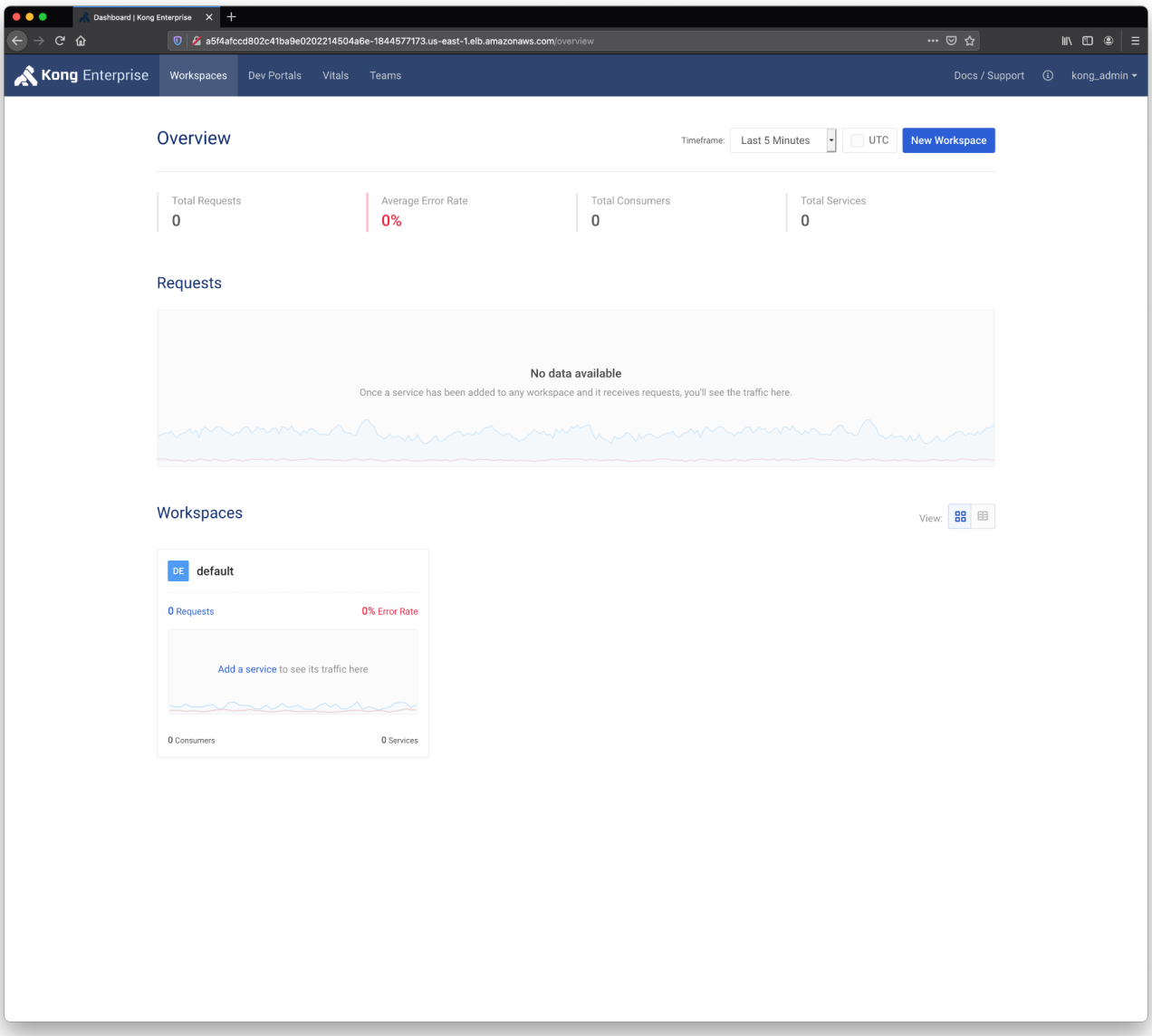
```
$ http af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.com
HTTP/1.1 404 Not Found
Connection: keep-alive
Content-Length: 48
Content-Type: application/json; charset=utf-8
Date: Fri, 25 Jun 2021 13:37:38 GMT
Server: kong/2.2.1.3-enterprise-edition
X-Kong-Response-Latency: 0

{
    "message": "no Route matched with those values"
}
```

Logging to Kong Manager

Login to Kong Manager using the specific ELB:

```
$ kubectl get service -n kong kong-kong-manager
--output=jsonpath='{.status.loadBalancer.ingress[0].hostname}'
ac47d907d972b4f648bdb496f3b508e9-1933016071.eu-central-1.elb.amazonaws.com
```

http://ac47d907d972b4f648bdb496f3b508e9-1933016071.eu-central-1.elb.amazonaws.com:8002



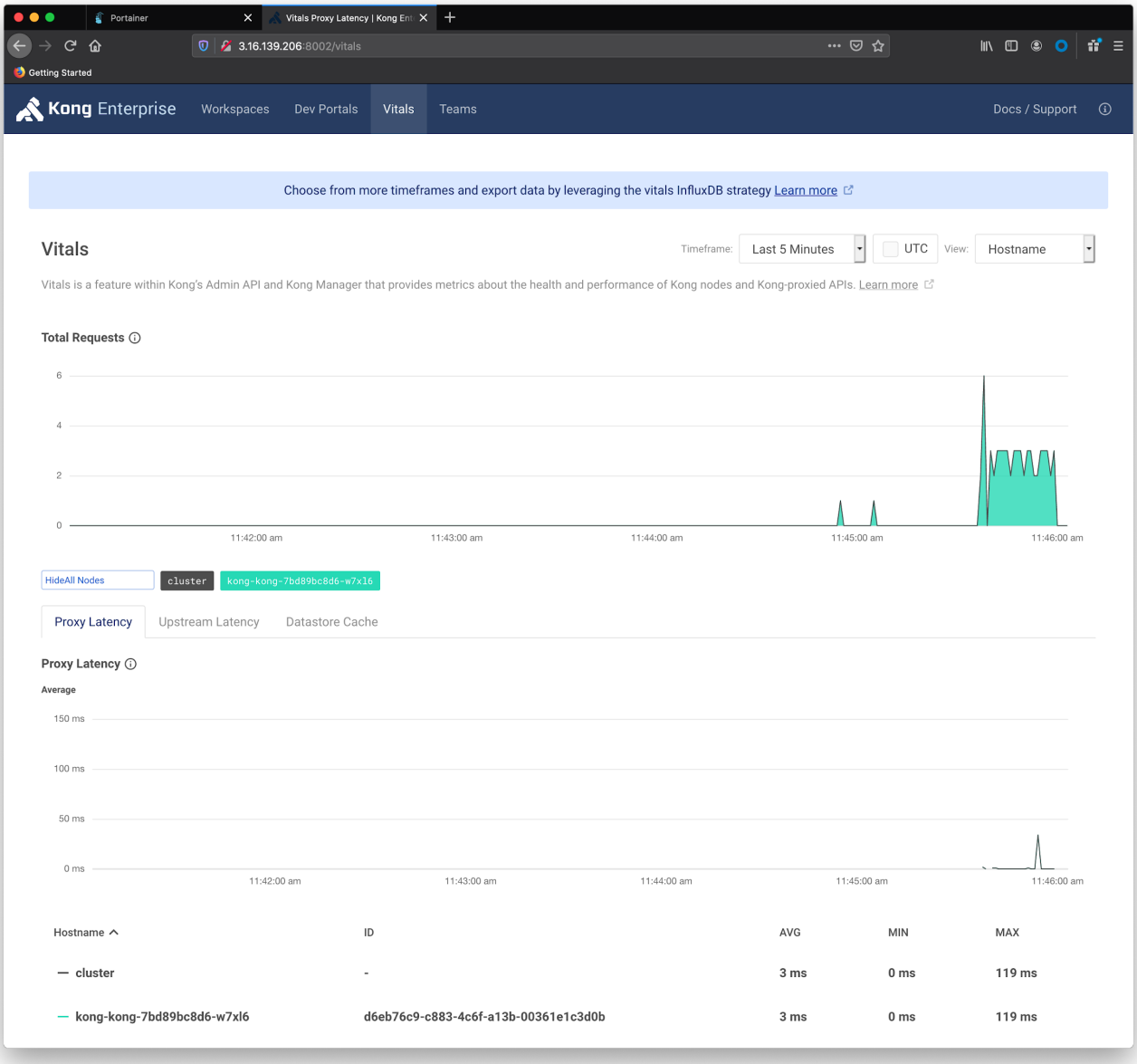
Defining a Service and a Route

From your laptop define a service and a route sending requests to the Control Plane.

```
http
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com:
8001/services name=sampleservice
url='http://sample.kong-app.svc.cluster.local:5000'
```

```
http
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com:
8001/services/sampleservice/routes name='httpbinroute'
paths:='["/sample"]'
```

```
while [ 1 ]; do curl
http://af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazon
aws.com:80/sample/hello; echo; done
```



Ingress Creation

Kubernetes External Service

```
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: route1-ext
  namespace: default
spec:
  type: ExternalName
  externalName: httpbin.org
EOF
```

Ingress

```
cat <<EOF | kubectl apply -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: route1
  namespace: default
  annotations:
    konghq.com/strip-path: "true"
    kubernetes.io/ingress.class: kong
spec:
  rules:
  - http:
      paths:
        - path: /route1
          backend:
            serviceName: route1-ext
            servicePort: 80
EOF
```

Ingress

Basic Consumption

```
$ http
af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.com
/route1/get
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 618
Content-Type: application/json
Date: Fri, 25 Jun 2021 13:40:33 GMT
Server: gunicorn/19.9.0
Via: kong/2.2.1.3-enterprise-edition
X-Kong-Proxy-Latency: 1
X-Kong-Upstream-Latency: 180

{
    "args": {},
    "headers": {
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Host":
"a1b63a482d92a45a69bd27050f3625da-1340677491.eu-central-1.elb.amazonaws.co
m",
        "User-Agent": "HTTPie/2.4.0",
        "X-Amzn-Trace-Id": "Root=1-60d5dcd1-4a5de1213b4b2fdd2b1002e2",
        "X-Forwarded-Host":
"a1b63a482d92a45a69bd27050f3625da-1340677491.eu-central-1.elb.amazonaws.co
m",
        "X-Forwarded-Path": "/route1/get",
        "X-Forwarded-Prefix": "/route1"
    },
    "origin": "192.168.28.64, 18.159.129.18",
    "url":
"http://a1b63a482d92a45a69bd27050f3625da-1340677491.eu-central-1.elb.amazo
naws.com/get"
}
```

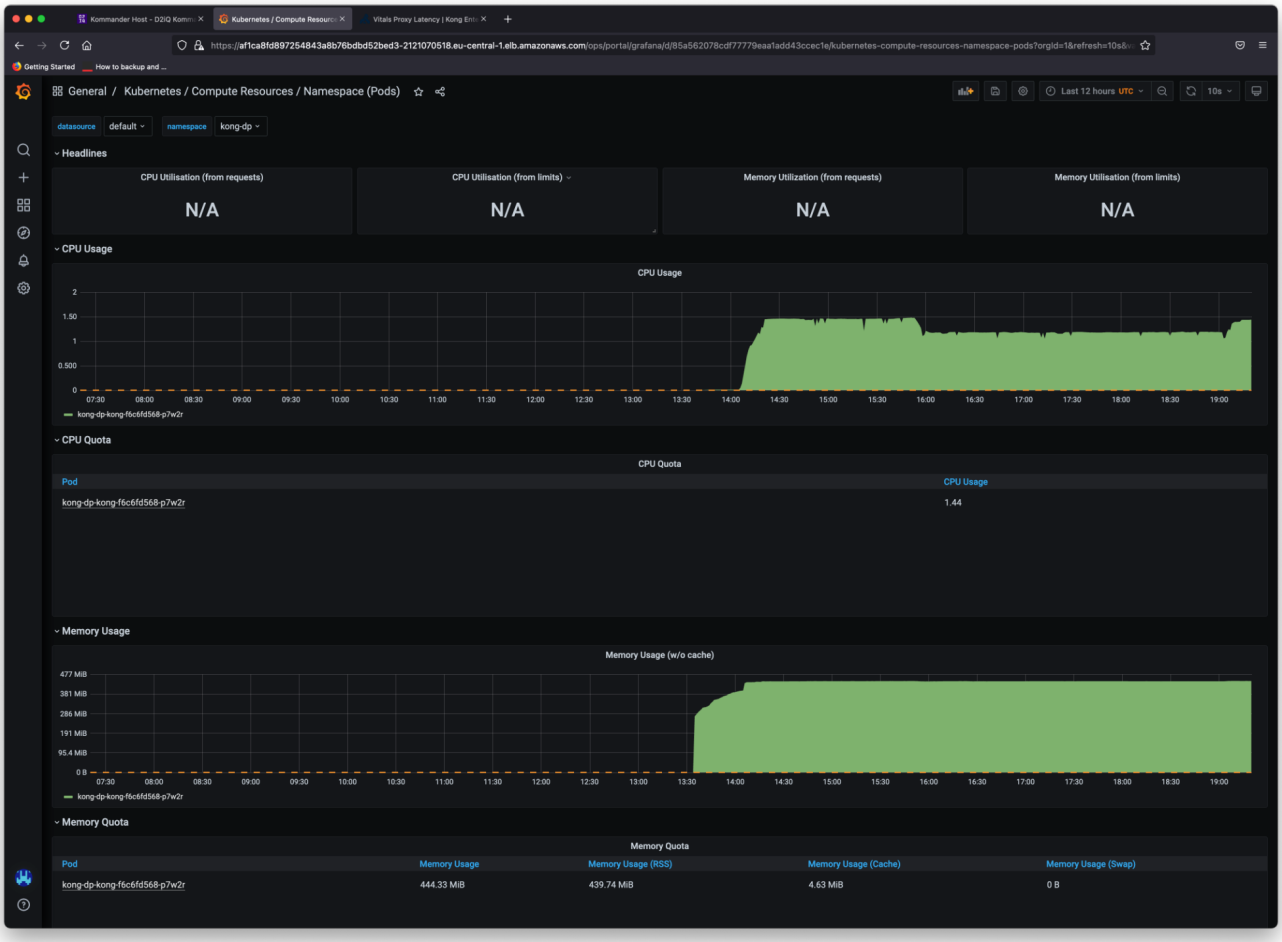
If you want to start a loop:

```
while [ 1 ]; do curl
http://af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazon
aws.com/route1/get; echo; done
```

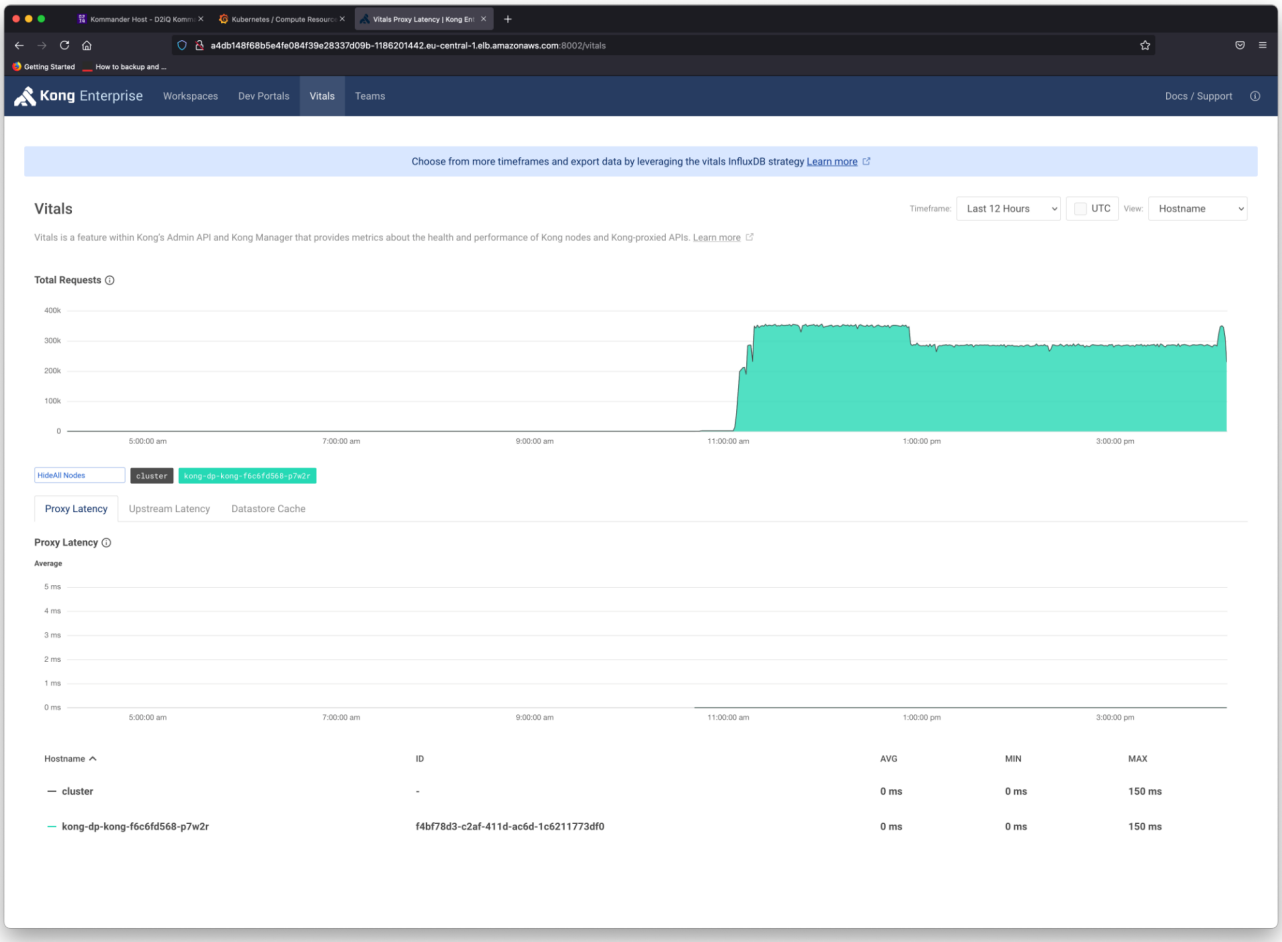
Fortio

```
# /root/go/bin/fortio load -c 120 -qps 2000 -t 0
http://af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazon
aws.com/route1/get
```

Konvoy Grafana



Kong Konnect Vitals



Ingress Monitoring

Monitoring Data Plane

Define a Global Prometheus Plugin:

```
cat <<EOF | kubectl apply -f -
apiVersion: configuration.konghq.com/v1
kind: KongClusterPlugin
metadata:
  name: prometheus-plugin
  annotations:
    kubernetes.io/ingress.class: kong
  labels:
    global: "true"
plugin: prometheus
EOF
```

Expose the Data Plane metric port as a Kubernetes Service

```
kubectl delete service kong-dp-status -n kong-dp

cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: kong-dp-status
  namespace: kong-dp
  labels:
    app: kong-dp-status
spec:
  selector:
    app.kubernetes.io/name: kong
  type: ClusterIP
  ports:
  - name: metrics
    protocol: TCP
    port: 8100
    targetPort: 8100
EOF

$ kubectl get service -n kong-dp
```

```
NAME                  TYPE           CLUSTER-IP   EXTERNAL-IP
PORT(S)                       AGE
kong-dp-kong-proxy   LoadBalancer   10.0.42.17
a1b63a482d92a45a69bd27050f3625da-1340677491.eu-central-1.elb.amazonaws.com
80:32330/TCP,443:31392/TCP   13m
kong-dp-status       ClusterIP      10.0.43.2    <none>
8100/TCP                      26s
```

```
kubectl port-forward service/kong-dp-status -n kong-dp 8100
```

```
$ http :8100/metrics | grep kong_http_status
# HELP kong_http_status HTTP status codes per service/route in Kong
# TYPE kong_http_status counter
kong_http_status{service="default.route1-ext.80",route="default.route1.00"
,code="200"} 9497
```

Add a new Konvoy Prometheus Service Monitor

```
cat <<EOF | kubectl apply -f -
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: kong-service-monitor
  namespace: kong-dp
  labels:
    release: prometheus-kubeaddons
spec:
  selector:
    matchLabels:
      app: kong-dp-status
  endpoints:
  - port: metrics
EOF
```

Change the cluster.yaml file add the new Service Monitor. Replace

```
- name: prometheus
  enabled: true
```

with

```
- name: prometheus
  enabled: true
  values: |
    prometheus:
      additionalServiceMonitors:
        - name: kong-service-monitor
          selector:
            matchLabels:
              app: kong-dp-status
          namespaceSelector:
            matchNames:
              - kong-dp
          endpoints:
            - port: metrics
              interval: 30s
```

Apply the new `cluster.yaml`

```
konvoy deploy addons

…….
Addons deployed successfully!

Run `./konvoy apply kubeconfig` to update kubectl credentials.

Navigate to the URL below to access various services running in the
cluster.

https://a3327e66f1f6b499994cd6a8632b0328-324595487.eu-central-1.elb.amazon
aws.com/ops/landing
And login using the credentials below.
  Username: zealous_lamport
  Password:
lXD12s4vxwFxfbHNStQ3LBooJAemULm0BBEo3DLGa5vGG1TzDqUQS8ID1FVEJIwD

If the cluster was recently created, the dashboard and services may take a
few minutes to be accessible.
```

Checking the new Kong Data Plane Metrics in Konvoy Prometheus

```
kubectl port-forward service/prometheus-operated -n kubeaddons 9090
```

Get the API consumption rate

```
$ curl -gs
'http://localhost:9090/api/v1/query?query=sum(rate(kong_http_status{code="
200"}[1m]))' | jq -r .data.result[0].value[1]
6023.209105110467
```

Get the number of successful processed requests

```
curl -gs
'http://localhost:9090/api/v1/query?query=kong_http_status{code="200"}' |
jq -r .data.result[0].value[1]
81022768
```

Prometheus Snapshot

https://prometheus.io/docs/prometheus/latest/querying/api/#snapshot

Create a Snapshot

```
$ http post http://localhost:9090/api/v1/admin/tsdb/snapshot
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Length: 98
Content-Type: application/json
Date: Fri, 25 Jun 2021 22:16:48 GMT

{
    "data": {
        "name": "20210625T221643Z-105f7527c2603641"
    },
    "status": "success"
}
```

Copy the Snapshot to local directory

```
kubectl -n kubeaddons exec -it
prometheus-prometheus-kubeaddons-prom-prometheus-0 -c prometheus --
/bin/sh -c "ls /prometheus/snapshots/20210625T221643Z-105f7527c2603641"
01F92KY8YSV4GHPSSC1TF18JKC
```

Create a `snapshots` directory and run the following inside of it:

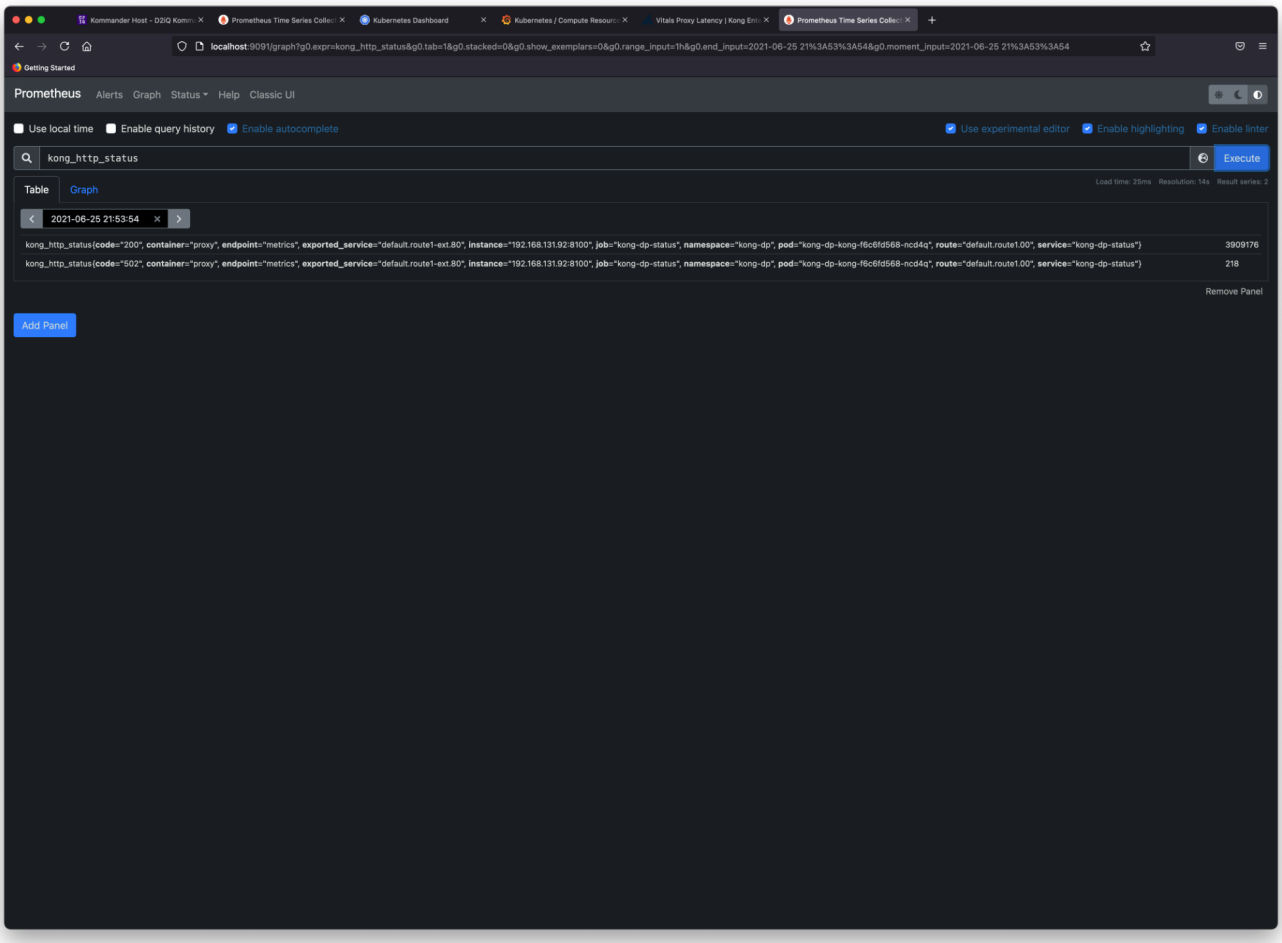
```
kubectl cp -n kubeaddons
prometheus-prometheus-kubeaddons-prom-prometheus-0:/prometheus/snapshots/2
0210625T221643Z-105f7527c2603641/01F92KY8YSV4GHPSSC1TF18JKC -c prometheus
./01F92KY8YSV4GHPSSC1TF18JKC
```

Local Prometheus Docker Installation

https://prometheus.io/docs/introduction/overview/
https://prometheus.io/docs/prometheus/latest/installation/
https://hub.docker.com/r/prom/prometheus

```
docker stop prometheus
docker container rm prometheus -v
docker image rm prom/prometheus
docker network rm kong-net

docker network create kong-net

docker run --network=kong-net --name=prometheus -d -p 9091:9090 -v
/Users/claudio/kong/Tech/D2IQ/certification/konvoy_v1.7.3/snapshots:/prome
theus -v
/Users/claudio/kong/Tech/D2IQ/certification/konvoy_v1.7.3/prometheus-local
.yml:/etc/prometheus/prometheus.yml:ro prom/prometheus
--config.file=/etc/prometheus/prometheus.yml
--storage.tsdb.path=/prometheus
```

Redirect your browser to http://localhost:9091. You should be able to submit some queries against the Prometheus Snapshot:



Kong Vitals

```
$ http
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com:
8001/services/default.route1-ext.80/routes | jq -r .data[0].id
c6efd1da-40fd-4716-b29e-5d39fb97ba69
```

Get the Unix Timestamp

```
date '+%s'
1624658605
```

```
http
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com:
8001/vitals/status_codes/by_route
```

```
route_id=c6efd1da-40fd-4716-b29e-5d39fb97ba69 interval=seconds
start_ts=1624658526

$ http
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com:
8001/default/vitals/nodes?start_ts=1624658605\&interval=minutes
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 571
Content-Type: application/json; charset=utf-8
Date: Fri, 25 Jun 2021 22:04:27 GMT
Server: kong/2.2.1.3-enterprise-edition
X-Kong-Admin-Latency: 4
X-Kong-Admin-Request-ID: HnDlYRii4ul0a3oAfyIh2XQnpVTpo2uJ
vary: Origin
vary: Origin

{
    "meta": {
        "earliest_ts": 1624658640,
        "interval": "minutes",
        "interval_width": 60,
        "latest_ts": 1624658640,
        "level": "node",
        "nodes": {
            "24f3829d-3d59-4d05-a166-eb4f392ad8e0": {
                "hostname": "kong-dp-kong-f6c6fd568-ncd4q"
            }
        },
        "stat_labels": [
            "cache_datastore_hits_total",
            "cache_datastore_misses_total",
            "latency_proxy_request_min_ms",
            "latency_proxy_request_max_ms",
            "latency_upstream_min_ms",
            "latency_upstream_max_ms",
            "requests_proxy_total",
            "latency_proxy_request_avg_ms",
            "latency_upstream_avg_ms"
        ]
    },
```

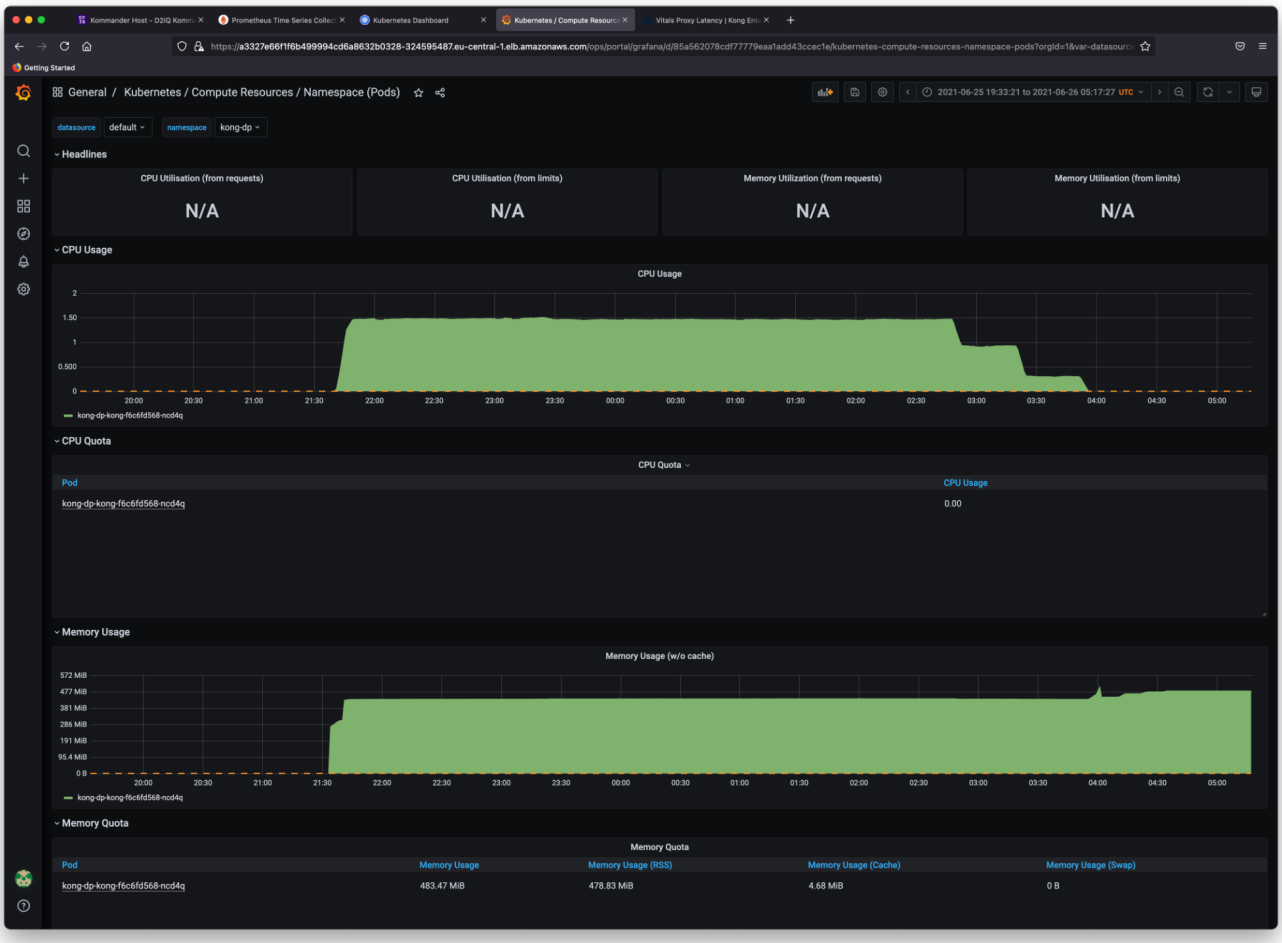
```
    "stats": {
        "24f3829d-3d59-4d05-a166-eb4f392ad8e0": {
            "1624658640": [
                0,
                0,
                0,
                16,
                89,
                759,
                113257,
                0,
                99
            ]
        }
    }
}
```

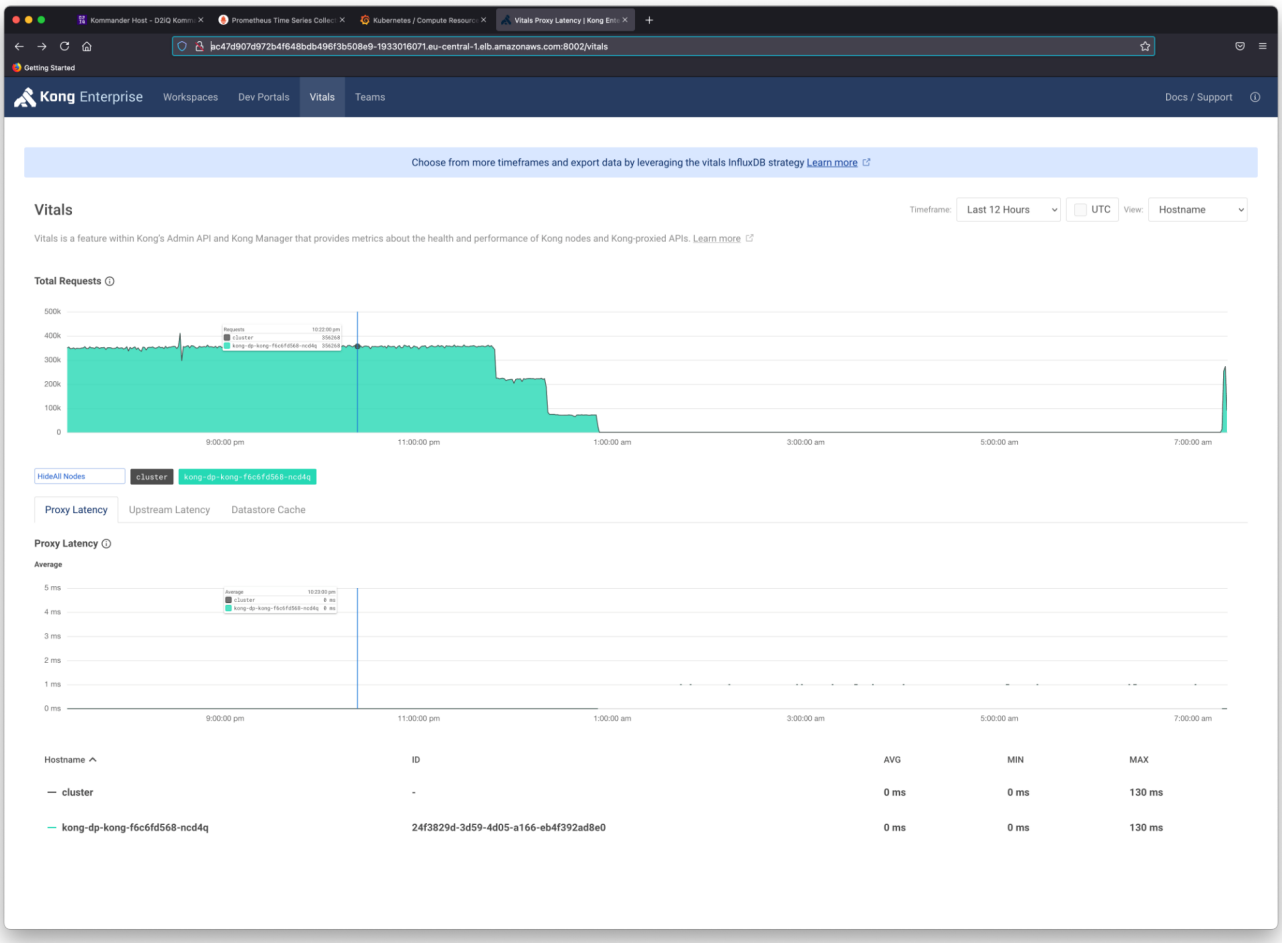

Snapshots 2

```
$ http post http://localhost:9090/api/v1/admin/tsdb/snapshot
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Length: 97
Content-Type: application/json
Date: Sat, 26 Jun 2021 10:21:34 GMT

{
    "data": {
        "name": "20210626T102130Z-728a4a821d0b7578"
    },
    "status": "success"
}


$ kubectl -n kubeaddons exec -it
prometheus-prometheus-kubeaddons-prom-prometheus-0 -c prometheus --
/bin/sh -c "ls /prometheus/snapshots/"
20210625T221643Z-105f7527c2603641  20210626T102130Z-728a4a821d0b7578
```

#successful processed requests

```
$ curl -gs
'http://localhost:9090/api/v1/query?query=kong_http_status{code="200"}' |
jq -r .data.result[0].value[1]
119021828
```



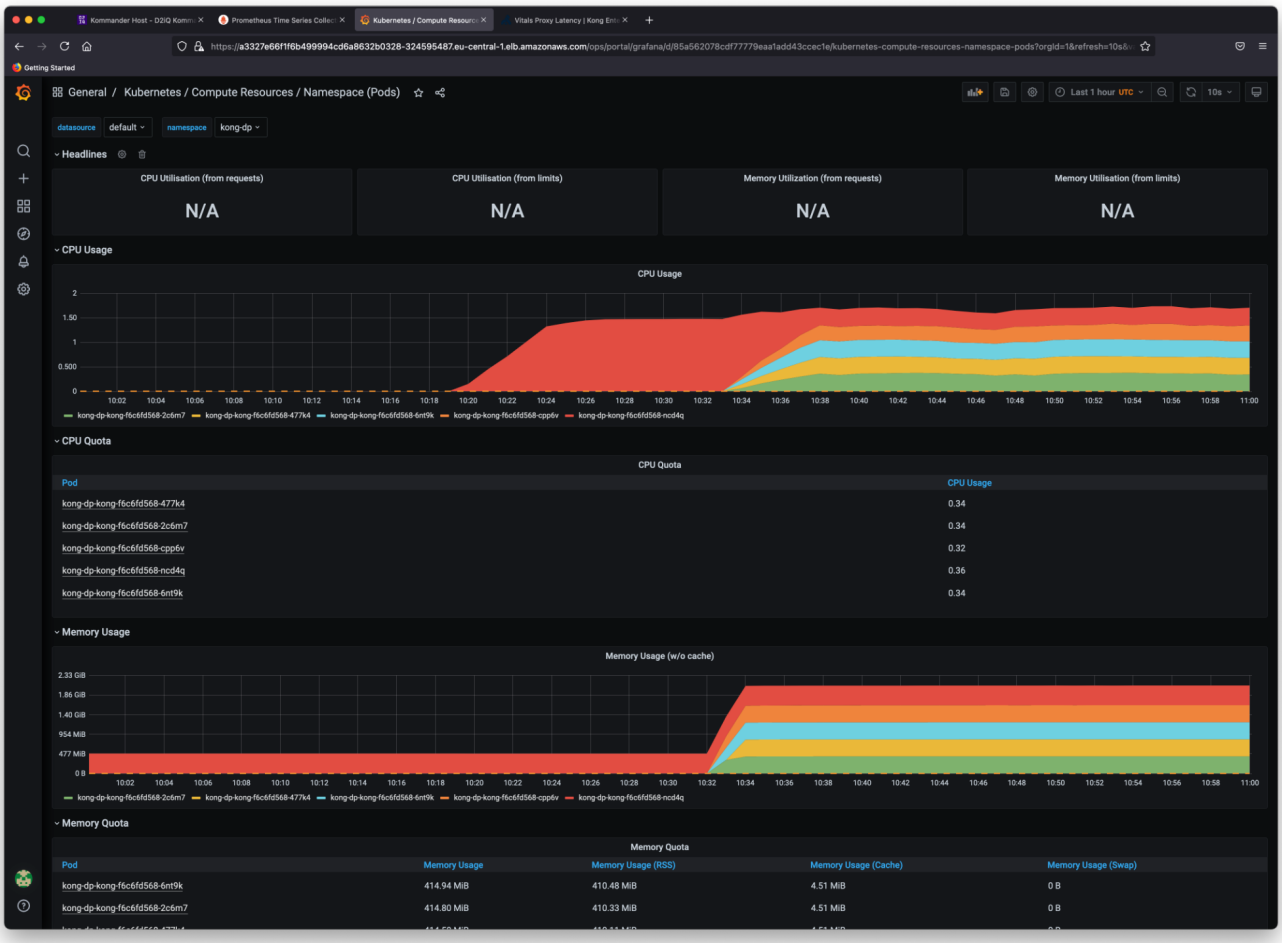

Kong Enterprise
Workspaces
Dev Portals
Vitals
Teams
Docs / Support
Choose from more timeframes and export data by leveraging the vitals InfluxDB strategy Learn more
Vitals
Timeframe: Last 12 Hours
UTC
View: Hostname
Vitals is a feature within Kong's Admin API and Kong Manager that provides metrics about the health and performance of Kong nodes and Kong-proxied APIs. Learn more
Total Requests
500k
400k
300k
200k
100k
0
9:00:00 pm
11:00:00 pm
1:00:00 am
3:00:00 am
5:00:00 am
7:00:00 am
HideAll Nodes
cluster
kong-dp-kong-f6c6fd568-ncd4q
Proxy Latency
Upstream Latency
Datastore Cache
Proxy Latency
Average
5 ms
4 ms
3 ms
2 ms
1 ms
0 ms
9:00:00 pm
11:00:00 pm
1:00:00 am
3:00:00 am
5:00:00 am
7:00:00 am
Hostname
ID
AVG
MIN
MAX
cluster
-
0 ms
0 ms
130 ms
kong-dp-kong-f6c6fd568-ncd4q
24f3829d-3d59-4d05-a166-eb4f392ad8e0
0 ms
0 ms
130 ms

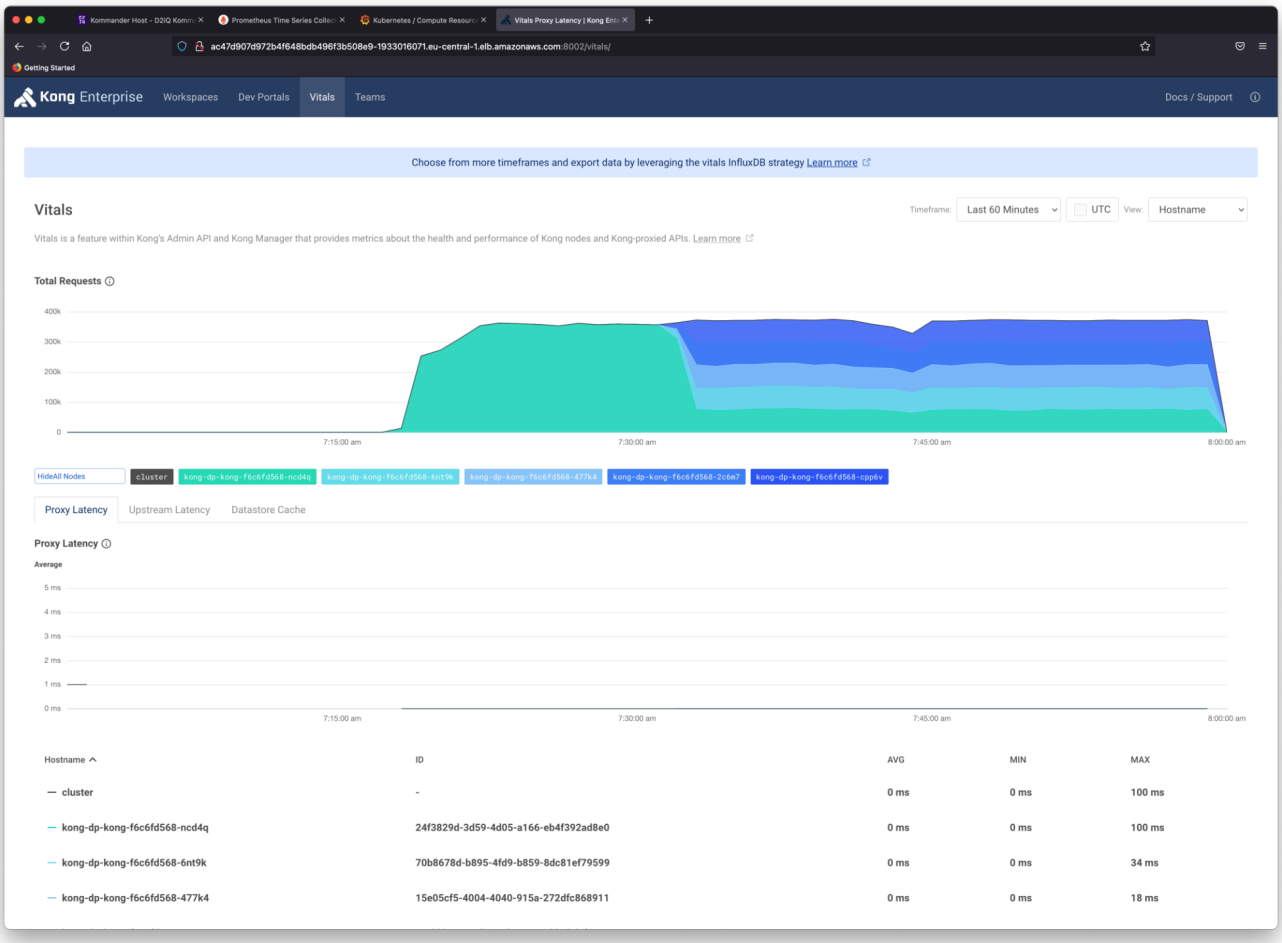
Scaling the Deployment

```
kubectl scale deployment.v1.apps/kong-dp-kong -n kong-dp --replicas=5
```

```
$ kubectl get pod -n kong-dp
NAME                             READY   STATUS    RESTARTS   AGE
kong-dp-kong-f6c6fd568-2c6m7     1/1     Running   0          5m43s
kong-dp-kong-f6c6fd568-477k4     1/1     Running   0          5m43s
kong-dp-kong-f6c6fd568-6nt9k     1/1     Running   0          5m43s
kong-dp-kong-f6c6fd568-cpp6v     1/1     Running   0          5m43s
kong-dp-kong-f6c6fd568-ncd4q     1/1     Running   0          13h
```

```
$ curl -gs
'http://localhost:9090/api/v1/query?query=kong_http_status{code="200"}' |
jq -r .data.result[0].value[1]
2002208
```

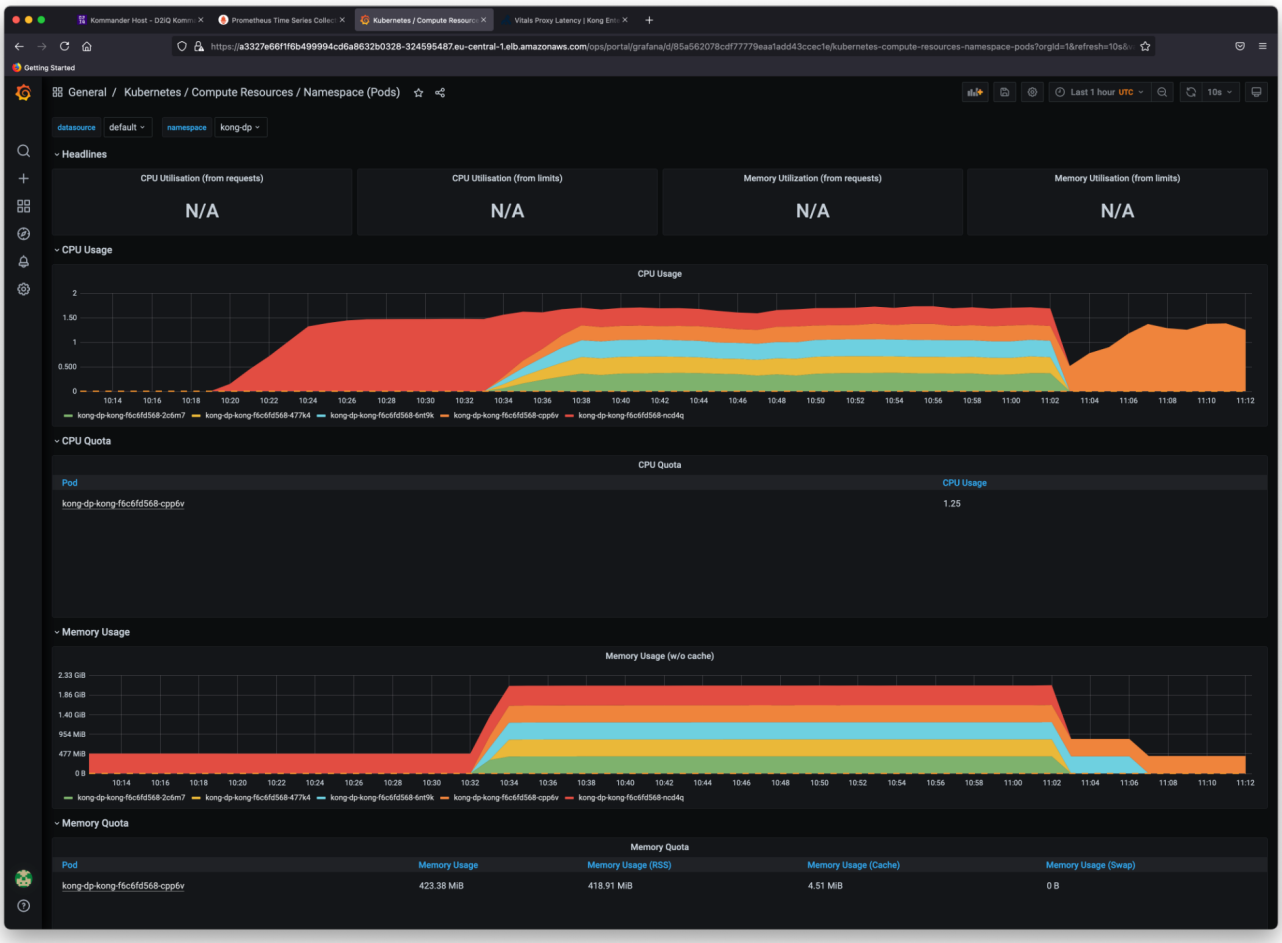


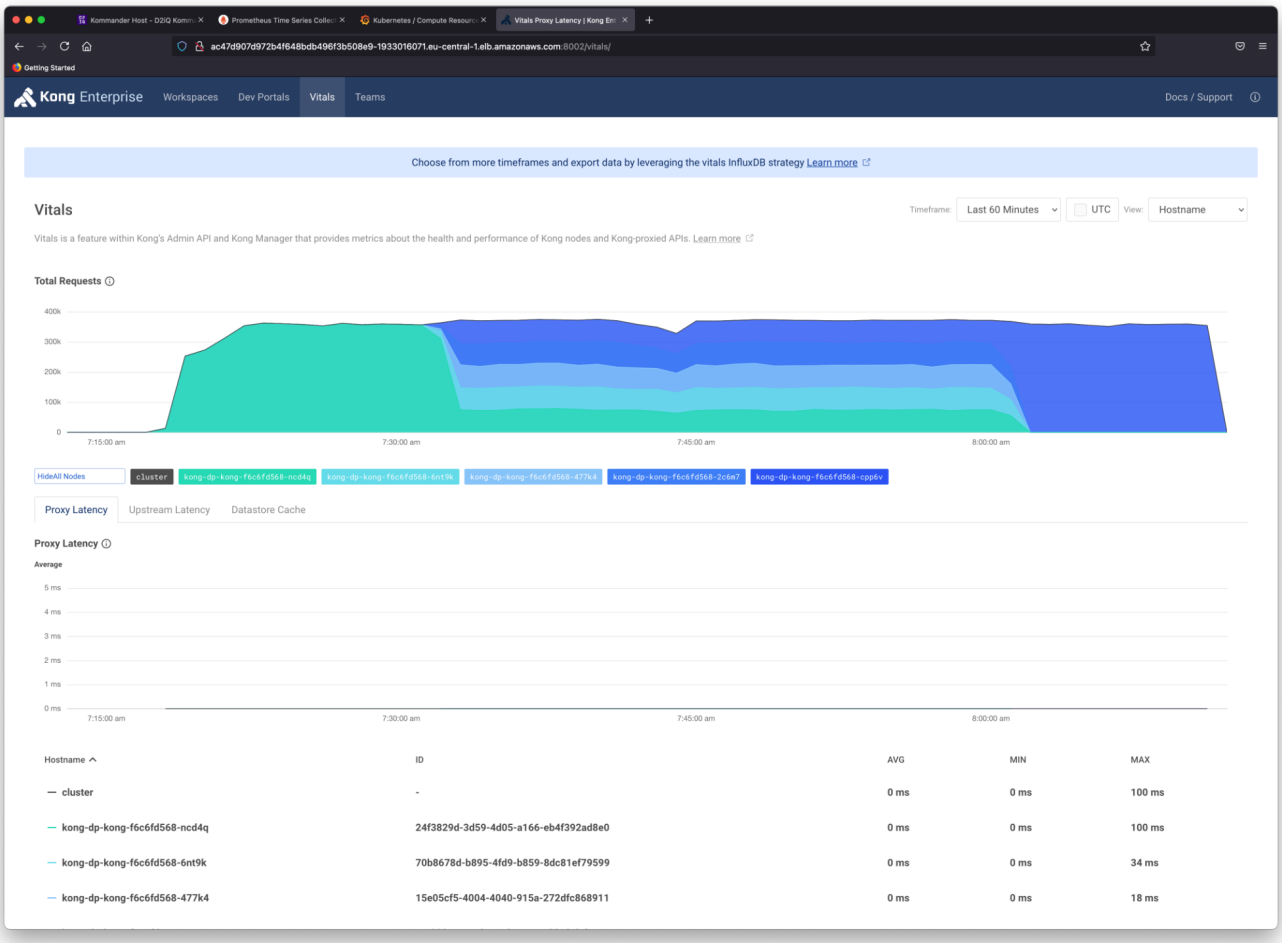



```
$ http post http://localhost:9090/api/v1/admin/tsdb/snapshot
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Length: 96
Content-Type: application/json
Date: Sat, 26 Jun 2021 11:01:07 GMT

{
    "data": {
        "name": "20210626T110104Z-19700fe91250cbb1"
    },
    "status": "success"
}
```

Reduce to number of replicas to 1 again

```
kubectl scale deployment.v1.apps/kong-dp-kong -n kong-dp --replicas=1
```


General / Kubernetes / Compute Resources / Namespace (Pods)
datasource default
namespace kong-dp
Headlines
CPU Utilisation (from requests)
N/A
CPU Utilisation (from limits)
N/A
Memory Utilization (from requests)
N/A
Memory Utilisation (from limits)
N/A
CPU Usage
CPU Usage
CPU Quota
CPU Quota
Pod
CPU Usage
kong-dp-kong-f6c6fd568-cpp6v
1.25
Memory Usage
Memory Usage (w/o cache)
Memory Quota
Memory Quota
Pod
Memory Usage
Memory Usage (RSS)
Memory Usage (Cache)
Memory Usage (Swap)
kong-dp-kong-f6c6fd568-cpp6v
423.38 MiB
418.91 MiB
4.51 MiB
0 B


ac47d907d972b4f648bdb496f3b508e9-1933016071.eu-central-1.elb.amazonaws.com:8002/vitals/
Kong Enterprise
Workspaces
Dev Portals
Vitals
Teams
Docs / Support
Choose from more timeframes and export data by leveraging the vitals InfluxDB strategy Learn more
Vitals
Timeframe: Last 60 Minutes
UTC
View: Hostname
Vitals is a feature within Kong's Admin API and Kong Manager that provides metrics about the health and performance of Kong nodes and Kong-proxied APIs. Learn more
Total Requests
400k
300k
200k
100k
0
7:15:00 am
7:30:00 am
7:45:00 am
8:00:00 am
HideAll Nodes
cluster
kong-dp-kong-f6c6fd568-ncd4q
kong-dp-kong-f6c6fd568-6nt9k
kong-dp-kong-f6c6fd568-477k4
kong-dp-kong-f6c6fd568-2c6m7
kong-dp-kong-f6c6fd568-cpp6v
Proxy Latency
Upstream Latency
Datastore Cache
Proxy Latency
Average
5 ms
4 ms
3 ms
2 ms
1 ms
0 ms
7:15:00 am
7:30:00 am
7:45:00 am
8:00:00 am
Hostname
ID
AVG
MIN
MAX
cluster
-
0 ms
0 ms
100 ms
kong-dp-kong-f6c6fd568-ncd4q
24f3829d-3d59-4d05-a166-eb4f392ad8e0
0 ms
0 ms
100 ms
kong-dp-kong-f6c6fd568-6nt9k
70b8678d-b895-4fd9-b859-8dc81ef79599
0 ms
0 ms
34 ms
kong-dp-kong-f6c6fd568-477k4
15e05cf5-4004-4040-915a-272dfc868911
0 ms
0 ms
18 ms

Kong Konnect Enterprise Upgrade

https://docs.konghq.com/enterprise/2.4.x/deployment/upgrades/migrations/
https://docs.konghq.com/kubernetes-ingress-controller/1.3.x/references/version-compatibility/

Upgrade to Kong Gateway 2.3.3.2

Control Plane

```
helm upgrade kong kong/kong -n kong \
--set ingressController.enabled=true \
--set ingressController.installCRDs=false \
--set
ingressController.image.repository=kong/kubernetes-ingress-controller \
--set ingressController.image.tag=1.2.0 \
--set image.repository=kong/kong-gateway \
--set image.tag=2.3.3.2-alpine \
--set env.database=postgres \
--set env.role=control_plane \
--set env.cluster_cert=/etc/secrets/kong-cluster-cert/tls.crt \
--set env.cluster_cert_key=/etc/secrets/kong-cluster-cert/tls.key \
--set cluster.enabled=true \
--set cluster.tls.enabled=true \
--set cluster.tls.servicePort=8005 \
--set cluster.tls.containerPort=8005 \
--set clustertelemetry.enabled=true \
--set clustertelemetry.tls.enabled=true \
--set clustertelemetry.tls.servicePort=8006 \
--set clustertelemetry.tls.containerPort=8006 \
--set proxy.enabled=true \
--set admin.enabled=true \
--set admin.http.enabled=true \
--set admin.type=LoadBalancer \
--set admin.labels.enable-metrics=true \
--set enterprise.enabled=true \
--set enterprise.license_secret=kong-enterprise-license \
--set enterprise.portal.enabled=false \
--set enterprise.rbac.enabled=false \
--set enterprise.smtp.enabled=false \
```

```
--set manager.enabled=true \
--set manager.type=LoadBalancer \
--set secretVolumes[0]=kong-cluster-cert \
--set postgresql.enabled=true \
--set postgresql.postgresqlUsername=kong \
--set postgresql.postgresqlDatabase=kong \
--set postgresql.postgresqlPassword=kong
```

```
$ kubectl get pod -n kong
NAME                                    READY   STATUS      RESTARTS   AGE
kong-kong-57b78b56c9-8jl7q              2/2     Running     0          15h
kong-kong-6d59cb8bf9-xbgpd              1/2     Running     0          14s
kong-kong-post-upgrade-migrations-xql54 0/1     Completed   0          12s
kong-kong-pre-upgrade-migrations-nqznn  0/1     Completed   0          40s
kong-postgresql-0                       1/1     Running     0          15h
```

```
$ kubectl get service -n kong
NAME                        TYPE           CLUSTER-IP    EXTERNAL-IP
PORT(S)                        AGE
kong-kong-admin             LoadBalancer   10.0.4.111
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com
8001:31889/TCP,8444:30857/TCP   15h
kong-kong-cluster           ClusterIP      10.0.41.177   <none>
8005/TCP                       15h
kong-kong-clustertelemetry  ClusterIP      10.0.55.196   <none>
8006/TCP                       15h
kong-kong-manager           LoadBalancer   10.0.4.37
ac47d907d972b4f648bdb496f3b508e9-1933016071.eu-central-1.elb.amazonaws.com
8002:30219/TCP,8445:30537/TCP   15h
kong-kong-portal            NodePort       10.0.3.175    <none>
8003:30328/TCP,8446:31468/TCP   15h
kong-kong-portalapi         NodePort       10.0.23.193   <none>
8004:32019/TCP,8447:30673/TCP   15h
kong-kong-proxy             LoadBalancer   10.0.26.106
ab27be50edb204d4290648d128fddb30-1912033636.eu-central-1.elb.amazonaws.com
80:31290/TCP,443:30591/TCP      15h
kong-postgresql             ClusterIP      10.0.0.8      <none>
5432/TCP                       15h
kong-postgresql-headless    ClusterIP      None          <none>
5432/TCP                       15h
```

Checking the Admin

```
$ http
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com:
8001 | jq .version
"2.3.3.2-enterprise-edition"
```

Data Plane

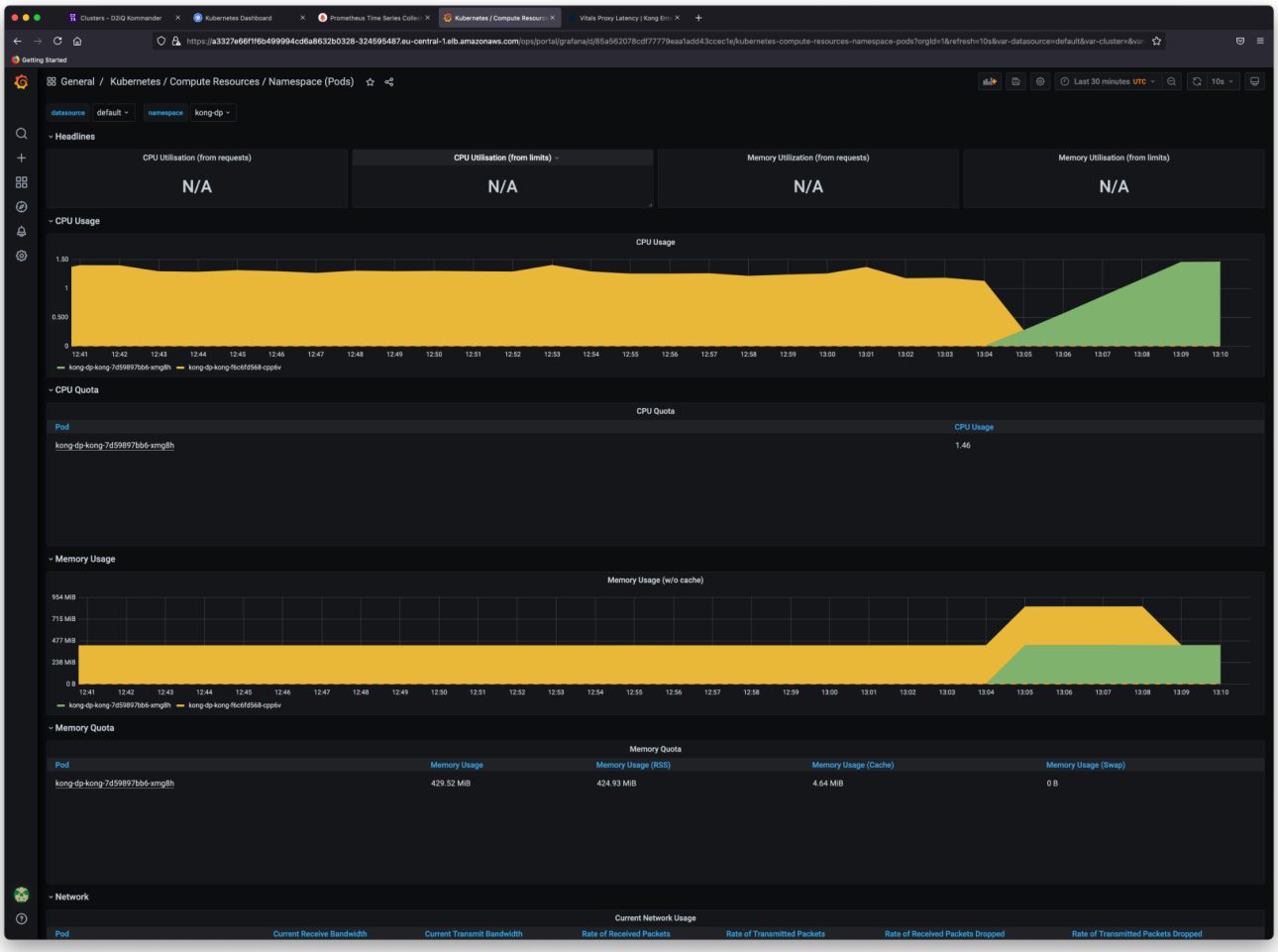
```
helm upgrade kong-dp kong/kong -n kong-dp \
--set ingressController.enabled=false \
--set image.repository=kong/kong-gateway \
--set image.tag=2.3.3.2-alpine \
--set env.database=off \
--set env.role=data_plane \
--set env.cluster_cert=/etc/secrets/kong-cluster-cert/tls.crt \
--set env.cluster_cert_key=/etc/secrets/kong-cluster-cert/tls.key \
--set
env.lua_ssl_trusted_certificate=/etc/secrets/kong-cluster-cert/tls.crt \
--set
env.cluster_control_plane=kong-kong-cluster.kong.svc.cluster.local:8005 \
--set
env.cluster_telemetry_endpoint=kong-kong-clustertelemetry.kong.svc.cluster
.local:8006 \
--set proxy.enabled=true \
--set proxy.type=LoadBalancer \
--set enterprise.enabled=true \
--set enterprise.license_secret=kong-enterprise-license \
--set enterprise.portal.enabled=false \
--set enterprise.rbac.enabled=false \
--set enterprise.smtp.enabled=false \
--set manager.enabled=false \
--set portal.enabled=false \
--set portalapi.enabled=false \
--set env.status_listen=0.0.0.0:8100 \
--set secretVolumes[0]=kong-cluster-cert
```

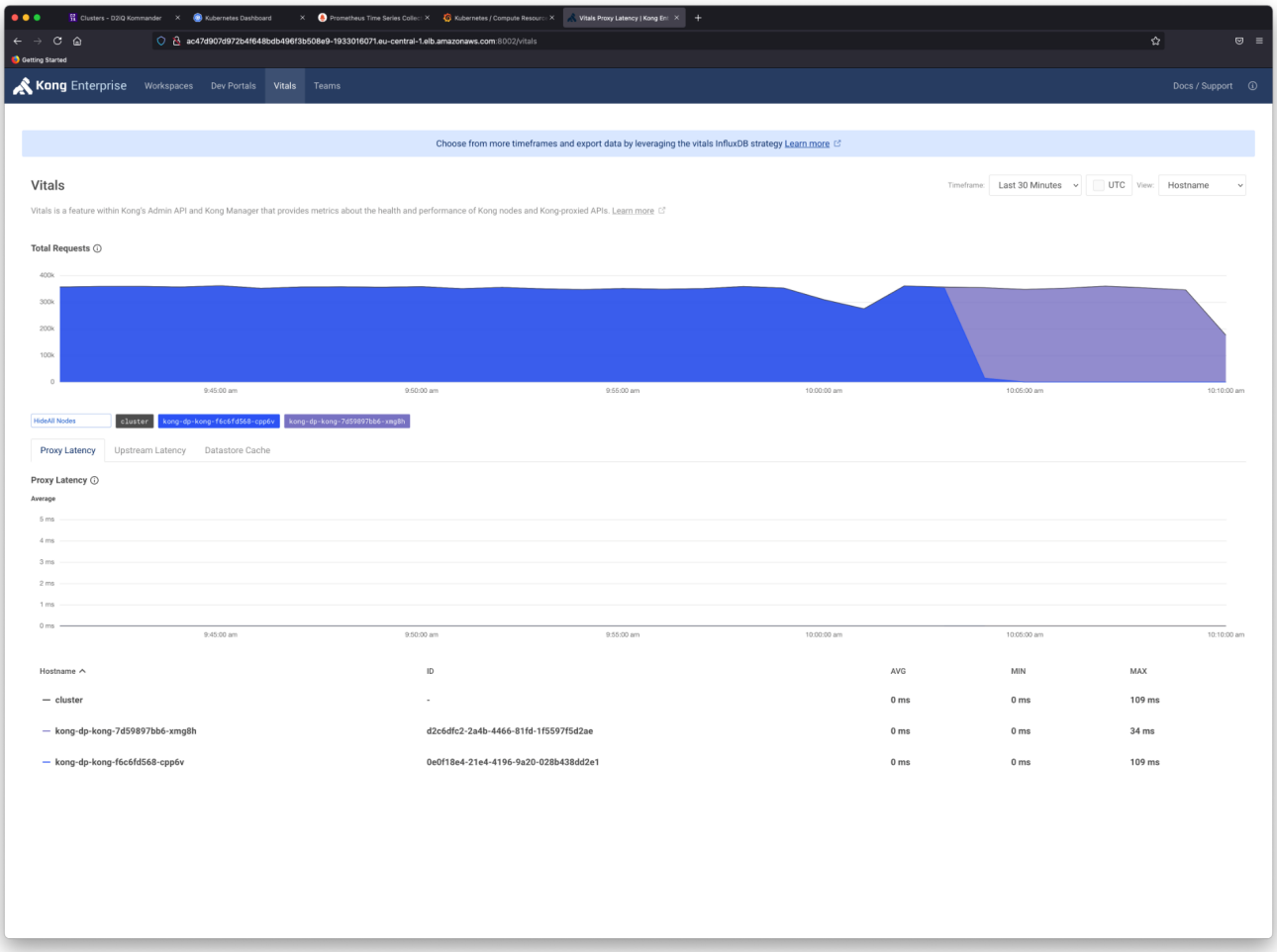
```
$ kubectl get pod -n kong-dp
NAME                           READY   STATUS    RESTARTS   AGE
kong-dp-kong-7d59897bb6-xmg8h   1/1     Running   0          3m47s

$ kubectl get service -n kong-dp
NAME                 TYPE           CLUSTER-IP    EXTERNAL-IP
PORT(S)                      AGE
kong-dp-kong-proxy   LoadBalancer   10.0.48.95
af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.com
80:32025/TCP,443:31228/TCP   15h
kong-dp-status       ClusterIP      10.0.62.128   <none>
8100/TCP                     15h
```

New Ingress Creation and Consumption

```
cat <<EOF | kubectl apply -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: newroute
  namespace: default
  annotations:
    konghq.com/strip-path: "true"
    kubernetes.io/ingress.class: kong
spec:
  rules:
  - http:
      paths:
        - path: /newroute
          backend:
            serviceName: route1-ext
            servicePort: 80
EOF


$ http
af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.com
/newroute/get
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 620
Content-Type: application/json
Date: Sat, 26 Jun 2021 13:05:45 GMT
Server: gunicorn/19.9.0
Via: kong/2.3.3.2-enterprise-edition
X-Kong-Proxy-Latency: 0
X-Kong-Upstream-Latency: 95

{
    "args": {},
    "headers": {
```

```
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Host":
"af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.co
m",
        "User-Agent": "HTTPie/2.4.0",
        "X-Amzn-Trace-Id": "Root=1-60d72629-044cebf64516728d46e56996",
        "X-Forwarded-Host":
"af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.co
m",
        "X-Forwarded-Path": "/newroute/get",
        "X-Forwarded-Prefix": "/newroute"
    },
    "origin": "192.168.196.0, 3.67.184.22",
    "url":
"http://af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazo
naws.com/get"
}


kubectl delete ingress newroute
```


General / Kubernetes / Compute Resources / Namespace (Pods)
datasource default
namespace kong-dp
Headlines
CPU Utilisation (from requests)
N/A
CPU Utilisation (from limits)
N/A
Memory Utilization (from requests)
N/A
Memory Utilisation (from limits)
N/A
CPU Usage
CPU Usage
CPU Quota
CPU Quota
Pod
CPU Usage
kong-dp-kong-7d59897bb6-xmg8h
1.46
Memory Usage
Memory Usage (w/o cache)
Memory Quota
Memory Quota
Pod
Memory Usage
Memory Usage (RSS)
Memory Usage (Cache)
Memory Usage (Swap)
kong-dp-kong-7d59897bb6-xmg8h
429.52 MiB
424.93 MiB
4.64 MiB
0 B
Network
Current Network Usage
Pod
Current Receive Bandwidth
Current Transmit Bandwidth
Rate of Received Packets
Rate of Transmitted Packets
Rate of Received Packets Dropped
Rate of Transmitted Packets Dropped



Upgrade to Kong Gateway 2.4.1.1 and Kong Ingress Controller 1.3.1

Control Plane

```
helm upgrade kong kong/kong -n kong \
--set ingressController.enabled=true \
--set ingressController.installCRDs=false \
--set
ingressController.image.repository=kong/kubernetes-ingress-controller \
--set ingressController.image.tag=1.3.1-alpine \
--set image.repository=kong/kong-gateway \
```

```
--set image.tag=2.4.1.1-alpine \
--set env.database=postgres \
--set env.role=control_plane \
--set env.cluster_cert=/etc/secrets/kong-cluster-cert/tls.crt \
--set env.cluster_cert_key=/etc/secrets/kong-cluster-cert/tls.key \
--set cluster.enabled=true \
--set cluster.tls.enabled=true \
--set cluster.tls.servicePort=8005 \
--set cluster.tls.containerPort=8005 \
--set clustertelemetry.enabled=true \
--set clustertelemetry.tls.enabled=true \
--set clustertelemetry.tls.servicePort=8006 \
--set clustertelemetry.tls.containerPort=8006 \
--set proxy.enabled=true \
--set admin.enabled=true \
--set admin.http.enabled=true \
--set admin.type=LoadBalancer \
--set admin.labels.enable-metrics=true \
--set enterprise.enabled=true \
--set enterprise.license_secret=kong-enterprise-license \
--set enterprise.portal.enabled=false \
--set enterprise.rbac.enabled=false \
--set enterprise.smtp.enabled=false \
--set manager.enabled=true \
--set manager.type=LoadBalancer \
--set secretVolumes[0]=kong-cluster-cert \
--set postgresql.enabled=true \
--set postgresql.postgresqlUsername=kong \
--set postgresql.postgresqlDatabase=kong \
--set postgresql.postgresqlPassword=kong
```

```
$ kubectl get pod -n kong
NAME                                    READY   STATUS        RESTARTS   AGE
kong-kong-64cdcf65df-fxz2r              2/2     Running       0          45s
kong-kong-post-upgrade-migrations-cv5kz 0/1     Completed     0          44s
kong-kong-pre-upgrade-migrations-rc4jx  0/1     Completed     0          70s
kong-postgresql-0                       1/1     Running       0          15h
```

```
$ kubectl get service -n kong
NAME                         TYPE           CLUSTER-IP    EXTERNAL-IP
PORT(S)                         AGE
```

```
kong-kong-admin              LoadBalancer   10.0.4.111
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com
8001:31889/TCP,8444:30857/TCP   15h
kong-kong-cluster            ClusterIP      10.0.41.177   <none>
8005/TCP                        15h
kong-kong-clustertelemetry   ClusterIP      10.0.55.196   <none>
8006/TCP                        15h
kong-kong-manager            LoadBalancer   10.0.4.37
ac47d907d972b4f648bdb496f3b508e9-1933016071.eu-central-1.elb.amazonaws.com
8002:30219/TCP,8445:30537/TCP   15h
kong-kong-portal             NodePort       10.0.3.175    <none>
8003:30328/TCP,8446:31468/TCP   15h
kong-kong-portalapi          NodePort       10.0.23.193   <none>
8004:32019/TCP,8447:30673/TCP   15h
kong-kong-proxy              LoadBalancer   10.0.26.106
ab27be50edb204d4290648d128fddb30-1912033636.eu-central-1.elb.amazonaws.com
80:31290/TCP,443:30591/TCP      15h
kong-postgresql              ClusterIP      10.0.0.8      <none>
5432/TCP                        15h
kong-postgresql-headless     ClusterIP      None          <none>
5432/TCP                        15h
```

Checking the Admin

```
$ http
a62c55481cf11447f8f631dada1cc961-193326851.eu-central-1.elb.amazonaws.com:
8001 | jq .version
"2.4.1.1-enterprise-edition"
```

Data Plane

```
helm upgrade kong-dp kong/kong -n kong-dp \
--set ingressController.enabled=false \
--set image.repository=kong/kong-gateway \
--set image.tag=2.4.1.1-alpine \
--set env.database=off \
--set env.role=data_plane \
--set env.cluster_cert=/etc/secrets/kong-cluster-cert/tls.crt \
--set env.cluster_cert_key=/etc/secrets/kong-cluster-cert/tls.key \
--set
env.lua_ssl_trusted_certificate=/etc/secrets/kong-cluster-cert/tls.crt \
--set
env.cluster_control_plane=kong-kong-cluster.kong.svc.cluster.local:8005 \
```

```
--set
env.cluster_telemetry_endpoint=kong-kong-clustertelemetry.kong.svc.cluster
.local:8006 \
--set proxy.enabled=true \
--set proxy.type=LoadBalancer \
--set enterprise.enabled=true \
--set enterprise.license_secret=kong-enterprise-license \
--set enterprise.portal.enabled=false \
--set enterprise.rbac.enabled=false \
--set enterprise.smtp.enabled=false \
--set manager.enabled=false \
--set portal.enabled=false \
--set portalapi.enabled=false \
--set env.status_listen=0.0.0.0:8100 \
--set secretVolumes[0]=kong-cluster-cert
```

```
$ kubectl get pod -n kong-dp
NAME                           READY   STATUS        RESTARTS   AGE
kong-dp-kong-5fcc66c796-bh7kx   1/1     Running       0          27s

$ kubectl get service -n kong-dp
NAME                 TYPE           CLUSTER-IP    EXTERNAL-IP
PORT(S)                      AGE
kong-dp-kong-proxy   LoadBalancer   10.0.48.95
af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.com
80:32025/TCP,443:31228/TCP   15h
kong-dp-status       ClusterIP      10.0.62.128   <none>
8100/TCP                     15h
```

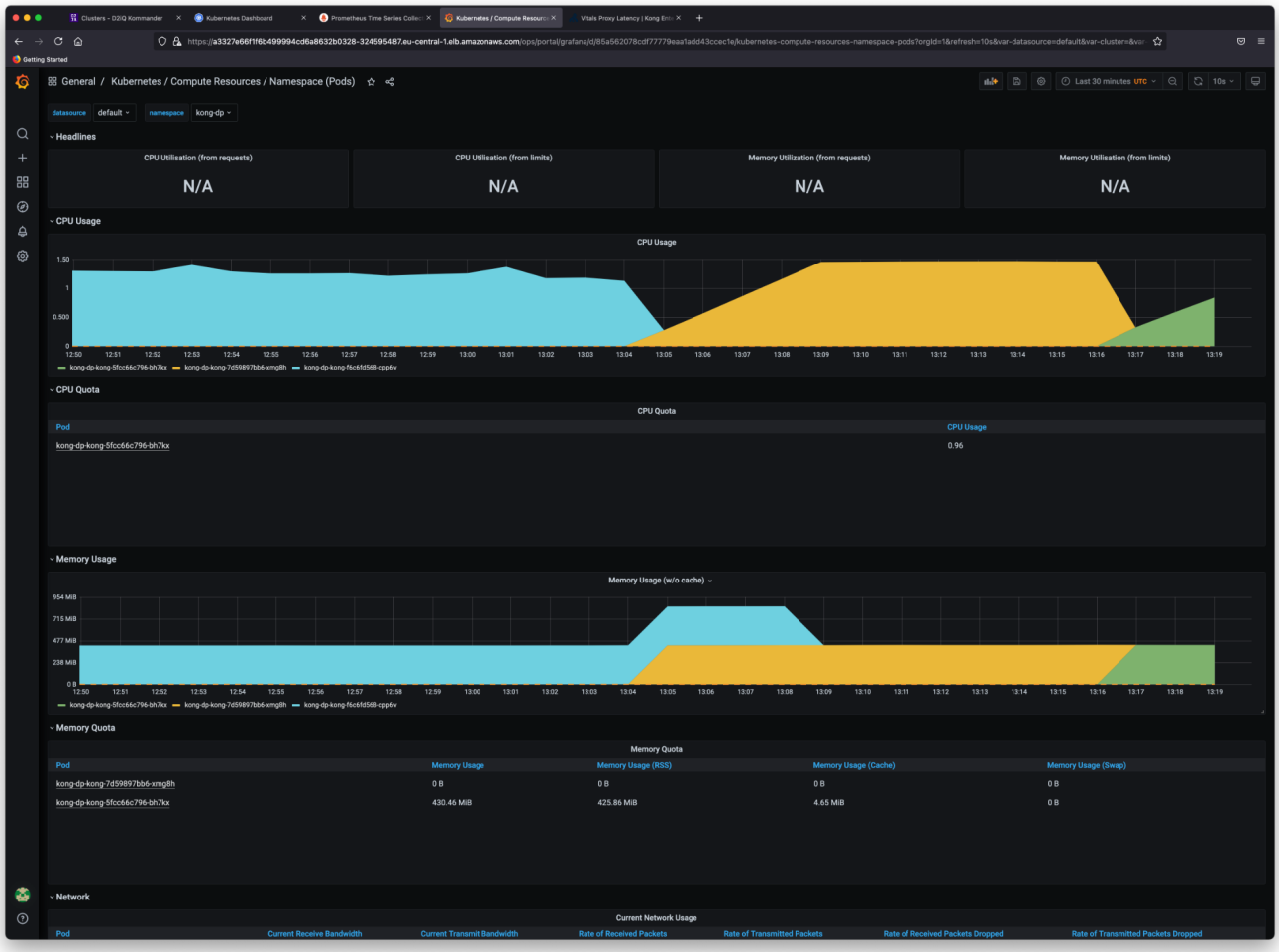
New Ingress Creation and Consumption

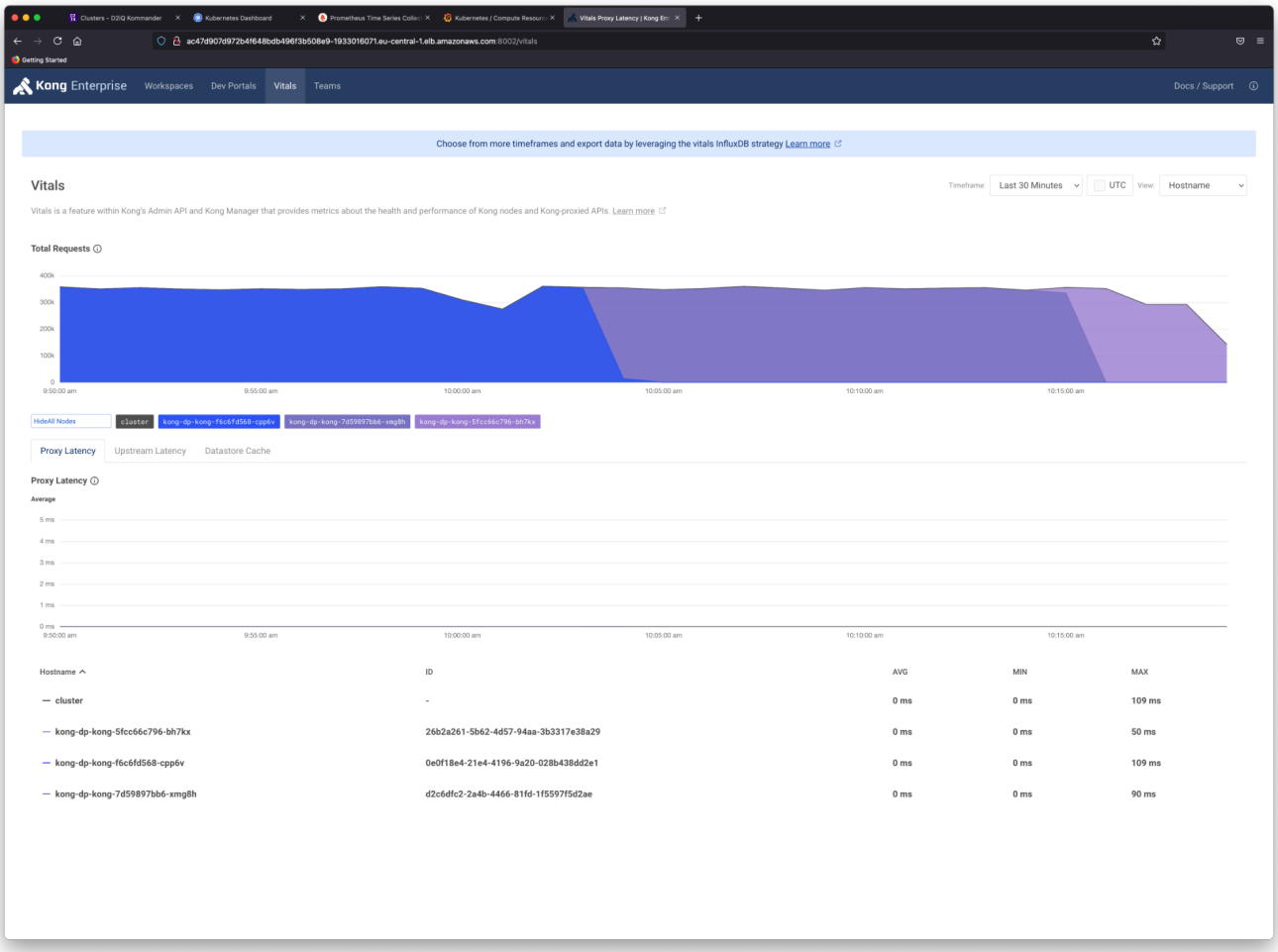
```
cat <<EOF | kubectl apply -f -
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: newroute
  namespace: default
  annotations:
    konghq.com/strip-path: "true"
    kubernetes.io/ingress.class: kong
spec:
  rules:
  - http:
```

```
      paths:
        - path: /newroute
          backend:
            serviceName: route1-ext
            servicePort: 80
EOF


$ http
af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.com
/newroute/get
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 619
Content-Type: application/json
Date: Sat, 26 Jun 2021 13:16:49 GMT
Server: gunicorn/19.9.0
Via: kong/2.4.1.1-enterprise-edition
X-Kong-Proxy-Latency: 0
X-Kong-Upstream-Latency: 91

{
    "args": {},
    "headers": {
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Host":
"af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.co
m",
        "User-Agent": "HTTPie/2.4.0",
        "X-Amzn-Trace-Id": "Root=1-60d728c1-4f06dacc73cd37e1794d4e51",
        "X-Forwarded-Host":
"af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazonaws.co
m",
        "X-Forwarded-Path": "/newroute/get",
        "X-Forwarded-Prefix": "/newroute"
    },
    "origin": "10.0.129.130, 3.67.184.22",
```

```
    "url":
"http://af1b40109a1ce44c48f93dfc6055145e-1222331437.eu-central-1.elb.amazo
naws.com/get"
}
```

```
kubectl delete ingress newroute
```



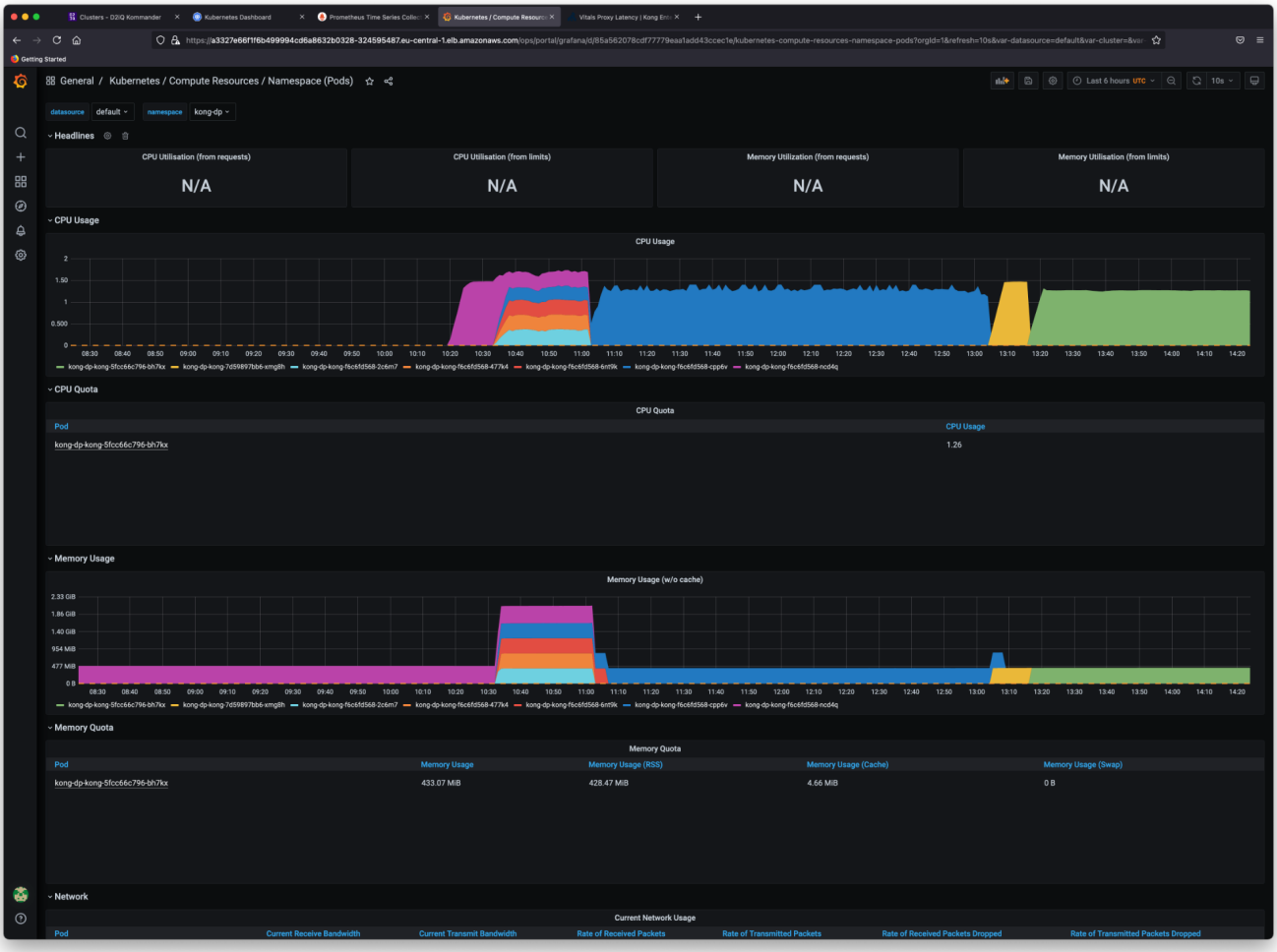

Kong Enterprise
Workspaces
Dev Portals
Vitals
Teams
Docs / Support
Choose from more timeframes and export data by leveraging the vitals InfluxDB strategy Learn more
Vitals
Timeframe: Last 30 Minutes
UTC
View: Hostname
Vitals is a feature within Kong's Admin API and Kong Manager that provides metrics about the health and performance of Kong nodes and Kong-proxied APIs. Learn more
Total Requests
400k
300k
200k
100k
0
9:50:00 am
9:55:00 am
10:00:00 am
10:05:00 am
10:10:00 am
10:15:00 am
HideAll Nodes
cluster
kong-dp-kong-f6c6fd568-cpp6v
kong-dp-kong-7d59897bb6-xmg8h
kong-dp-kong-5fcc66c796-bh7kx
Proxy Latency
Upstream Latency
Datastore Cache
Proxy Latency
Average
5 ms
4 ms
3 ms
2 ms
1 ms
0 ms
Hostname
ID
AVG
MIN
MAX
cluster - 0 ms 0 ms 109 ms
kong-dp-kong-5fcc66c796-bh7kx 26b2a261-5b62-4d57-94aa-3b3317e38a29 0 ms 0 ms 50 ms
kong-dp-kong-f6c6fd568-cpp6v 0e0f18e4-21e4-4196-9a20-028b438dd2e1 0 ms 0 ms 109 ms
kong-dp-kong-7d59897bb6-xmg8h d2c6dfc2-2a4b-4466-81fd-1f5597f5d2ae 0 ms 0 ms 90 ms

Lifecycle

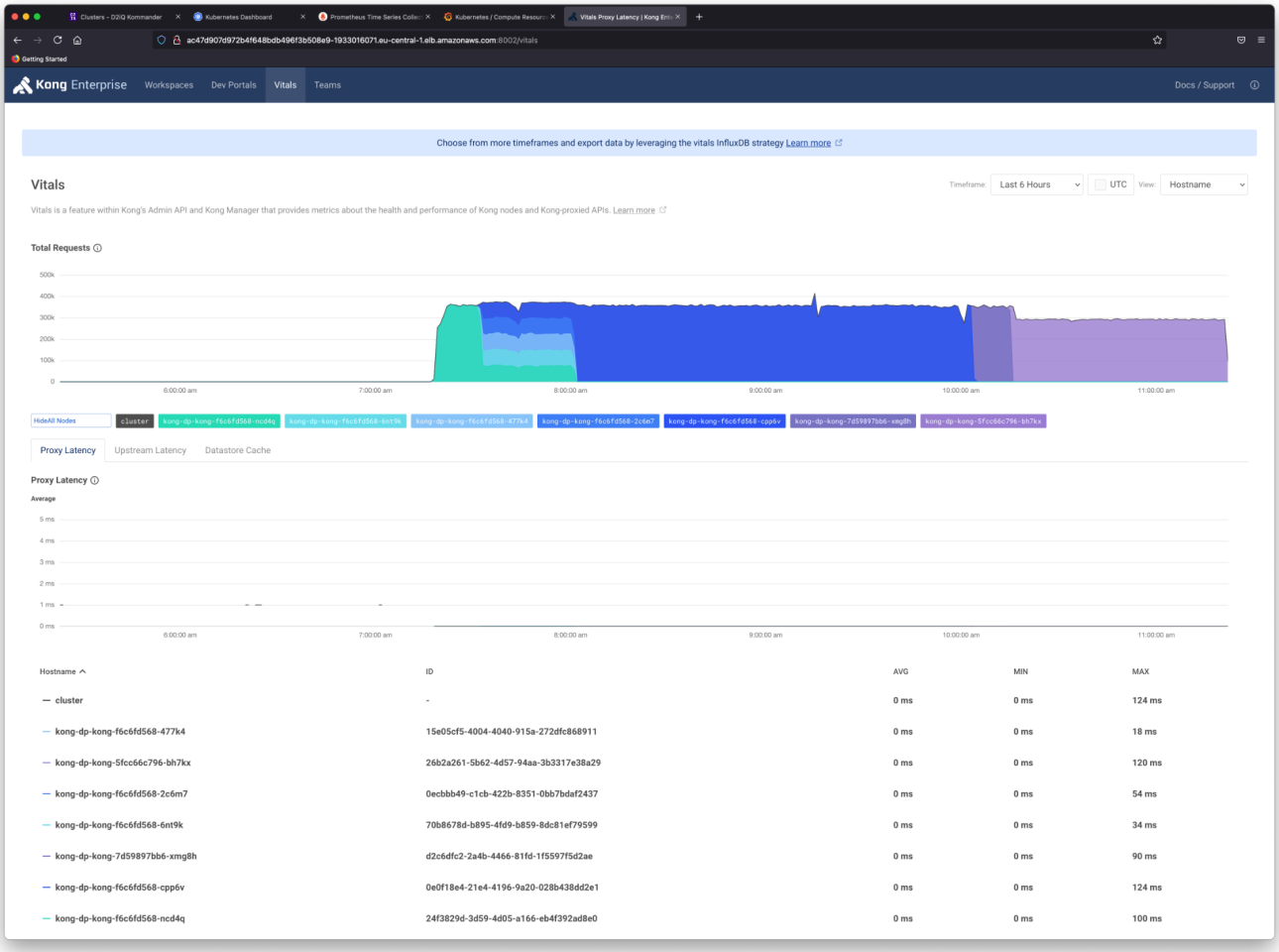
The charts present different workloads:

1. Single pod
2. After scaling the deployment to 5 pods
3. After reducing the deployment to 1 pod again
4. After Konnect upgrade to 2.3
5. After Konnect upgrade to 2.4

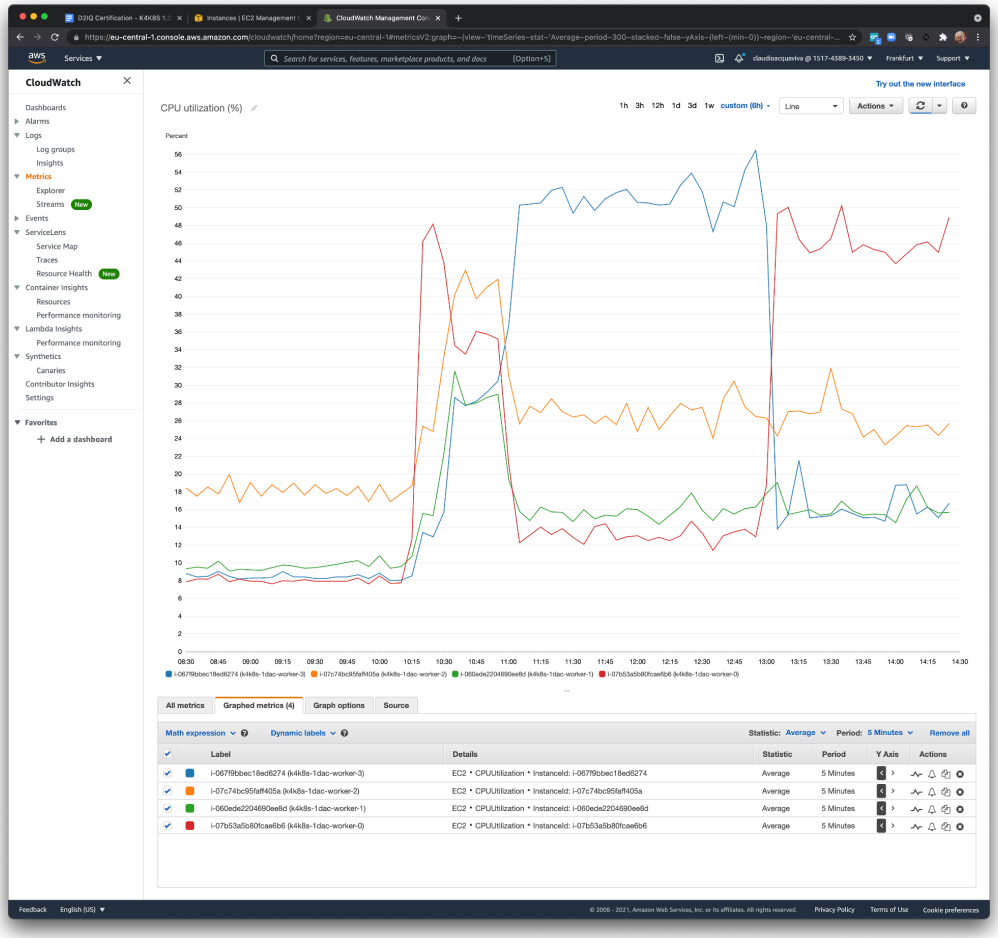
Konvoy Grafana



Kong Vitals


Kong Enterprise
Workspaces
Dev Portals
Vitals
Teams
Docs / Support
Choose from more timeframes and export data by leveraging the vitals InfluxDB strategy Learn more
Vitals
Timeframe: Last 6 Hours
UTC
View: Hostname
Vitals is a feature within Kong's Admin API and Kong Manager that provides metrics about the health and performance of Kong nodes and Kong-proxied APIs. Learn more
Total Requests
500k
400k
300k
200k
100k
0
6:00:00 am
7:00:00 am
8:00:00 am
9:00:00 am
10:00:00 am
11:00:00 am
Hide All Nodes
cluster
kong-dp-kong-f6c6fd568-ncd4q
kong-dp-kong-f6c6fd568-6nt9k
kong-dp-kong-f6c6fd568-477k4
kong-dp-kong-f6c6fd568-2c6m7
kong-dp-kong-f6c6fd568-cpp6v
kong-dp-kong-7d59897bb6-xmg8h
kong-dp-kong-5fcc66c796-bh7kx
Proxy Latency
Upstream Latency
Datastore Cache
Proxy Latency
Average
5 ms
4 ms
3 ms
2 ms
1 ms
0 ms
6:00:00 am
7:00:00 am
8:00:00 am
9:00:00 am
10:00:00 am
11:00:00 am
Hostname
ID
AVG
MIN
MAX
cluster - 0 ms 0 ms 124 ms
kong-dp-kong-f6c6fd568-477k4 15e05cf5-4004-4040-915a-272dfc868911 0 ms 0 ms 18 ms
kong-dp-kong-5fcc66c796-bh7kx 26b2a261-5b62-4d57-94aa-3b3317e38a29 0 ms 0 ms 120 ms
kong-dp-kong-f6c6fd568-2c6m7 0ecbbb49-c1cb-422b-8351-0bb7bdaf2437 0 ms 0 ms 54 ms
kong-dp-kong-f6c6fd568-6nt9k 70b8678d-b895-4fd9-b859-8dc81ef79599 0 ms 0 ms 34 ms
kong-dp-kong-7d59897bb6-xmg8h d2c6dfc2-2a4b-4466-81fd-1f5597f5d2ae 0 ms 0 ms 90 ms
kong-dp-kong-f6c6fd568-cpp6v 0e0f18e4-21e4-4196-9a20-028b438dd2e1 0 ms 0 ms 124 ms
kong-dp-kong-f6c6fd568-ncd4q 24f3829d-3d59-4d05-a166-eb4f392ad8e0 0 ms 0 ms 100 ms

AWS Cloud Watch



HPA

Checking the Kong Data Plane Metrics

```
kubectl port-forward service/prometheus-operated -n kubeaddons 9090
```

Get the API consumption rate

```
http
:9090/api/v1/query?query=sum\(rate\(kong_http_status{code=\"200\"}[1m]\)\)
| jq -r .data.result[0].value[1]
4820.704254396579
```

Get the number of successful processed requests

```
$ http :9090/api/v1/query?query=sum\(kong_http_status{code=\"200\"}\) | jq
-r .data.result[].value[1]
32992347
```

Turn HPA on

```
helm upgrade kong-dp kong/kong -n kong-dp \
--set ingressController.enabled=false \
--set image.repository=kong/kong-gateway \
--set image.tag=2.4.1.1-alpine \
--set env.database=off \
--set env.role=data_plane \
--set env.cluster_cert=/etc/secrets/kong-cluster-cert/tls.crt \
--set env.cluster_cert_key=/etc/secrets/kong-cluster-cert/tls.key \
--set
env.lua_ssl_trusted_certificate=/etc/secrets/kong-cluster-cert/tls.crt \
--set
env.cluster_control_plane=kong-kong-cluster.kong.svc.cluster.local:8005 \
--set
env.cluster_telemetry_endpoint=kong-kong-clustertelemetry.kong.svc.cluster
.local:8006 \
--set proxy.enabled=true \
--set proxy.type=LoadBalancer \
--set enterprise.enabled=true \
--set enterprise.license_secret=kong-enterprise-license \
--set enterprise.portal.enabled=false \
--set enterprise.rbac.enabled=false \
```

```
--set enterprise.smtp.enabled=false \
--set manager.enabled=false \
--set portal.enabled=false \
--set portalapi.enabled=false \
--set env.status_listen=0.0.0.0:8100 \
--set secretVolumes[0]=kong-cluster-cert \
--set resources.requests.cpu="300m" \
--set resources.requests.memory="300Mi" \
--set resources.limits.cpu="1200m" \
--set resources.limits.memory="800Mi" \
--set autoscaling.enabled=true \
--set autoscaling.minReplicas=1 \
--set autoscaling.maxReplicas=20 \
--set autoscaling.metrics[0].type=Resource \
--set autoscaling.metrics[0].resource.name=cpu \
--set autoscaling.metrics[0].resource.target.type=Utilization \
--set autoscaling.metrics[0].resource.target.averageUtilization=75
```

Checking HPA

```
$ kubectl get hpa -n kong-dp
NAME           REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
kong-dp-kong   Deployment/kong-dp-kong   76%/75%   1         20        7          14m

$ kubectl get pod -n kong-dp
NAME                            READY   STATUS    RESTARTS   AGE
kong-dp-kong-8b8889554-5wkb7   1/1     Running   0          13m
kong-dp-kong-8b8889554-bljhv   1/1     Running   0          10m
kong-dp-kong-8b8889554-hz24m   1/1     Running   0          13m
kong-dp-kong-8b8889554-kmzb7   1/1     Running   0          11m
kong-dp-kong-8b8889554-m6p8s   1/1     Running   0          13m
kong-dp-kong-8b8889554-sztbx   1/1     Running   0          11m
kong-dp-kong-8b8889554-xfr98   1/1     Running   0          14m
```

Check the Kong Data Plane Metrics again

Get the API consumption rate

```
$ http
:9090/api/v1/query?query=sum\(rate\(kong_http_status{code=\"200\"}[1m]\)\)
| jq -r .data.result[0].value[1]
```

```
5547.190371449822
```

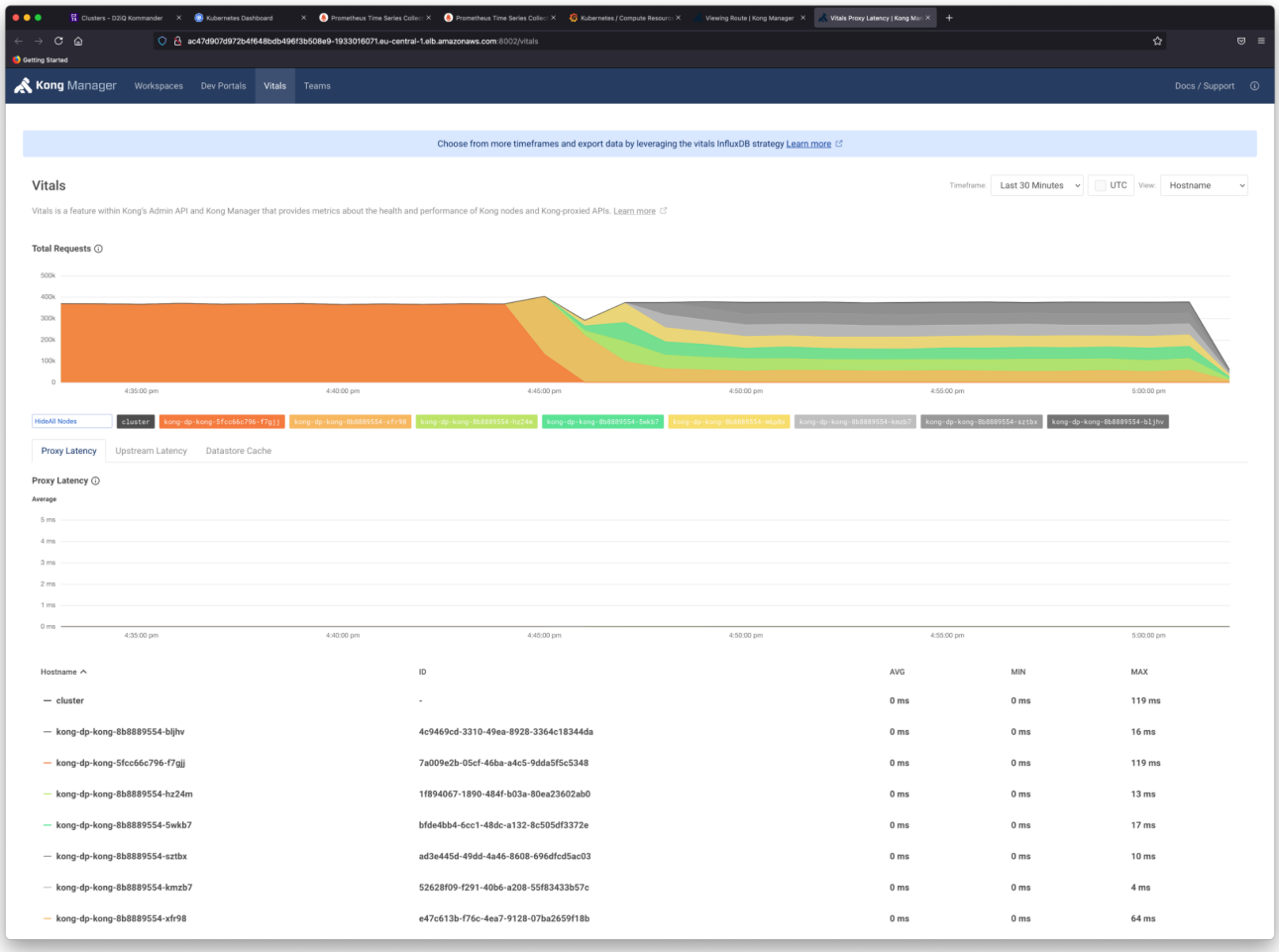
Checking HPA

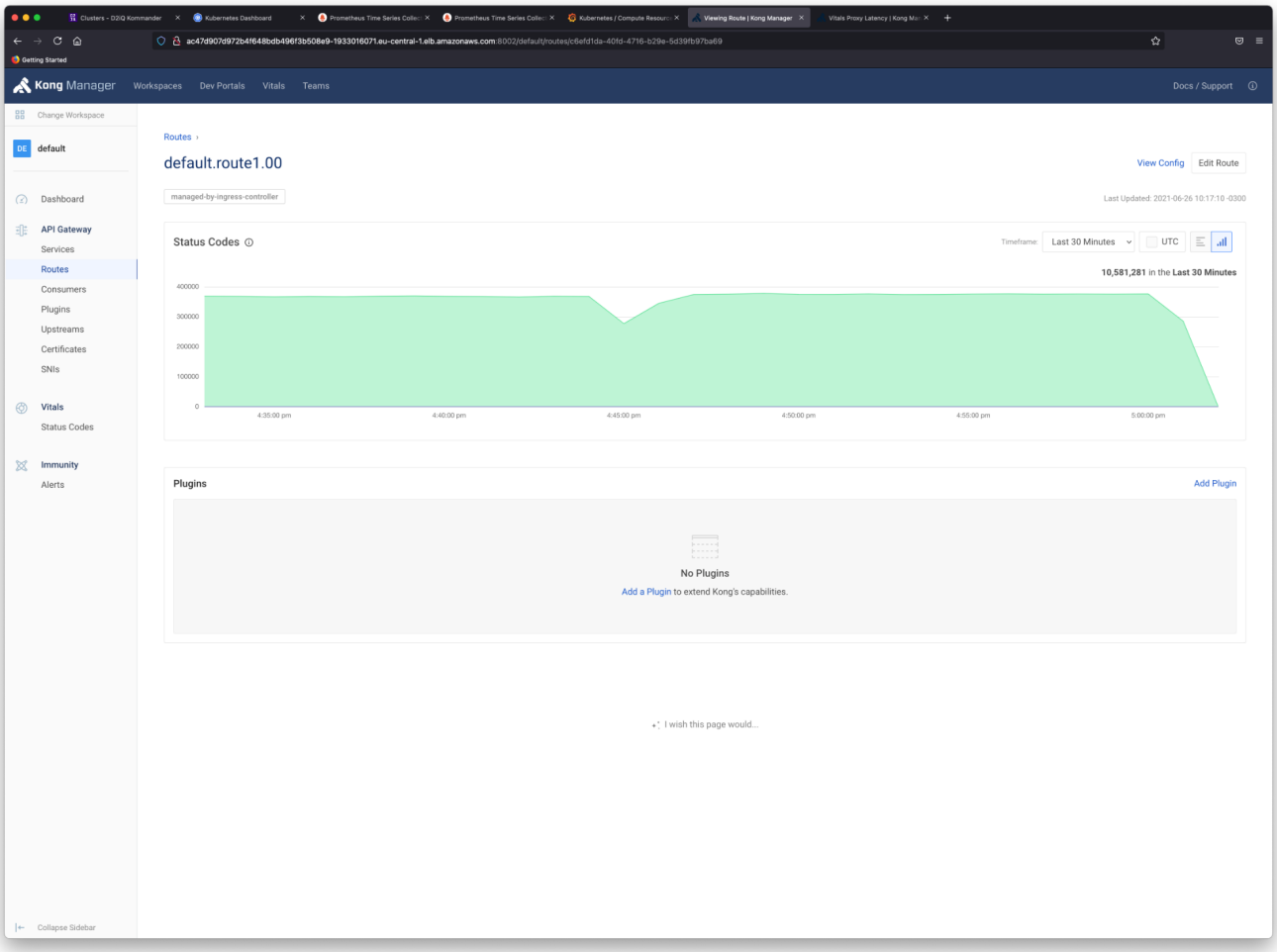
```
$ kubectl get hpa --all-namespaces
NAMESPACE   NAME           REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
kong-dp     kong-dp-kong   Deployment/kong-dp-kong   20%/50%   1         10        2          20m


$ kubectl describe hpa kong-dp-kong -n kong-dp
Name:                                                  kong-dp-kong
Namespace:                                             kong-dp
Labels:                                                app.kubernetes.io/instance=kong-dp
                                                       app.kubernetes.io/managed-by=Helm
                                                       app.kubernetes.io/name=kong
                                                       app.kubernetes.io/version=2.4
                                                       helm.sh/chart=kong-2.1.0
Annotations:                                           meta.helm.sh/release-name: kong-dp
                                                       meta.helm.sh/release-namespace: kong-dp
CreationTimestamp:                                     Sat, 26 Jun 2021 16:45:07 -0300
Reference:                                             Deployment/kong-dp-kong
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  68% (204m) / 75%
Min replicas:                                          1
Max replicas:                                          20
Deployment pods:                                       7 current / 7 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    ReadyForNewScale    recommended size matches current size
  ScalingActive   True    ValidMetricFound    the HPA was able to successfully calculate a replica
count from cpu resource utilization (percentage of request)
  ScalingLimited  False   DesiredWithinRange  the desired count is within the acceptable range
Events:
  Type     Reason                        Age                From                       Message
  ----     ------                        ----               ----                       -------
  Warning  FailedGetResourceMetric       15m (x2 over 16m)  horizontal-pod-autoscaler  missing
request for cpu
  Warning  FailedComputeMetricsReplicas  15m (x2 over 16m)  horizontal-pod-autoscaler  invalid
metrics (1 invalid out of 1), first error is: failed to get cpu utilization: missing request for cpu
  Warning  FailedGetResourceMetric       15m (x2 over 15m)  horizontal-pod-autoscaler  did not
receive metrics for any ready pods
  Warning  FailedComputeMetricsReplicas  15m (x2 over 15m)  horizontal-pod-autoscaler  invalid
metrics (1 invalid out of 1), first error is: failed to get cpu utilization: did not receive metrics
for any ready pods
  Normal   SuccessfulRescale             14m                horizontal-pod-autoscaler  New size: 4;
reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             13m                horizontal-pod-autoscaler  New size: 6;
reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             12m                horizontal-pod-autoscaler  New size: 7;
reason: cpu resource utilization (percentage of request) above target
```

If you want to change the policy run, for example:

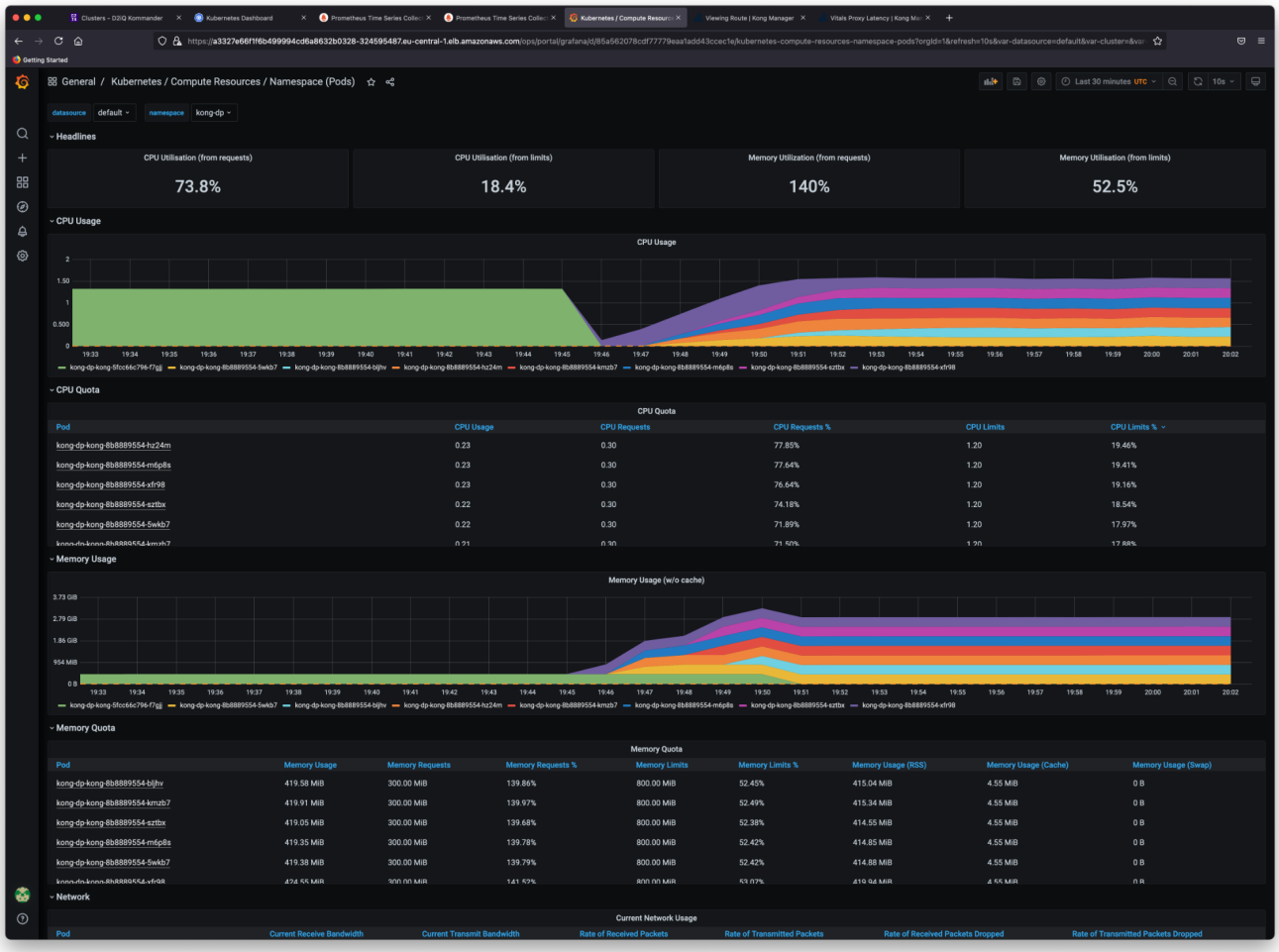
```
kubectl -n kong-dp patch hpa kong-dp-kong --patch
'{"spec":{"targetCPUUtilizationPercentage":75}}'

kubectl -n kong-dp patch hpa kong-dp-kong --patch
'{"spec":{"maxReplicas":20}}'
```

Konnect Vitals




Kong Manager
Workspaces
Dev Portals
Vitals
Teams
Docs / Support
Change Workspace
default
Dashboard
API Gateway
Services
Routes
Consumers
Plugins
Upstreams
Certificates
SNIs
Vitals
Status Codes
Immunity
Alerts
Collapse Sidebar
Routes
default.route1.00
View Config
Edit Route
managed-by-ingress-controller
Status Codes
Timeframe:
Last 30 Minutes
UTC
10,581,281 in the Last 30 Minutes
400000
300000
200000
100000
0
4:35:00 pm
4:40:00 pm
4:45:00 pm
4:50:00 pm
4:55:00 pm
5:00:00 pm
Plugins
Add Plugin
No Plugins
Add a Plugin to extend Kong's capabilities.
I wish this page would...

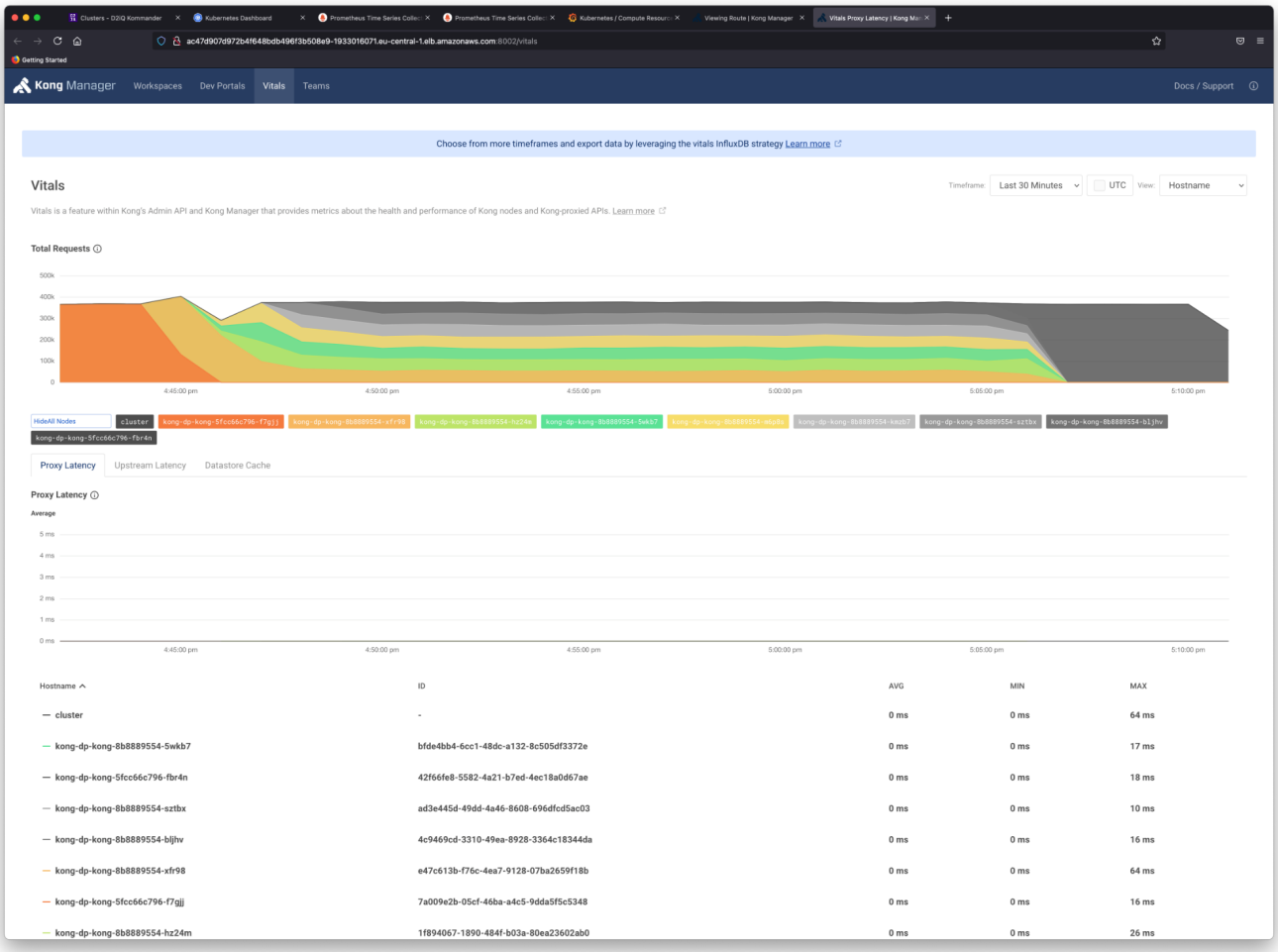
Konvoy Grafana

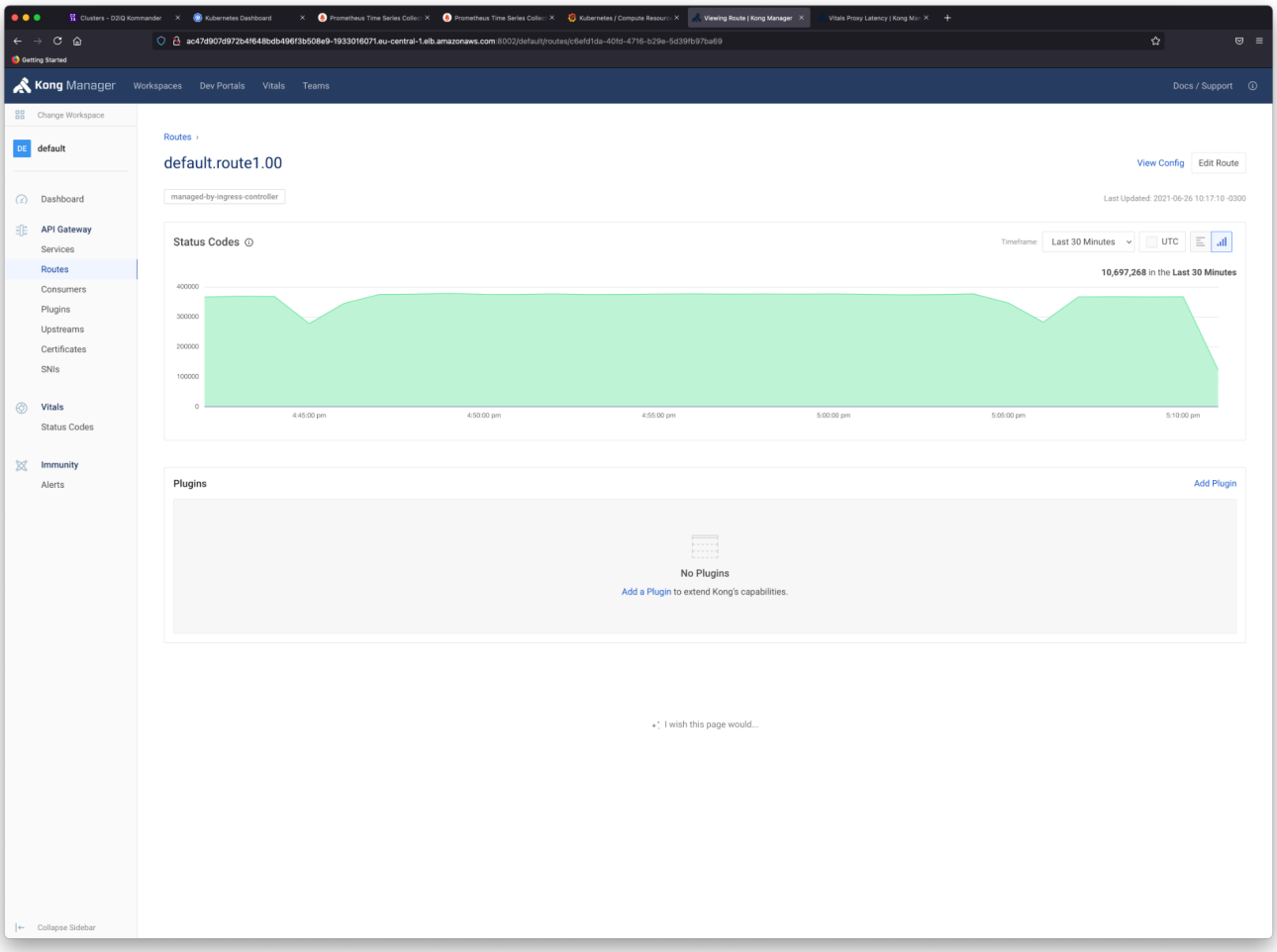


Turn HPA off

```
helm upgrade kong-dp kong/kong -n kong-dp \
--set ingressController.enabled=false \
--set image.repository=kong/kong-gateway \
--set image.tag=2.4.1.1-alpine \
--set env.database=off \
--set env.role=data_plane \
--set env.cluster_cert=/etc/secrets/kong-cluster-cert/tls.crt \
--set env.cluster_cert_key=/etc/secrets/kong-cluster-cert/tls.key \
--set
env.lua_ssl_trusted_certificate=/etc/secrets/kong-cluster-cert/tls.crt \
--set
env.cluster_control_plane=kong-kong-cluster.kong.svc.cluster.local:8005 \
```

```
--set
env.cluster_telemetry_endpoint=kong-kong-clustertelemetry.kong.svc.cluster
.local:8006 \
--set proxy.enabled=true \
--set proxy.type=LoadBalancer \
--set enterprise.enabled=true \
--set enterprise.license_secret=kong-enterprise-license \
--set enterprise.portal.enabled=false \
--set enterprise.rbac.enabled=false \
--set enterprise.smtp.enabled=false \
--set manager.enabled=false \
--set portal.enabled=false \
--set portalapi.enabled=false \
--set env.status_listen=0.0.0.0:8100 \
--set secretVolumes[0]=kong-cluster-cert


$ kubectl get pod -n kong-dp
NAME                             READY   STATUS    RESTARTS   AGE
kong-dp-kong-5fcc66c796-fbr4n   1/1     Running   0          43s

http
:9090/api/v1/query?query=sum\(rate\(kong_http_status{code=\"200\"}[1m]\)\)
| jq -r .data.result[0].value[1]
5246.609260447868
```


Kong Manager
Workspaces
Dev Portals
Vitals
Teams
Docs / Support
Choose from more timeframes and export data by leveraging the vitals InfluxDB strategy Learn more
Vitals
Vitals is a feature within Kong's Admin API and Kong Manager that provides metrics about the health and performance of Kong nodes and Kong-proxied APIs. Learn more
Timeframe: Last 30 Minutes
UTC
View: Hostname
Total Requests
500k
400k
300k
200k
100k
0
4:45:00 pm
4:50:00 pm
4:55:00 pm
5:00:00 pm
5:05:00 pm
5:10:00 pm
HideAll Nodes
cluster
kong-dp-kong-5fcc66c796-f7gjj
kong-dp-kong-8b8889554-xfr98
kong-dp-kong-8b8889554-hz24m
kong-dp-kong-8b8889554-5wkb7
kong-dp-kong-8b8889554-abp8s
kong-dp-kong-8b8889554-kmzb7
kong-dp-kong-8b8889554-sztbx
kong-dp-kong-8b8889554-bljhv
kong-dp-kong-5fcc66c796-fbr4n
Proxy Latency
Upstream Latency
Datastore Cache
Proxy Latency
Average
5 ms
4 ms
3 ms
2 ms
1 ms
0 ms
4:45:00 pm
4:50:00 pm
4:55:00 pm
5:00:00 pm
5:05:00 pm
5:10:00 pm
Hostname
ID
AVG
MIN
MAX
cluster - 0 ms 0 ms 64 ms
kong-dp-kong-8b8889554-5wkb7 bfde4bb4-6cc1-48dc-a132-8c505df3372e 0 ms 0 ms 17 ms
kong-dp-kong-5fcc66c796-fbr4n 42f66fe8-5582-4a21-b7ed-4ec18a0d67ae 0 ms 0 ms 18 ms
kong-dp-kong-8b8889554-sztbx ad3e445d-49dd-4a46-8608-696dfcd5ac03 0 ms 0 ms 10 ms
kong-dp-kong-8b8889554-bljhv 4c9469cd-3310-49ea-8928-3364c18344da 0 ms 0 ms 16 ms
kong-dp-kong-8b8889554-xfr98 e47c613b-f76c-4ea7-9128-07ba2659f18b 0 ms 0 ms 64 ms
kong-dp-kong-5fcc66c796-f7gjj 7a009e2b-05cf-46ba-a4c5-9dda5f5c5348 0 ms 0 ms 16 ms
kong-dp-kong-8b8889554-hz24m 1f894067-1890-484f-b03a-80ea23602ab0 0 ms 0 ms 26 ms


Kong Manager
Workspaces
Dev Portals
Vitals
Teams
Docs / Support
Change Workspace
default
Dashboard
API Gateway
Services
Routes
Consumers
Plugins
Upstreams
Certificates
SNIs
Vitals
Status Codes
Immunity
Alerts
Collapse Sidebar
Routes
default.route1.00
View Config
Edit Route
managed-by-ingress-controller
Status Codes
Timeframe:
Last 30 Minutes
UTC
10,697,268 in the Last 30 Minutes
400000
300000
200000
100000
0
4:45:00 pm
4:50:00 pm
4:55:00 pm
5:00:00 pm
5:05:00 pm
5:10:00 pm
Plugins
Add Plugin
No Plugins
Add a Plugin to extend Kong's capabilities.
I wish this page would...


General / Kubernetes / Compute Resources / Namespace (Pods)
Headlines
CPU Utilisation (from requests)
N/A
CPU Utilisation (from limits)
N/A
Memory Utilization (from requests)
N/A
Memory Utilisation (from limits)
N/A
CPU Usage
CPU Quota
Pod
CPU Usage
kong-dp-kong-5fcc66c796-fbr4n
1.28
Memory Usage
Memory Usage (w/o cache)
Memory Quota
Pod
Memory Usage
Memory Usage (RSS)
Memory Usage (Cache)
Memory Usage (Swap)
kong-dp-kong-8b8889554-5wkb7
0 B
0 B
0 B
0 B
kong-dp-kong-8b8889554-hz24m
0 B
0 B
0 B
0 B
kong-dp-kong-8b8889554-bljhv
246.19 MiB
241.54 MiB
4.57 MiB
0 B
kong-dp-kong-5fcc66c796-fbr4n
430.78 MiB
426.28 MiB
4.56 MiB
0 B
kong-dp-kong-8b8889554-sztbx
0 B
0 B
0 B
0 B
Network
Current Network Usage
Pod
Current Receive Bandwidth
Current Transmit Bandwidth
Rate of Received Packets
Rate of Transmitted Packets
Rate of Received Packets Dropped
Rate of Transmitted Packets Dropped

K4K8S Ingress Controller Policies

Rate Limiting Policy Definition

Since we have the Microservice exposed through a route defined in the Ingress Controller, let's protect it with a Rate Limiting Policy first.

Create the plugin

```
cat <<EOF | kubectl apply -f -
apiVersion: configuration.konghq.com/v1
kind: KongPlugin
metadata:
  name: rl-by-minute
  namespace: kong
config:
  minute: 3
  policy: local
plugin: rate-limiting
EOF
```

If you want to delete it run:

```
$ kubectl delete kongplugin rl-by-minute -n kong
```

Apply the plugin to the route

```
kubectl patch ingress route1 -p
'{"metadata":{"annotations":{"konghq.com/plugins":"rl-by-minute"}}}' -n kong
```

Deleting the annotation

In case you want to disapply the plugin to the ingress run:

```
$ kubectl annotate ingress route1 konghq.com/plugins- -n kong
```

Test the plugin

```
$ http
a8f867f57399a42aa8552685ceded6bb-310569142.eu-north-1.elb.amazonaws.com/ro
ute1/hw2
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 48
Content-Type: text/html; charset=utf-8
Date: Fri, 09 Apr 2021 22:30:59 GMT
RateLimit-Limit: 3
RateLimit-Remaining: 2
RateLimit-Reset: 1
Server: Werkzeug/1.0.1 Python/3.8.3
Via: kong/2.3.3
X-Kong-Proxy-Latency: 1
X-Kong-Upstream-Latency: 120
X-RateLimit-Limit-Minute: 3
X-RateLimit-Remaining-Minute: 2

Hello World, Benigno: 2021-04-09 22:30:59.945637
```

As expected, we get an error for the 4th request::

```
$ http
a8f867f57399a42aa8552685ceded6bb-310569142.eu-north-1.elb.amazonaws.com/ro
ute1/hw2
HTTP/1.1 429 Too Many Requests
Connection: keep-alive
Content-Length: 41
Content-Type: application/json; charset=utf-8
Date: Fri, 09 Apr 2021 22:31:06 GMT
RateLimit-Limit: 3
RateLimit-Remaining: 0
RateLimit-Reset: 54
Retry-After: 54
Server: kong/2.3.3
X-Kong-Response-Latency: 1
X-RateLimit-Limit-Minute: 3
X-RateLimit-Remaining-Minute: 0
```

```
{
    "message": "API rate limit exceeded"
}
```

API Key Policy Definition

Now, let's add an API Key Policy to this route:

Create the plugin

```
cat <<EOF | kubectl apply -f -
apiVersion: configuration.konghq.com/v1
kind: KongPlugin
metadata:
  name: apikey
  namespace: kong
plugin: key-auth
EOF
```

If you want to delete it run:

```
$ kubectl delete kongplugin apikey -n kong
```

Apply the plugin to the route

Now, let's add an API Key Policy to this route keeping the original Rate Limiting plugin:

```
kubectl patch ingress route1 -p
'{"metadata":{"annotations":{"konghq.com/plugins":"apikey, rl-by-minute"}}}' -n kong
```

Test the plugin

As expected, if we try to consume the route we get an error:

```
$ http
a8f867f57399a42aa8552685ceded6bb-310569142.eu-north-1.elb.amazonaws.com/route1/hw2
HTTP/1.1 401 Unauthorized
Connection: keep-alive
Content-Length: 45
Content-Type: application/json; charset=utf-8
```

```
Date: Fri, 09 Apr 2021 22:32:46 GMT
Server: kong/2.3.3
WWW-Authenticate: Key realm="kong"
X-Kong-Response-Latency: 1

{
    "message": "No API key found in request"
}
```

Provisioning a Key

```
$ kubectl create secret generic consumerapikey --from-literal=kongCredType=key-auth
--from-literal=key=kuma-secret -n kong
secret/consumerapikey created
```

If you want to delete it run:

```
$ kubectl delete secret consumerapikey -n kong
```

Creating a Consumer with the Key

```
cat <<EOF | kubectl apply -f -
apiVersion: configuration.konghq.com/v1
kind: KongConsumer
metadata:
  name: consumer1
  namespace: kong
  annotations:
    kubernetes.io/ingress.class: kong
username: consumer1
credentials:
- consumerapikey
EOF
```

If you want to delete it run:

```
$ kubectl delete kongconsumer consumer1 -n kong
```

Consume the route with the API Key

```
$ http
a8f867f57399a42aa8552685ceded6bb-310569142.eu-north-1.elb.amazonaws.com/route1/hw2
apikey:kuma-secret
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 48
Content-Type: text/html; charset=utf-8
Date: Fri, 09 Apr 2021 22:33:43 GMT
RateLimit-Limit: 3
RateLimit-Remaining: 2
RateLimit-Reset: 17
Server: Werkzeug/1.0.1 Python/3.8.3
Via: kong/2.3.3
X-Kong-Proxy-Latency: 1
X-Kong-Upstream-Latency: 124
X-RateLimit-Limit-Minute: 3
X-RateLimit-Remaining-Minute: 2

Hello World, Benigno: 2021-04-09 22:33:43.591777
```

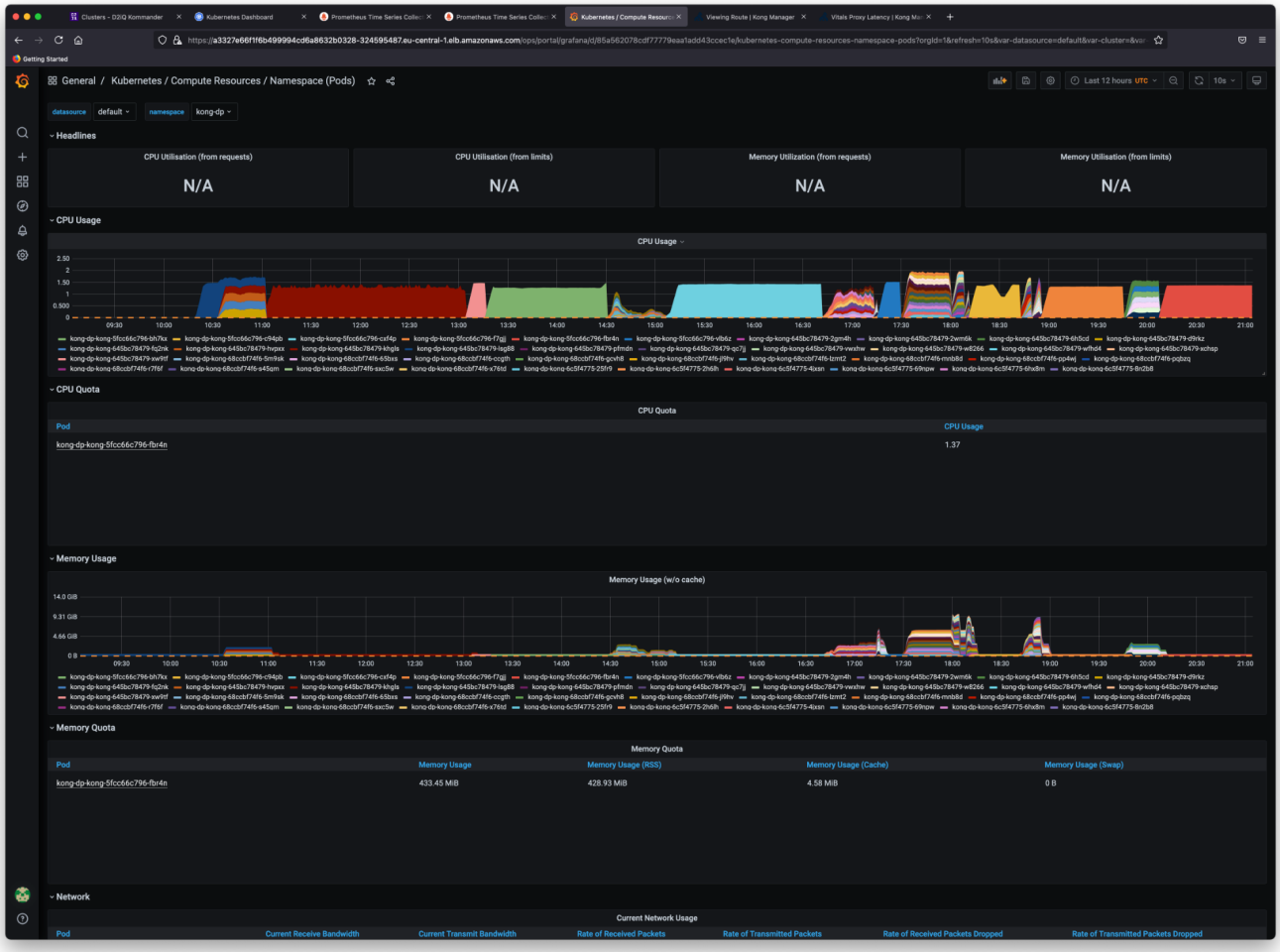
Getting the Rate Limiting error

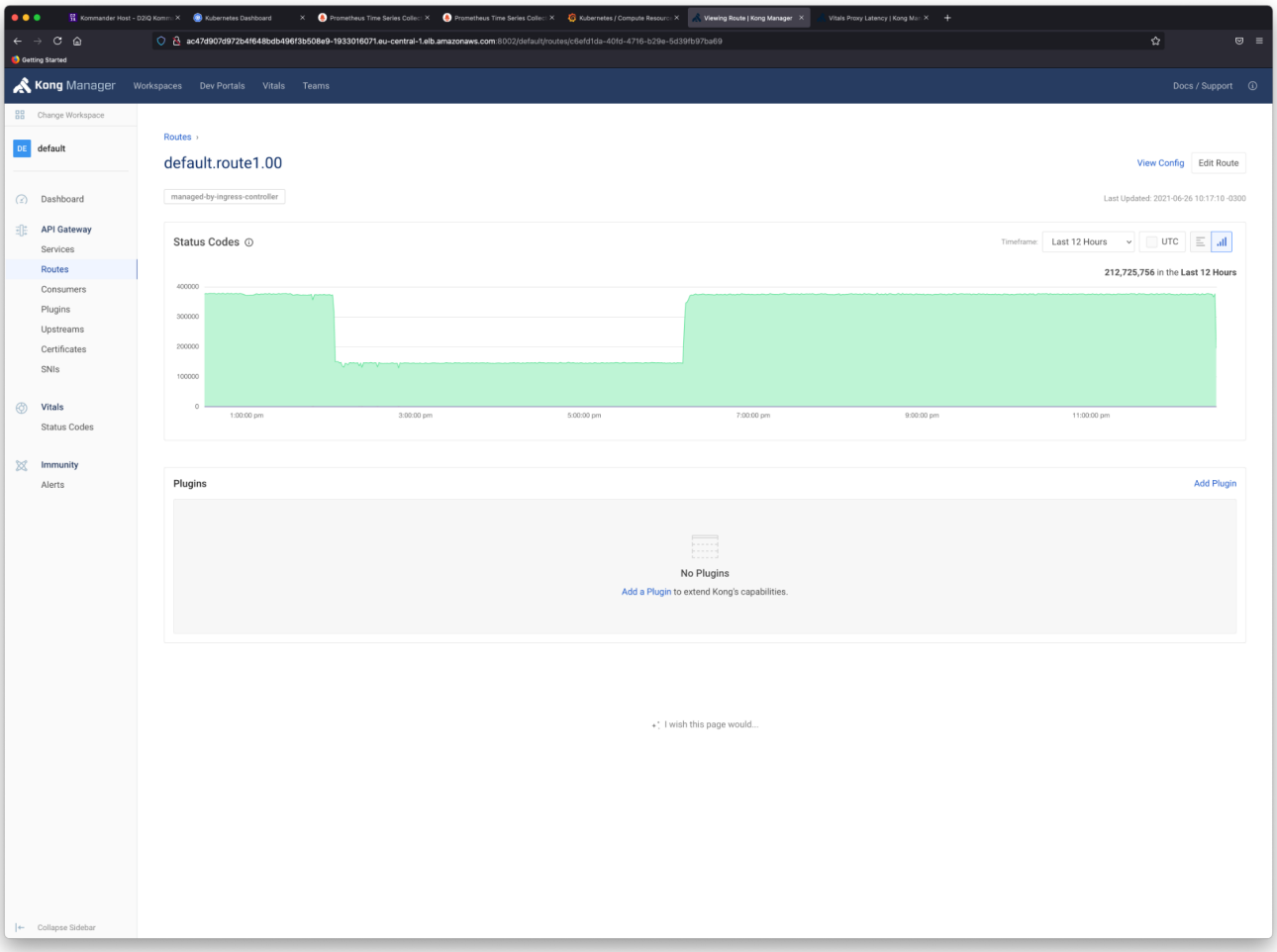
Again, if we try the 4th request in a single minute we get the rate limiting error

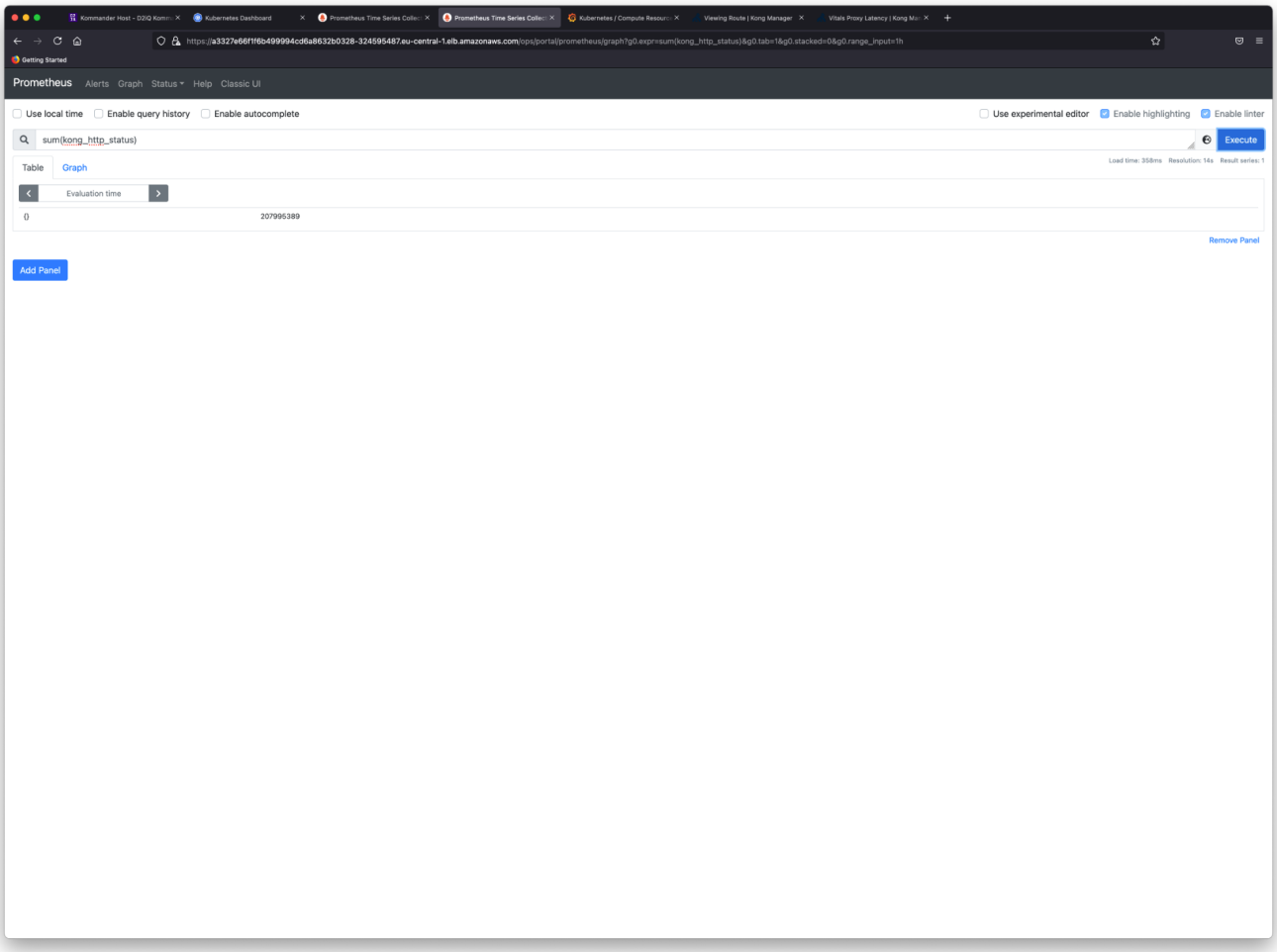
```
$ http
a8f867f57399a42aa8552685ceded6bb-310569142.eu-north-1.elb.amazonaws.com/route1/hw2
apikey:kuma-secret
HTTP/1.1 429 Too Many Requests
Connection: keep-alive
Content-Length: 41
Content-Type: application/json; charset=utf-8
Date: Fri, 09 Apr 2021 22:34:07 GMT
RateLimit-Limit: 3
RateLimit-Remaining: 0
RateLimit-Reset: 53
Retry-After: 53
Server: kong/2.3.3
X-Kong-Response-Latency: 1
X-RateLimit-Limit-Minute: 3
X-RateLimit-Remaining-Minute: 0

{
```

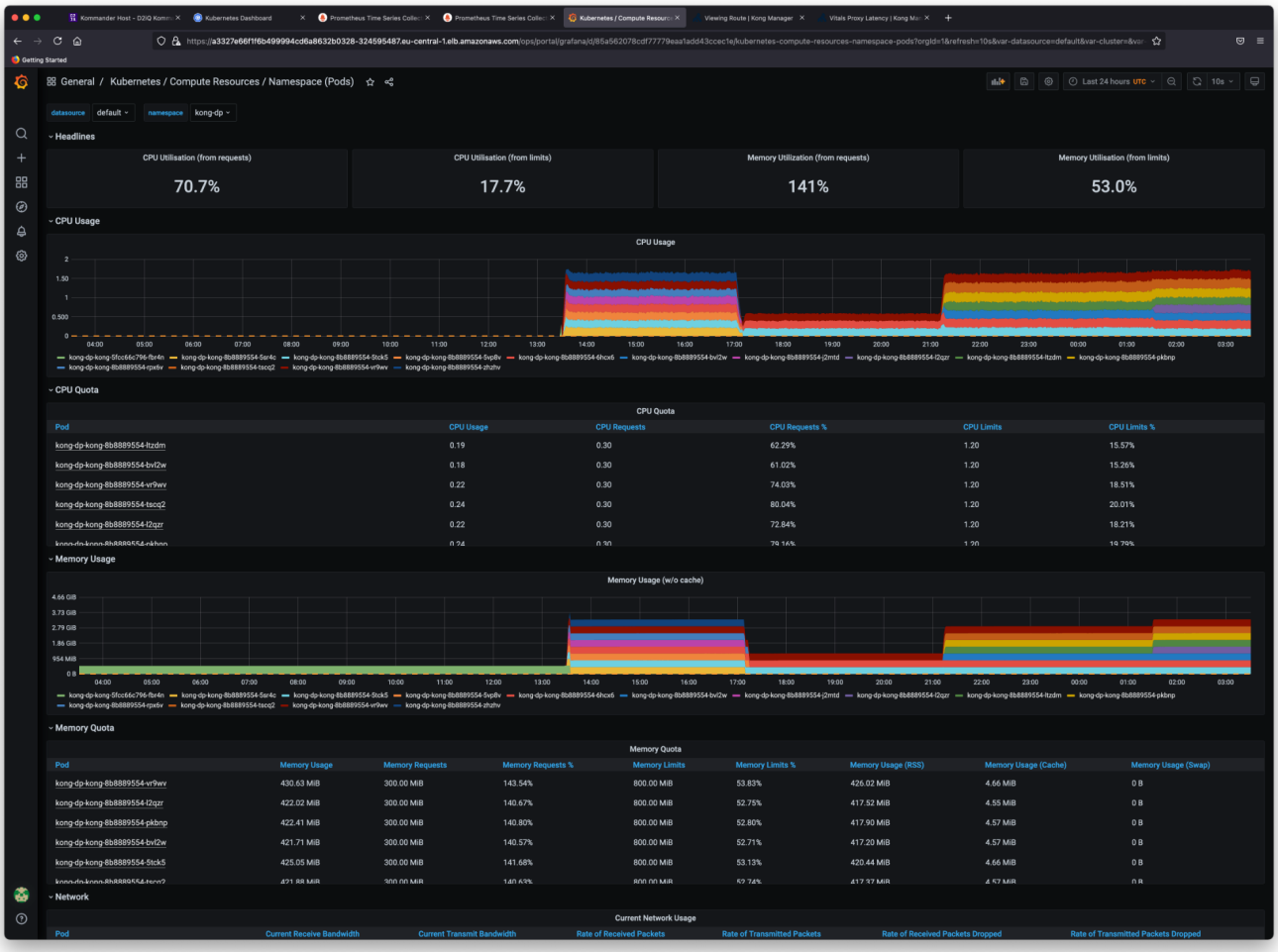
```
    "message": "API rate limit exceeded"
}
```

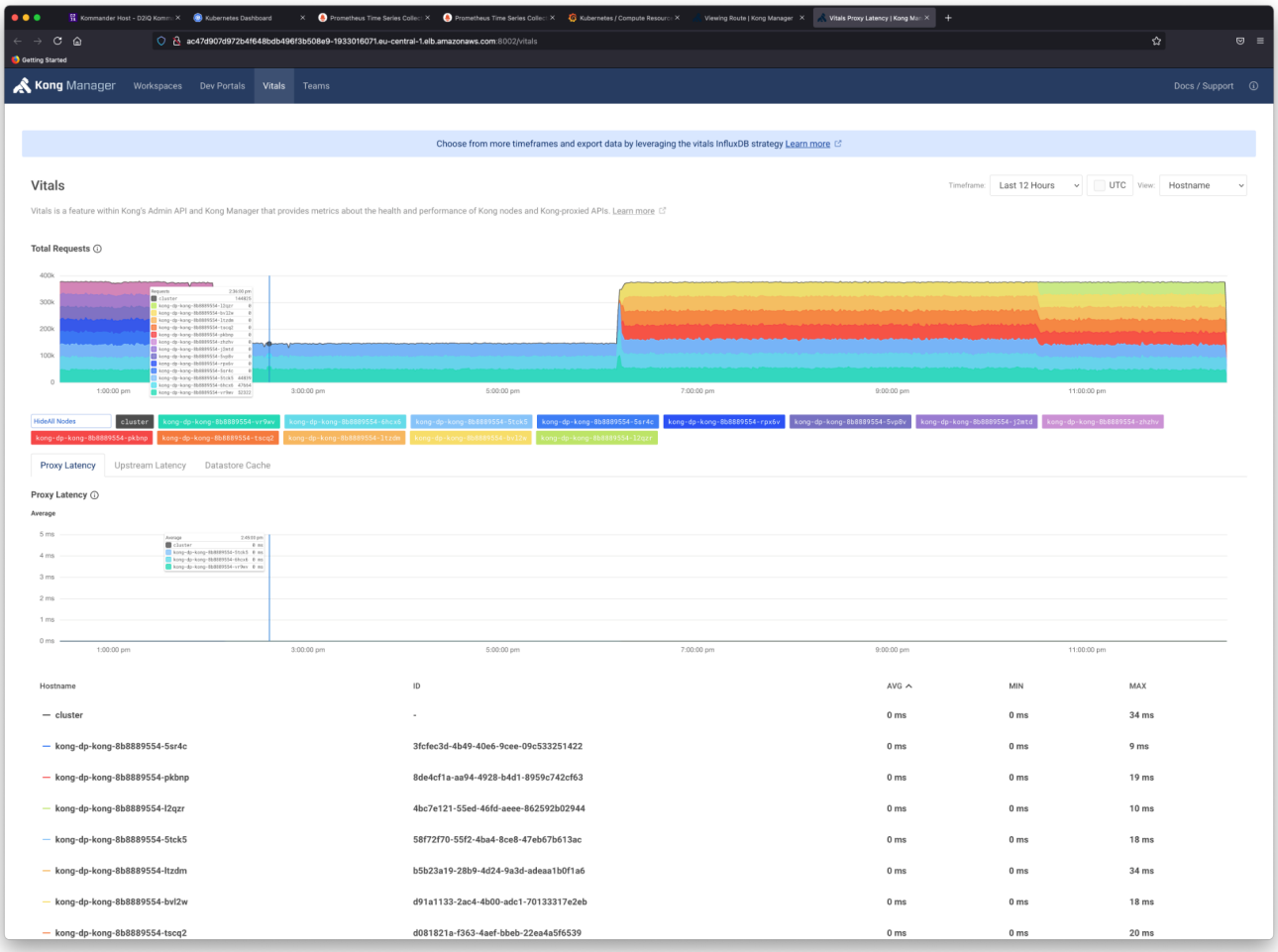
Screenshots




Kong Manager
Workspaces
Dev Portals
Vitals
Teams
Docs / Support
Change Workspace
default
Dashboard
API Gateway
Services
Routes
Consumers
Plugins
Upstreams
Certificates
SNIs
Vitals
Status Codes
Immunity
Alerts
Collapse Sidebar
Routes
default.route1.00
View Config
Edit Route
managed-by-ingress-controller
Last Updated: 2021-06-26 10:17:10 -0300
Status Codes
Timeframe:
Last 12 Hours
UTC
212,725,756 in the Last 12 Hours
400000
300000
200000
100000
0
1:00:00 pm
3:00:00 pm
5:00:00 pm
7:00:00 pm
9:00:00 pm
11:00:00 pm
Plugins
Add Plugin
No Plugins
Add a Plugin to extend Kong's capabilities.
I wish this page would...


Prometheus
Alerts
Graph
Status
Help
Classic UI
Use local time
Enable query history
Enable autocomplete
Use experimental editor
Enable highlighting
Enable linter
sum(kong_http_status)
Execute
Table
Graph
Evaluation time
{}
207995389
Remove Panel
Add Panel


General / Kubernetes / Compute Resources / Namespace (Pods)
Headlines
CPU Utilisation (from requests)
70.7%
CPU Utilisation (from limits)
17.7%
Memory Utilization (from requests)
141%
Memory Utilisation (from limits)
53.0%
CPU Usage
CPU Quota
Memory Usage
Memory Quota
Network
Current Network Usage


Kong Manager
Workspaces
Dev Portals
Vitals
Teams
Docs / Support
Choose from more timeframes and export data by leveraging the vitals InfluxDB strategy Learn more
Vitals
Vitals is a feature within Kong's Admin API and Kong Manager that provides metrics about the health and performance of Kong nodes and Kong-proxied APIs. Learn more
Timeframe: Last 12 Hours
UTC
View: Hostname
Total Requests
Proxy Latency
Upstream Latency
Datastore Cache
Proxy Latency
Average
Hostname
ID
AVG
MIN
MAX
cluster - 0 ms 0 ms 34 ms
kong-dp-kong-8b8889554-5sr4c 3fcfec3d-4b49-40e6-9cee-09c533251422 0 ms 0 ms 9 ms
kong-dp-kong-8b8889554-pkbnp 8de4cf1a-aa94-4928-b4d1-8959c742cf63 0 ms 0 ms 19 ms
kong-dp-kong-8b8889554-l2qzr 4bc7e121-55ed-46fd-aeee-862592b02944 0 ms 0 ms 10 ms
kong-dp-kong-8b8889554-5tck5 58f72f70-55f2-4ba4-8ce8-47eb67b613ac 0 ms 0 ms 18 ms
kong-dp-kong-8b8889554-ltzdm b5b23a19-28b9-4d24-9a3d-adeaa1b0f1a6 0 ms 0 ms 34 ms
kong-dp-kong-8b8889554-bvl2w d91a1133-2ac4-4b00-adc1-70133317e2eb 0 ms 0 ms 18 ms
kong-dp-kong-8b8889554-tscq2 d081821a-f363-4aef-bbeb-22ea4a5f6539 0 ms 0 ms 20 ms