Name

[Company name]  [Company address]

Service mesh poc

Contents

[Introduction 2](#_Toc77346173)

[Purpose 2](#_Toc77346174)

[Scope 2](#_Toc77346175)

[This document is intended for the team members who requires to evaluate the PoC. 2](#_Toc77346176)

[Installation Manual 2](#_Toc77346177)

[Pre-requisites 2](#_Toc77346178)

[Configuration of Virtual Machine 2](#_Toc77346179)

[Software Install procedure 4](#_Toc77346180)

[Software accounts 5](#_Toc77346181)

[PoC Install and testing 5](#_Toc77346182)

[Monitoring &Visualization 22](#_Toc77346183)

[1: ***Kiali*** (Visualization Tool) 🡪 Real Time 22](#_Toc77346184)

[2: ***Grafana*** (Visualization Tool) 🡪 Real Time: 25](#_Toc77346185)

[3: ***Prometheus*** (Monitoring): 🡪 (TODO:Explore Options, Alerts, Monitoring) 29](#_Toc77346186)

[4: ***Jaeger*** (Service Tracing)(TODO:Explore Options) 30](#_Toc77346187)

# Introduction

## Purpose

This document provides the instructions to be followed for the installation for a software used for the PoC as well as the steps on the verification of the components developed for the PoC.

## Scope

# This document is intended for the team members who requires to evaluate the PoC.

# Installation Manual

In order to facilitate the team tasks, it is vital to provide the delivery with accurate, exhaustive, complete and clear installation instructions.

## Pre-requisites

Following are the list of install prerequisites that must be fulfilled before the install can begin.

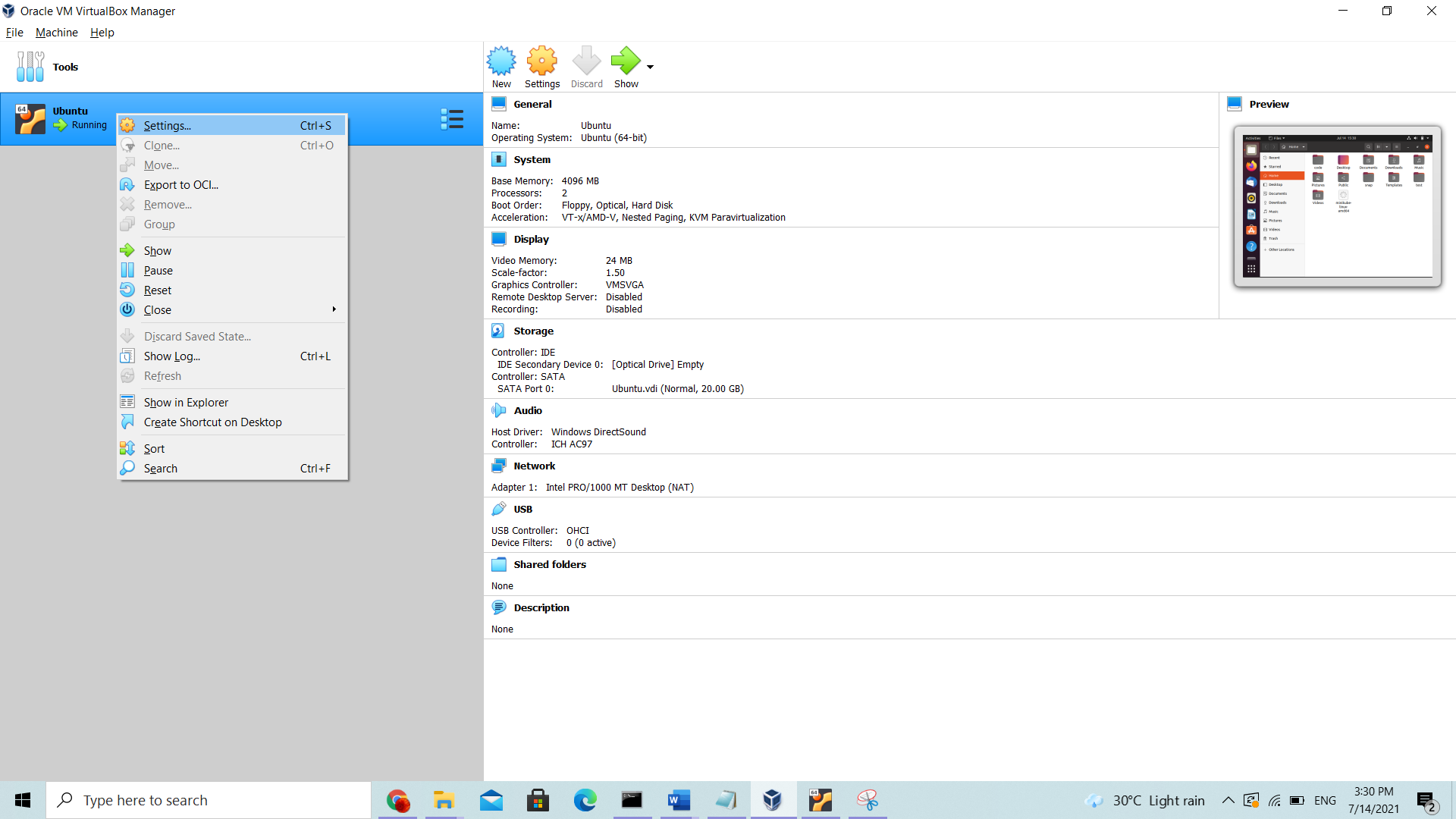
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S No** | **Type** | **Name with Version** | **Purpose** | **Path to download** |
| 1 | Operating System | Windows 7/10 | Infrastructure for Virtual environment |  |
| 2 | Virtual Box | Oracle Virtual Box 6.1 | Infrastructure to virtualize, where Open Source software like Ubuntu can be installed | Download [Link](https://dl5.filehippo.com/48b/14e/9623941a95aa2dde0fabab26cb1885a88d/VirtualBox-6.1.18-142142-Win.exe?Expires=1621390018&Signature=5012eb7b87043fa053e10ae91ae1916cc5df4a09&url=https://filehippo.com/download_virtualbox/&Filename=VirtualBox-6.1.18-142142-Win.exe)  Alternate Path  <https://www.virtualbox.org/wiki/Downloads> |
| 3 | Operating System | Ubuntu 20.04.2 | Infrastructure where the PoC will run. | Download [Link](https://mirrors.piconets.webwerks.in/ubuntu-mirror/ubuntu-releases/20.04.2.0/ubuntu-20.04.2.0-desktop-amd64.iso)  Alternate Path  <https://ubuntu.com/download/desktop> |

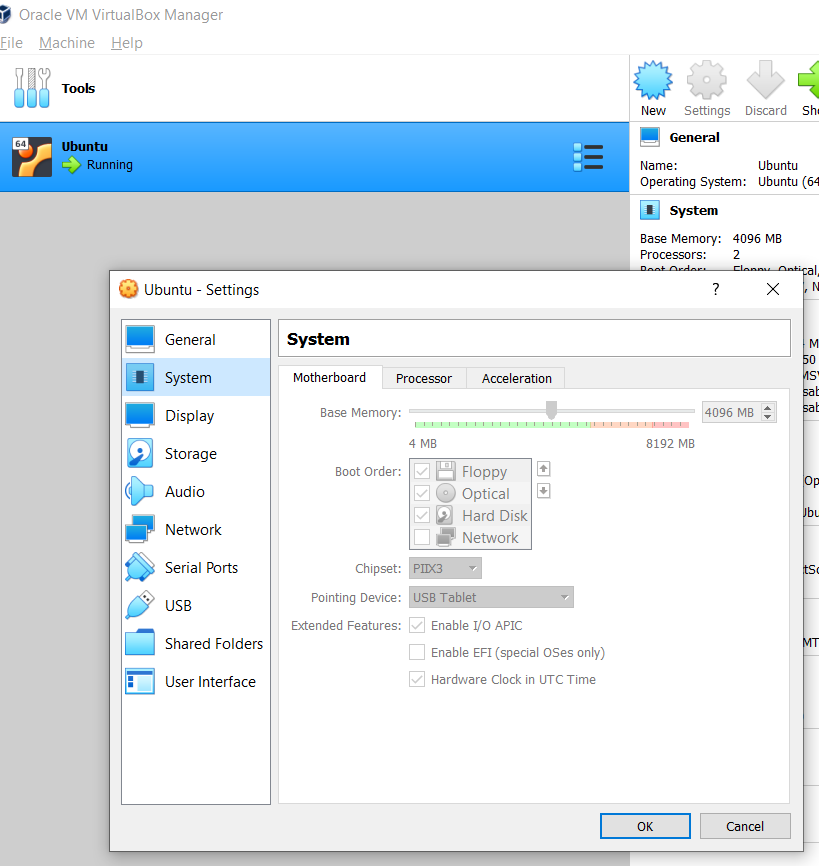
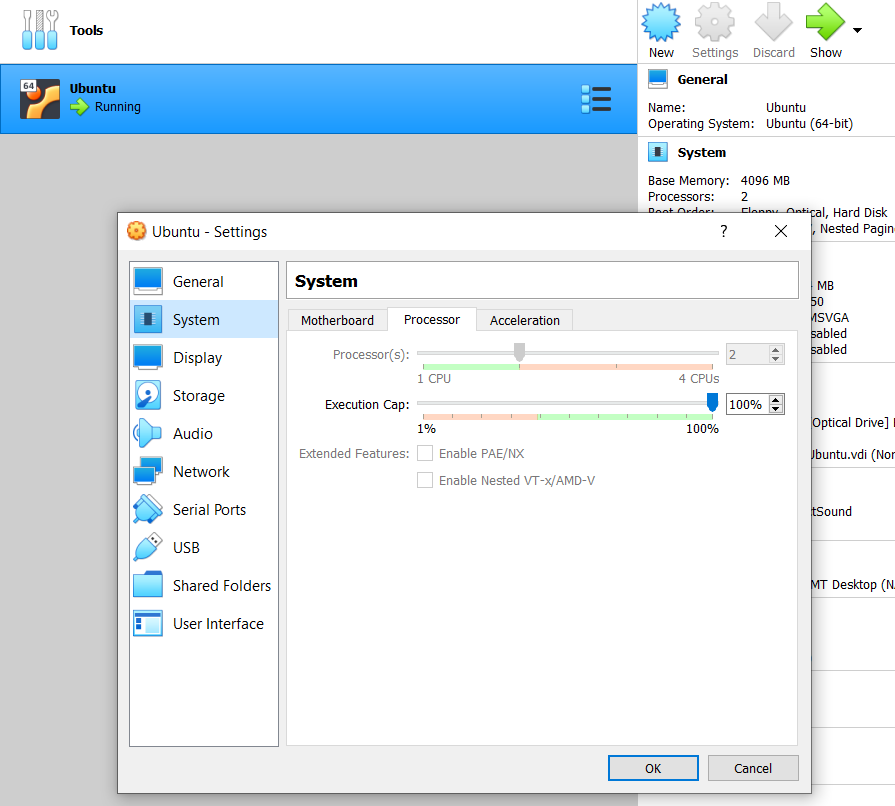
Follow the instructions from [here](https://itsfoss.com/install-linux-in-virtualbox/) to install pre-requisites without which we may not able to run the PoC.

**Note** : It is advisable to have the good configuration of Windows Machine so that the Operating System can be configured to run in Virtual Box without latency.

### Configuration of Virtual Machine

Optimal configuration of the Virtual Machine is 4096MB Base Memory, 2 Processor(s) shown below



Important : It is assumed that the Windows is Operating System used as Infrastructure for the evaluation of PoC. If you use the Mac or any Linux machines, the Pre-requisite can be skipped.

## Software Install procedure

Following software are required in the Ubuntu environment which needs to be installed below table order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S No** | **Name with Version** | **Usage** | **Path to download** | **Validate installation command line** |
| 1 | Git 2.25.1 | To checkout code from Git. By default Ubuntu provides it and no installation is required |  | git --version |
| 2 | Visual Studio Code | IDE for code | Download [link](https://az764295.vo.msecnd.net/stable/2aeda6b18e13c4f4f9edf6667158a6b8d408874b/code_1.58.1-1626158276_amd64.deb)  Alternate path  <https://code.visualstudio.com/download> |  |
| 3 | Docker | Used  for building and containerizing applications | Instruction available at [link](https://docs.docker.com/engine/install/ubuntu/#install-using-the-convenience-script)  Alternate path  <https://docs.docker.com/engine/install/ubuntu/> | docker --version |
| 4 | Docker-compose | Tool for defining and running multi-container Docker applications | Instruction available at [link](https://docs.docker.com/compose/install/#install-compose-on-linux-systems)  Alternate path  <https://docs.docker.com/compose/install/#install-compose> | docker-compose --version |
| 5 | Minikube and Kubernates | Minikube is open source tool that allows you to set up a single-node Kubernetes cluster. | Instruction available at [link](https://phoenixnap.com/kb/install-minikube-on-ubuntu) | minikube version  kubectl version -o json |
| 6 | Istio | Configure Service mesh with various parameters like traffic management, security, and policy | Instruction available at [link](https://istio.io/latest/docs/setup/getting-started/#download)  Alternate path 1  Download files [1-istio-init.yaml](https://github.com/ksrangarajan/service_mesh_poc/blob/main/1-istio-init.yaml) [2-istio-minikube.yaml](https://github.com/ksrangarajan/service_mesh_poc/blob/main/2-istio-minikube.yaml) and [4-label-default-namespace.yaml](https://github.com/ksrangarajan/service_mesh_poc/blob/main/4-label-default-namespace.yaml)  Run below commands in shell  kubectl apply -f  1-istio-init.yaml  kubectl apply -f  2-istio-minikube.yaml  kubectl apply -f  4-label-default-namespace.yaml  # Restart all pods to get sidecar injected  kubectl delete pods --all  Add the istioctl client to your path (Linux or macOS):  export PATH=$PWD/bin:$PATH  Alternate path 2  <https://istio.io/latest/docs/setup/getting-started/> | istioctl x precheck |

Important : Command line instruction executed in terminal are marked as

export PATH=$PWD/bin:$PATH

t is assumed that the Windows is Operating System used as Infrastructure for the evaluation of PoC. If you use the Mac or any Linux machines, the Pre-requisite can be skipped.

## Software accounts

We need to have account in the [docker hub](https://hub.docker.com/) for pulling or pushing service images. Hence create account for yourself [here](https://hub.docker.com/). All the images used in this document is available [here](https://hub.docker.com/r/ksrangar/services/tags?page=1&ordering=last_updated). The docker hub of your account will be referenced as <dockerhubuserid> in remaining part of the document.

**Note** : <dockerhubuserid> is the id in the docker hub created by yourself.

Example ksrangar

## PoC Install and testing

Following are the steps to be followed for the service mesh PoC execution

1. Create folder **code** in your desktop of the Ubuntu virtual machine
2. Open **Terminal** and navigate to the folder **code**
3. Use the Git command to clone the source code from [here](https://github.com/ksrangarajan/service_mesh_poc)

git clone <https://github.com/ksrangarajan/service_mesh_poc>

Although we cloned the code, we use the existing images for the evaluation and use configuration files (YAML) for the deployment and testing.

1. Pull the three images from docker hub using the below command.

docker pull ksrangar/services:v1

docker pull ksrangar/services:v2

docker pull ksrangar/services:v3

1. Tag the downloaded images to your docker account created in [docker hub](https://hub.docker.com/).

docker tag ksrangar /services:v1 <dockerhubuserid>/services:v1

docker tag ksrangar /services:v2 <dockerhubuserid>/services:v2

docker tag ksrangar /services:v3 <dockerhubuserid>/services:v3

We can validate whether the images are created correctly by following command

docker image ls

1. Push image to docker hub in account created using following command. It may prompt with credentials of your docker account which needs to be supplied for the execution.

docker push <dockerhubuserid>/services:v1

docker push <dockerhubuserid>/services:v2

docker push <dockerhubuserid>/services:v3

Navigate to the [docker hub](https://hub.docker.com/) to validate the images are available under your account.

1. Start the Minikube by using

minikube start

1. Navigate to the folder **code**
2. Run below commands to configure the MySQL

kubectl apply -f k8s/secret/mysql-srt.yaml  
kubectl apply -f k8s/config/mysql-cm.yaml  
kubectl apply -f k8s/volumes/mysql-pv.yaml  
kubectl apply -f k8s/services/mysql-sv.yaml  
kubectl apply -f k8s/deployments/mysql-dp.yaml  
  
kubectl run -it --rm --image=mysql:5.7.22 --restart=Never mysql-client -- mysql -h db -proot

This will create the secret which contains the credentials and apply to the MySQL and connect to the MySQL database.

1. Replace the <dockerhubuserid> with your docker hub id in following files

kubectl apply -f k8s/deployments/services-app-dp.yaml  
kubectl apply -f k8s/deployments/services-app-dp-v2.yaml  
kubectl apply -f k8s/jobs/migration.yaml

1. Run below commands to deploy Flask app.

kubectl apply -f k8s/services/services-app-sv.yaml  
kubectl apply -f k8s/deployments/services-app-dp.yaml  
kubectl apply -f k8s/deployments/services-app-dp-v2.yaml  
kubectl apply -f k8s/jobs/migration.yaml

If there is error while running k8s/jobs/migration.yaml do the following steps

Launch interactive shell using

kubectl exec --stdin --tty svc/services-app-service – sh

Then Run below commands

python manager.py db init

python manager.py db migrate

python manager.py db upgrade

The Kubernetes configuration is completed. Now the services should be up and running.

1. The configuration of Istio should be injected to Pods and should be restarted by running following

kubectl label namespace default istio-injection=enabled

# Restart all pods to get sidecar injected

kubectl delete pods –all

The pods can be validated by running

kubectl get pods

1. The configuration for the gateway and traffic split in Istio is done by running below commands

kubectl apply -f k8s/istio/gateway.yaml  
kubectl apply -f k8s/istio/traffic-splits/ts.yaml

1. Open a new terminal window and run below command so that the tunnel will be created with which the client can communicate. Never close the terminal which will stop processing request from Client.

minikube tunnel -c

1. To get the gateway end point run the below command which provides details. Run in them in new terminal Window

kubectl get svc -n istio-system -l app=istio-ingressgateway

This will provide the end point with which the traffic split between service can be tested by launching url in browser.

Open new terminal to validate the Istio Gateway and Virtual Service

#Istio Gateway

Kubectl get gw

#Istion Virtual Service

Kubectl get vs

1. Kiali which is part of Istio add-ons is used for the monitoring service health, how the traffic is split between services. It provides inbound and outbound logs along with tracing. This can be launched by running below command. Refer [here](#_1:_Kiali_(Visualization) for screenshots

istioctl dashboard kiali

1. Prometheus which is part of Istio add-ons is open source monitoring system and time series database. This should be running such that Grafana can be used for viewing dashboard. This can be validated by running below command.

kubectl -n istio-system get svc prometheus

1. Grafana which is part of Istio provide the real-time dashboard. This are pre-defined dashboard for Istio which can be used for monitor the health of Istio and its services. Refer [here](#_2:_Grafana_(Visualization) for screenshots. Below commands should be run before you open in browser.

kubectl -n istio-system get svc Grafana

istioctl dashboard grafana

Dashboard can be viewed by launching <http://localhost:3000/dashboard/db/istio-mesh-dashboard> in your web browser.

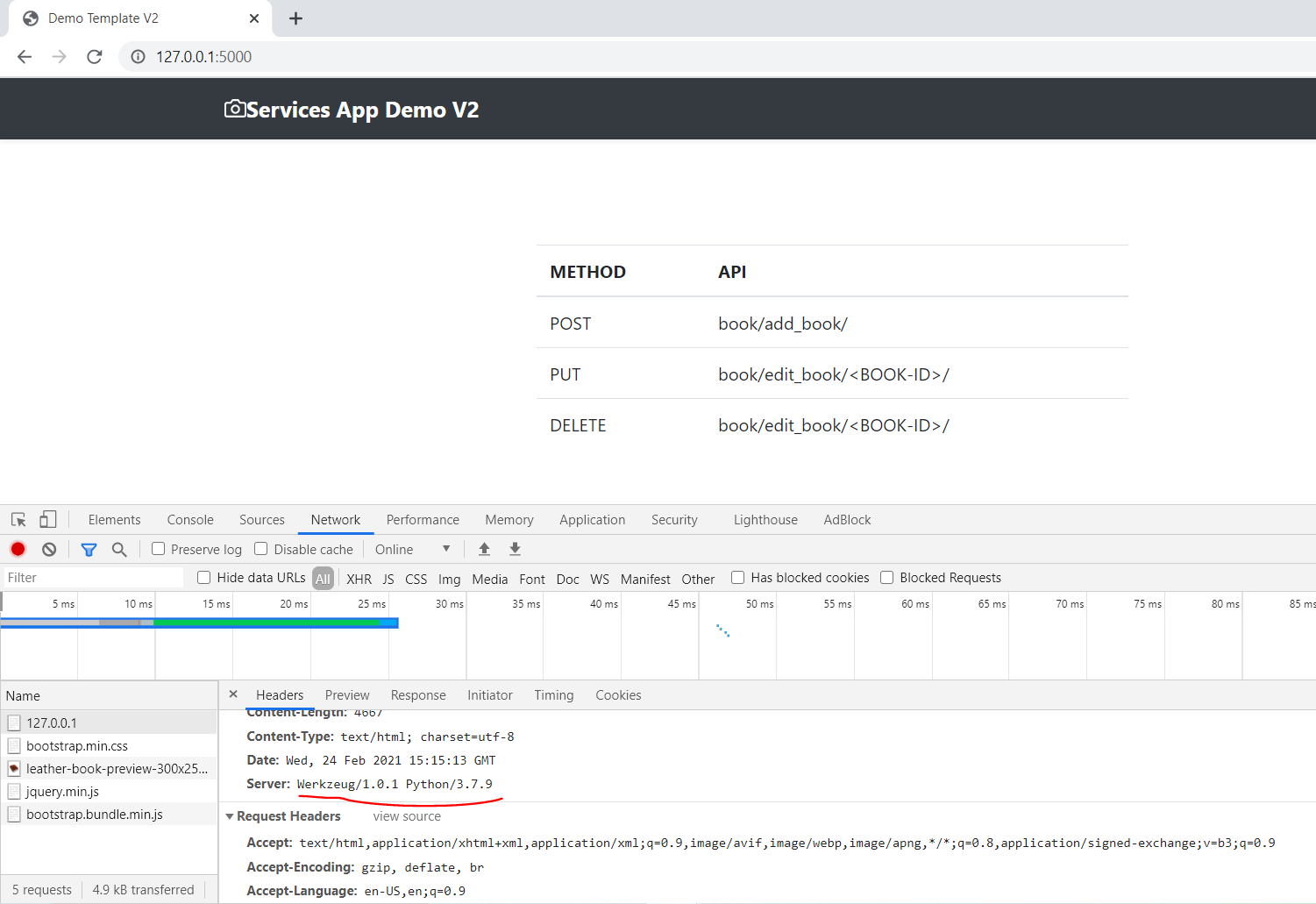
1. Jager which is part of Istio is used for tracing the application. Run the below command to start collecting trace.

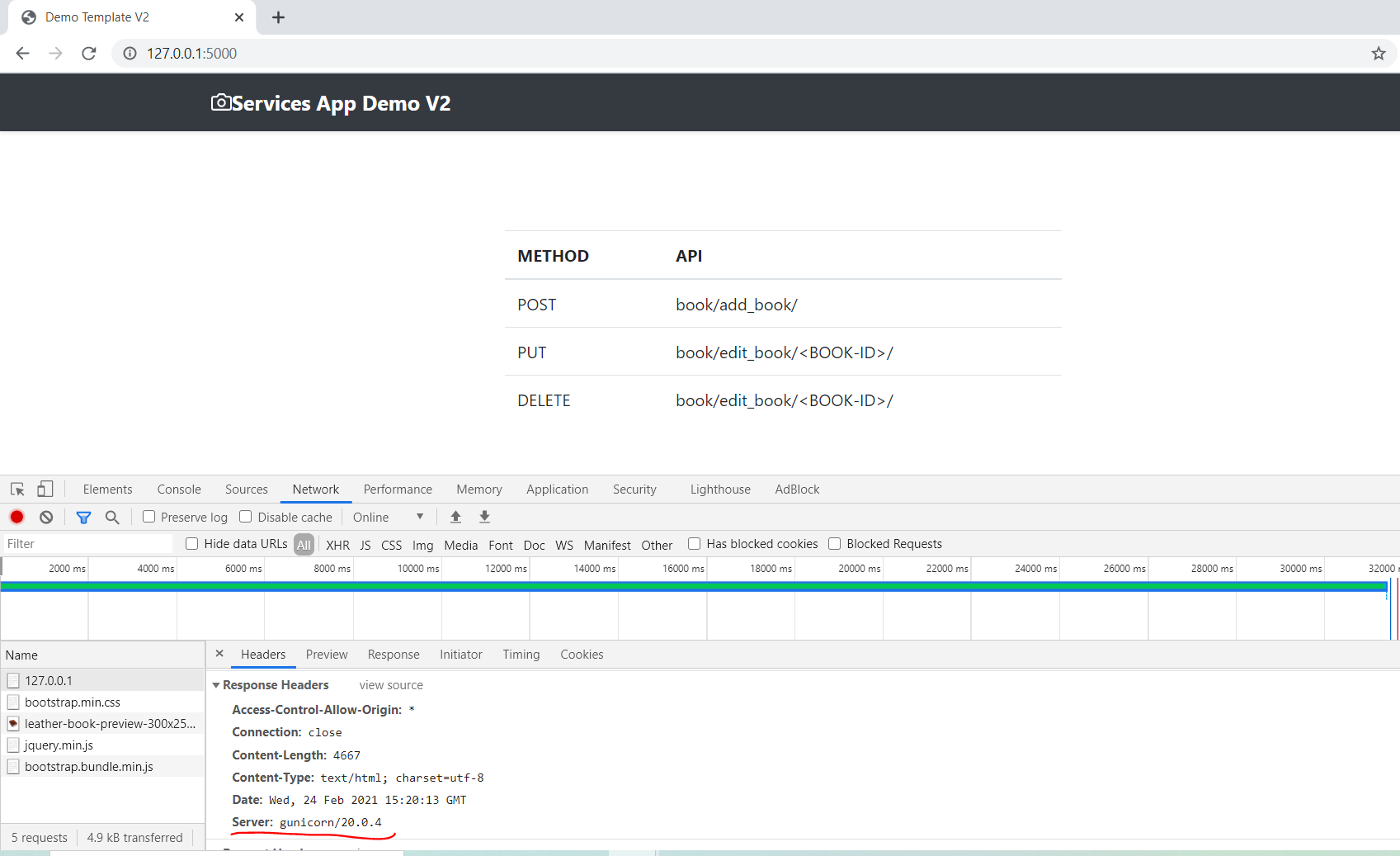
istioctl dashboard jaeger

Dashboard can be viewed by launching <http://localhost:16686/jaeger> in your web browser. Refer [here](#_4:_Jaeger_(Service) for screenshots.

**Service Mesh POC**

**1: Default Inbuild Flask Server:**



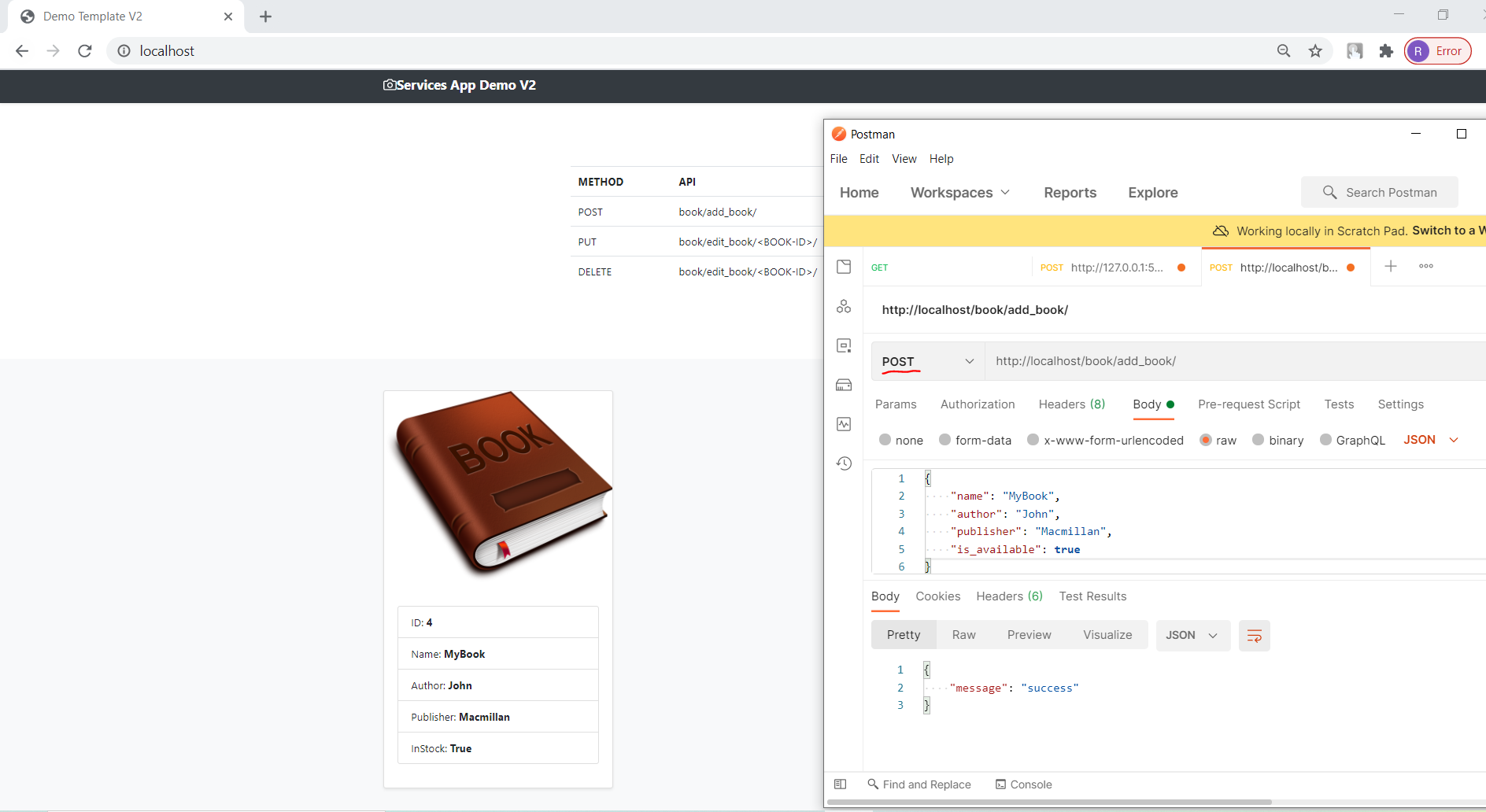
**2: Gunicorn Server:** 

**3: Nginx Server on Port 80:**

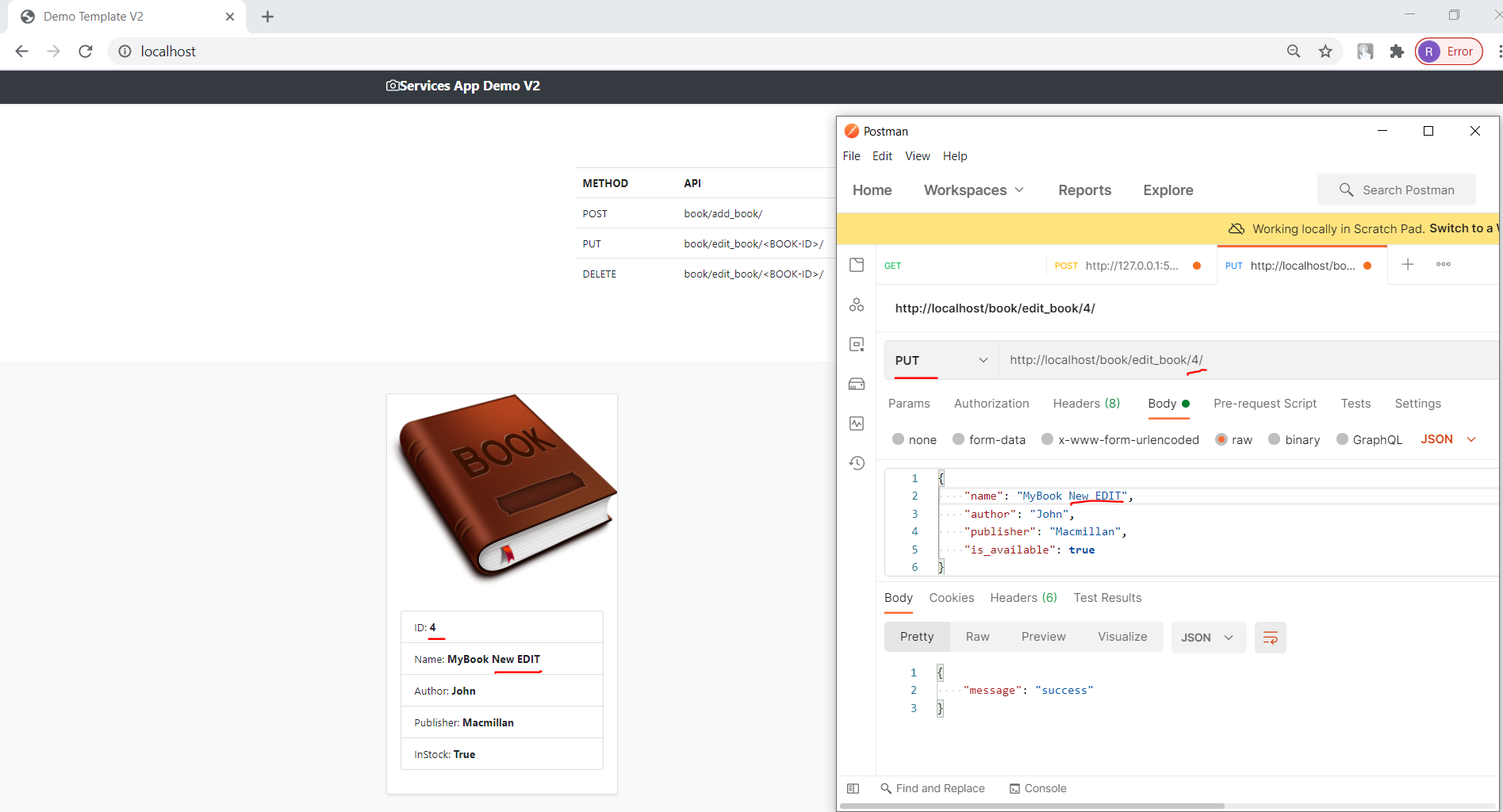


**REST Operations:**

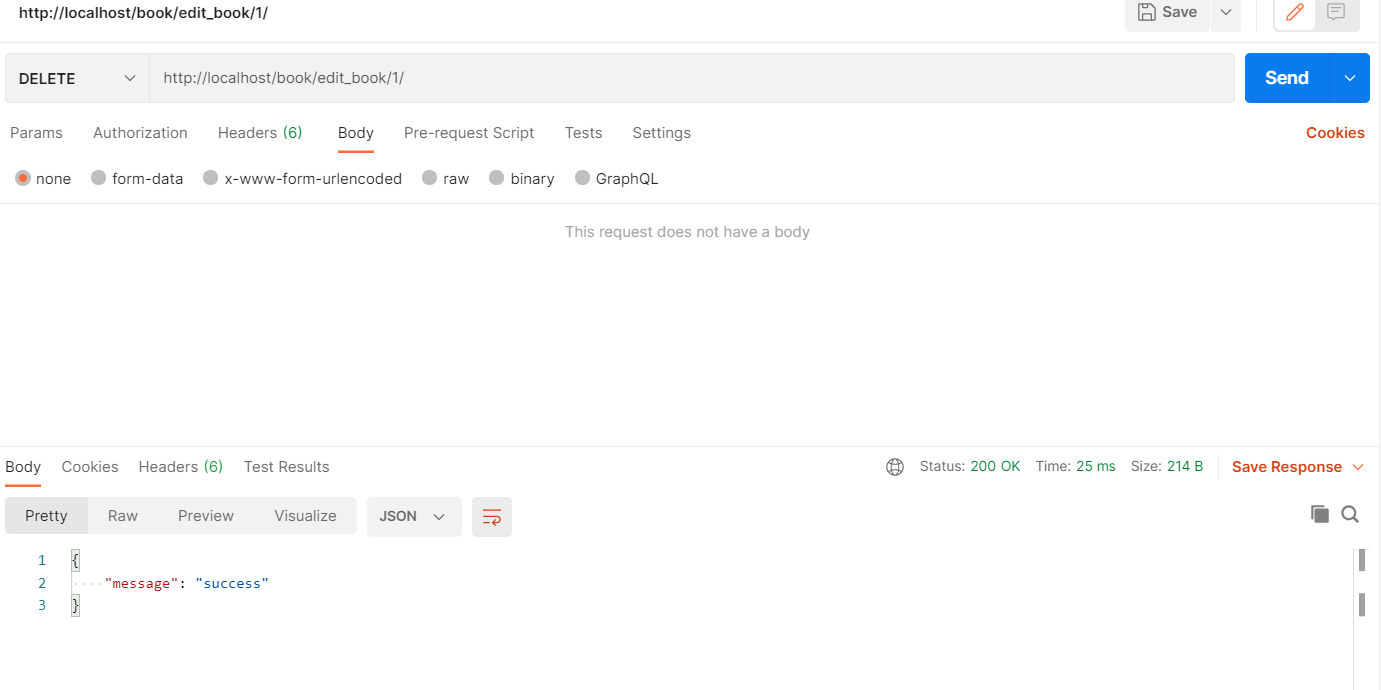
**1: POST:**

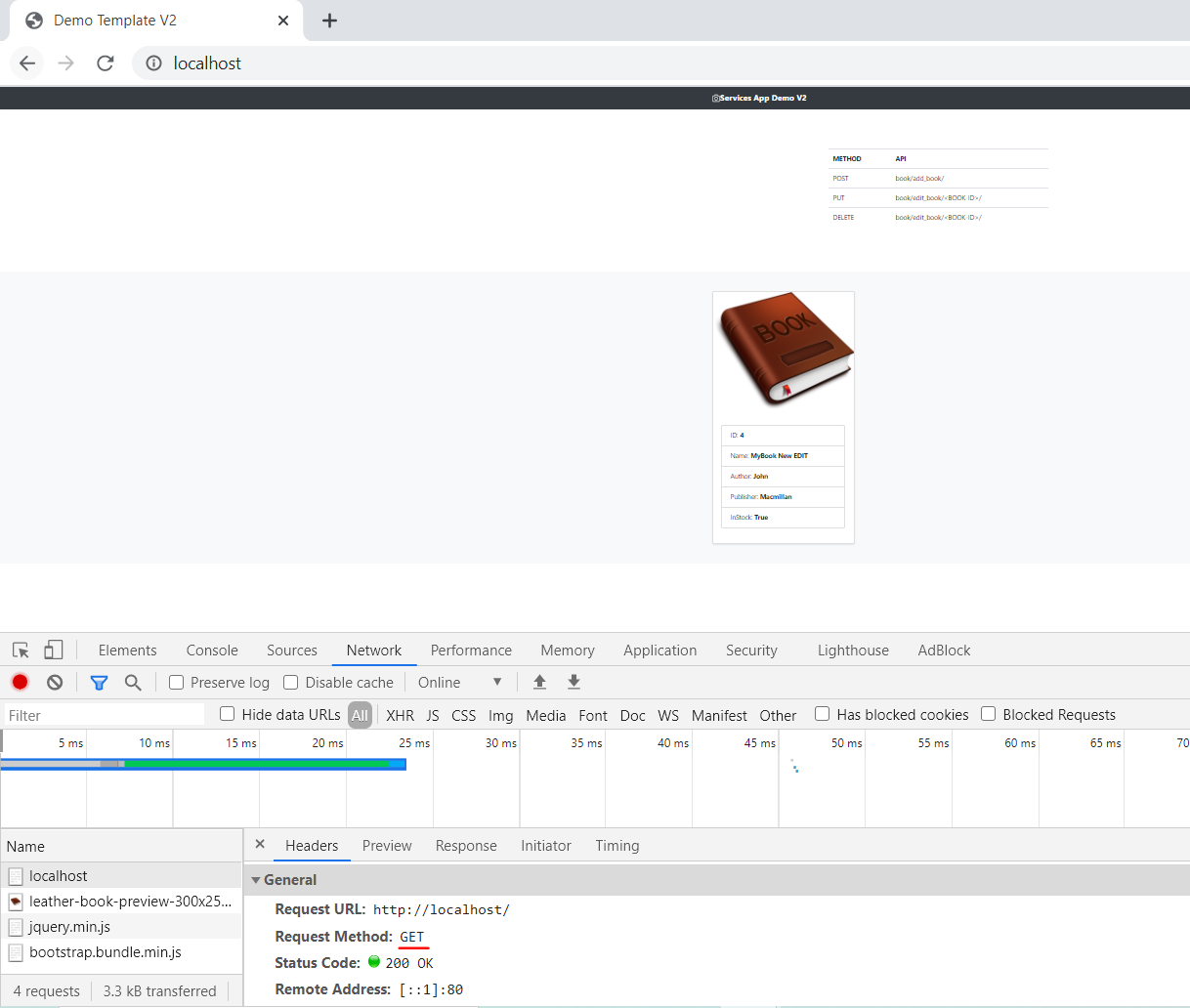


**2: PUT:**



**3: Delete:**



**4: GET (**Default method on UI home page**):**

**Nginx**

**1: Nginx as LoadBalancer:**

upstream services\_manager {

    server backend\_v1:5000;

    server backend\_v2:5001;

}

server {

    listen 80;

    location / {

        proxy\_pass http://services\_manager;

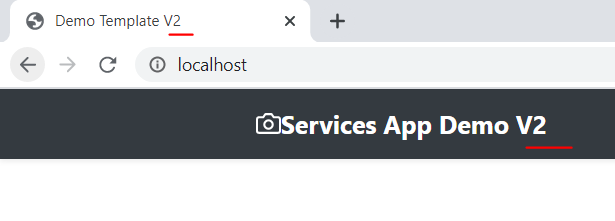
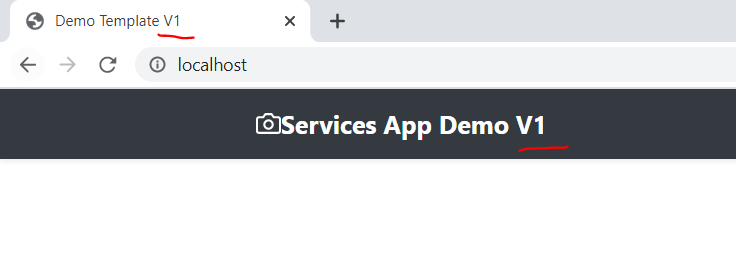
        proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

        proxy\_set\_header Host $host;

        proxy\_redirect off;

    }

}

* ***Default Waterfall Model***:
  + 

**2: Nginx as API-GateWay:**

upstream add\_book {

    server backend\_v1:5000;

}

upstream edit\_book {

    server backend\_v2:5001;

}

# LoadBalancer

upstream services\_manager {

    server backend\_v1:5000;

    server backend\_v2:5001;

}

server {

    listen 80;

    location / {

        proxy\_pass http://services\_manager;

        proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

        proxy\_set\_header Host $host;

        proxy\_redirect off;

    }

    # Add Book TO DB

    location /book/add\_book/ {

        proxy\_pass http://add\_book;

    }

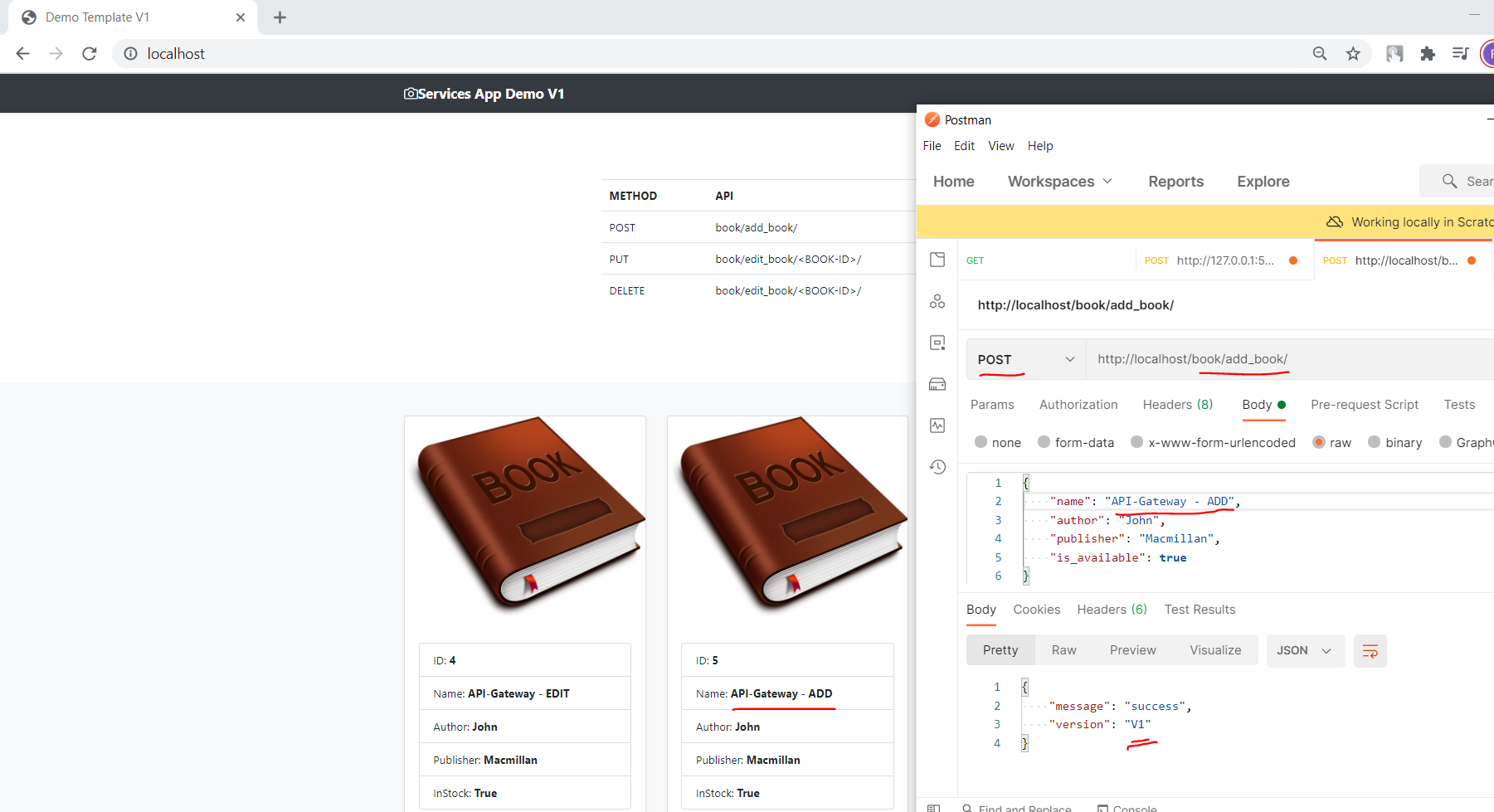
    # Edit & Delete book from DB

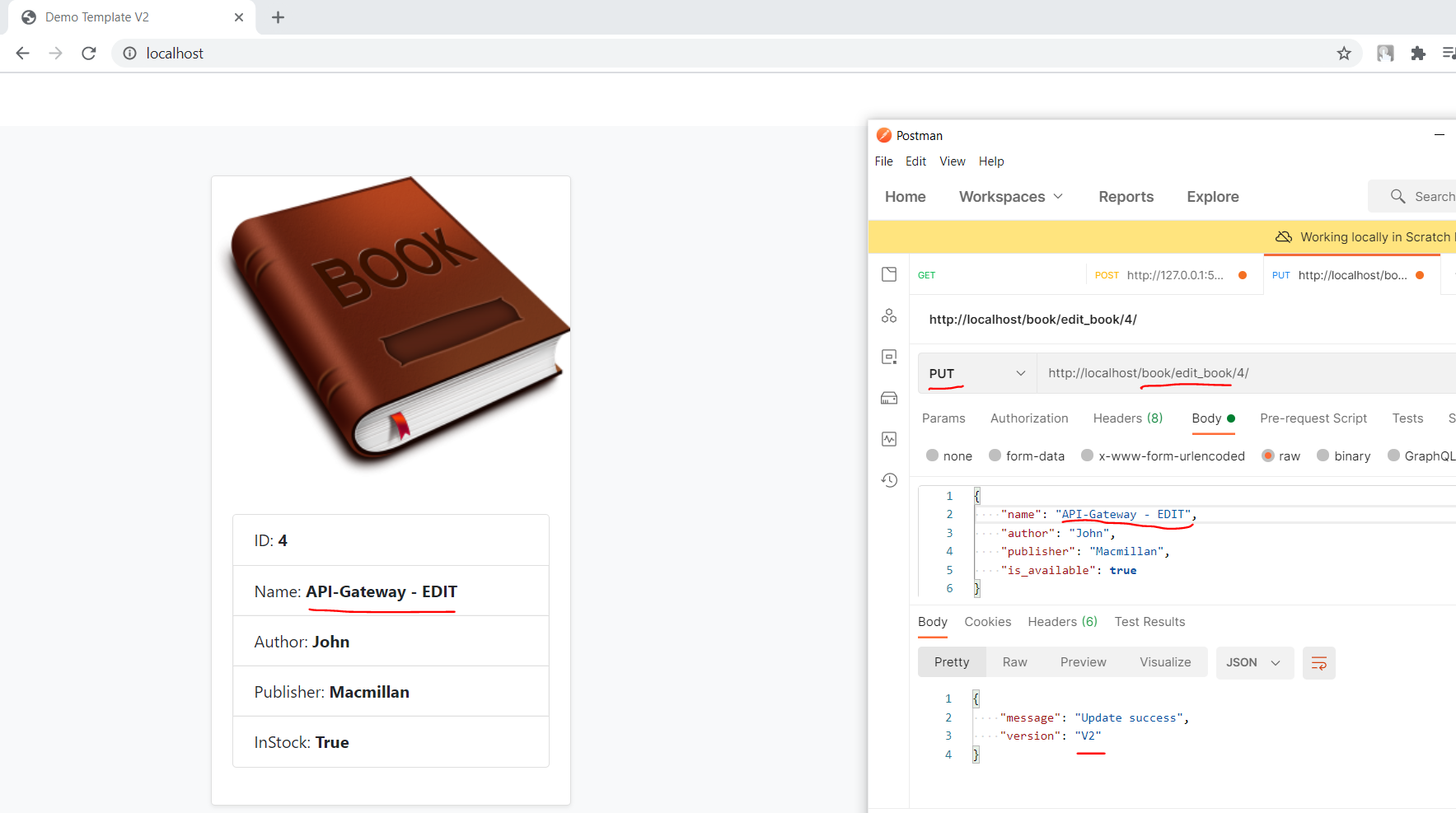
    location /book/edit\_book/ {

        proxy\_pass http://edit\_book;

    }

}

**ADD Record (POST):**

**EDIT Record (PUT):**

**3: Nginx as Webserver:**

upstream services\_manager {

    server backend:5000;

}

server {

    listen 80;

    location / {

        proxy\_pass http://services\_manager;

        proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

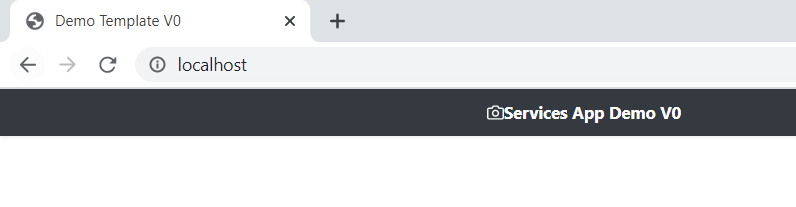
        proxy\_set\_header Host $host;

        proxy\_redirect off;

    }

}

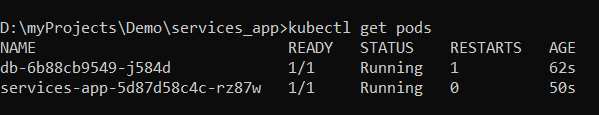
***Single Instance***



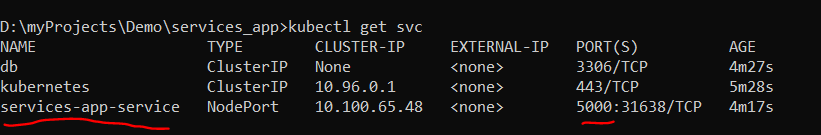
Note: Demo on Docker

[**Kubernetes**](https://kubernetes.io/)

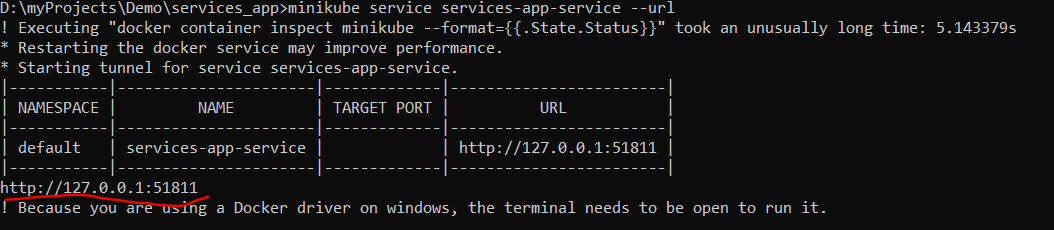
***1: Create Application and Database pods in K8s cluster***



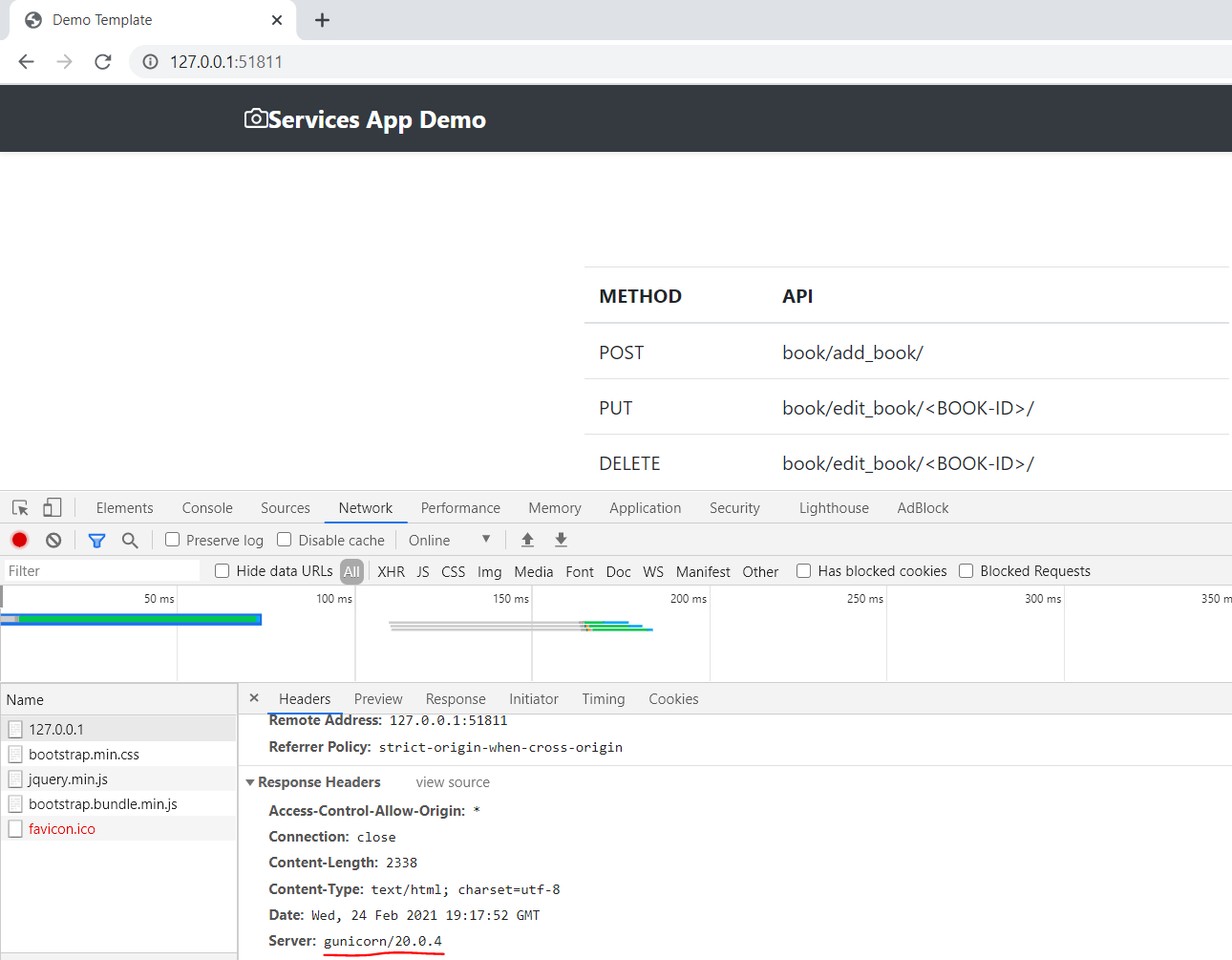
***2: Get Running Service for Application and Database***



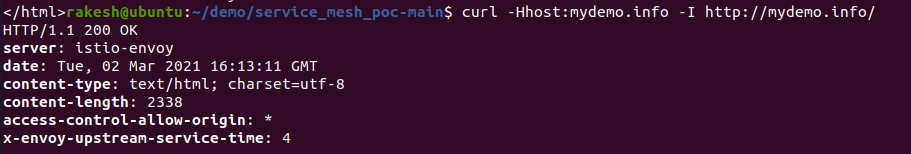
***3: Access Application inside K8s Cluster***



***Demo:***



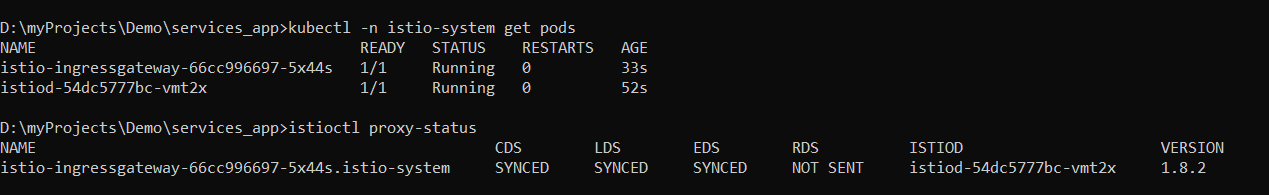
***4: Ingress controller:***



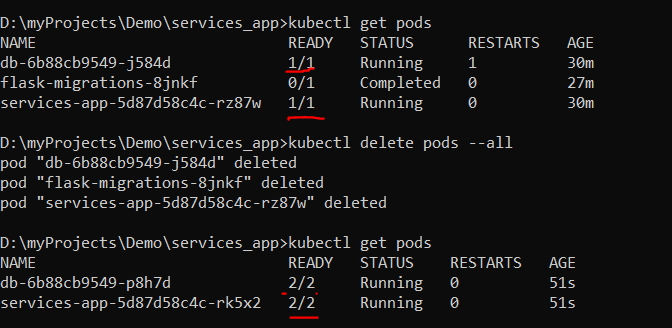
***Note: Demo on Minikub, Ubuntue***

[**Istio**](https://istio.io/latest/) **(Service Mesh):**

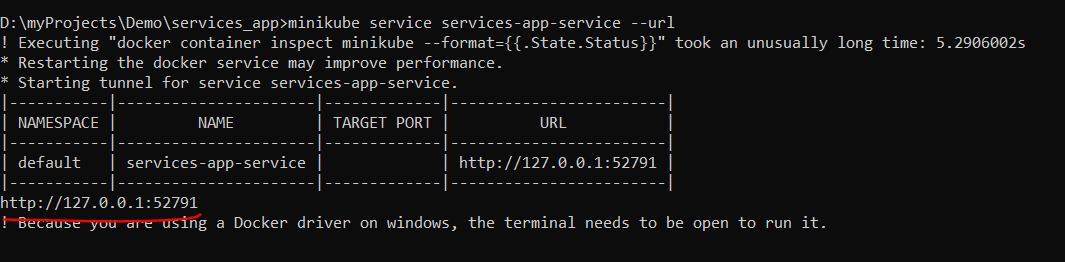
***1: Istio Pods and Service:***

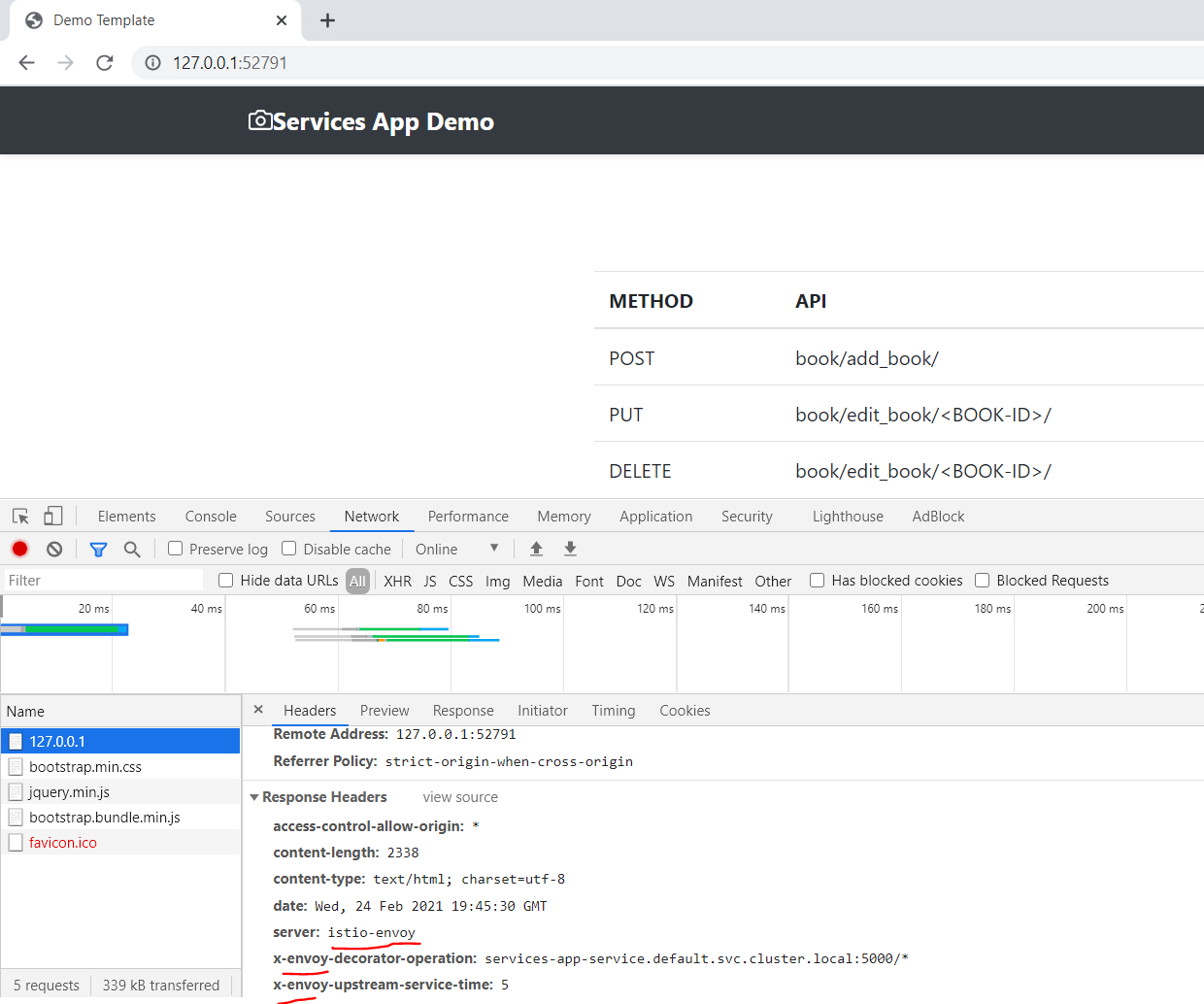


***2: Inject Istio side car (Envoy) to Application and DB pods.***

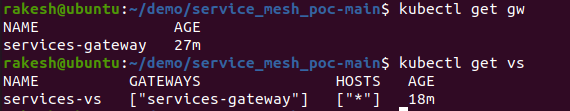


***3: Access Application inside K8s Cluster via Istio-envoy:***



***Demo:***

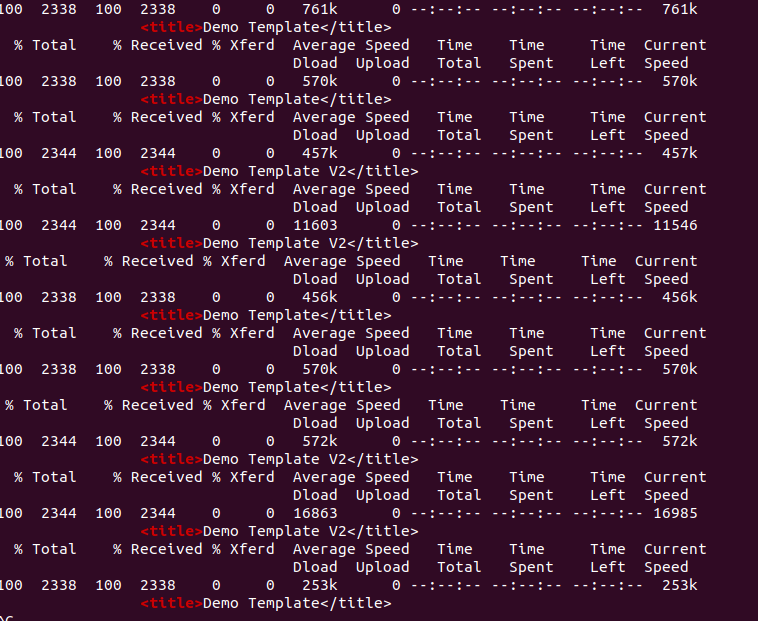
***4: Istio Gateway & VirtualService:***



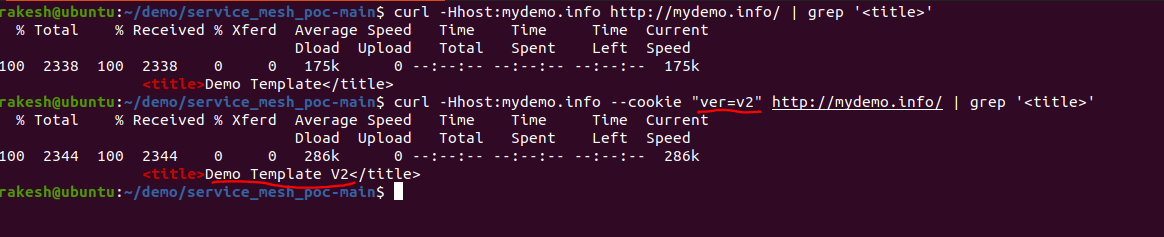
***5: Simple snippet to load test request and response***



***5:*** [***Traffic Split***](https://istio.io/latest/docs/tasks/traffic-management/traffic-shifting/) ***(50%(Demo Template)-50%(Demo Template V2)):***



***6: Canary deployment (If Cookie is set to v2 then redirect to V2 Service)***



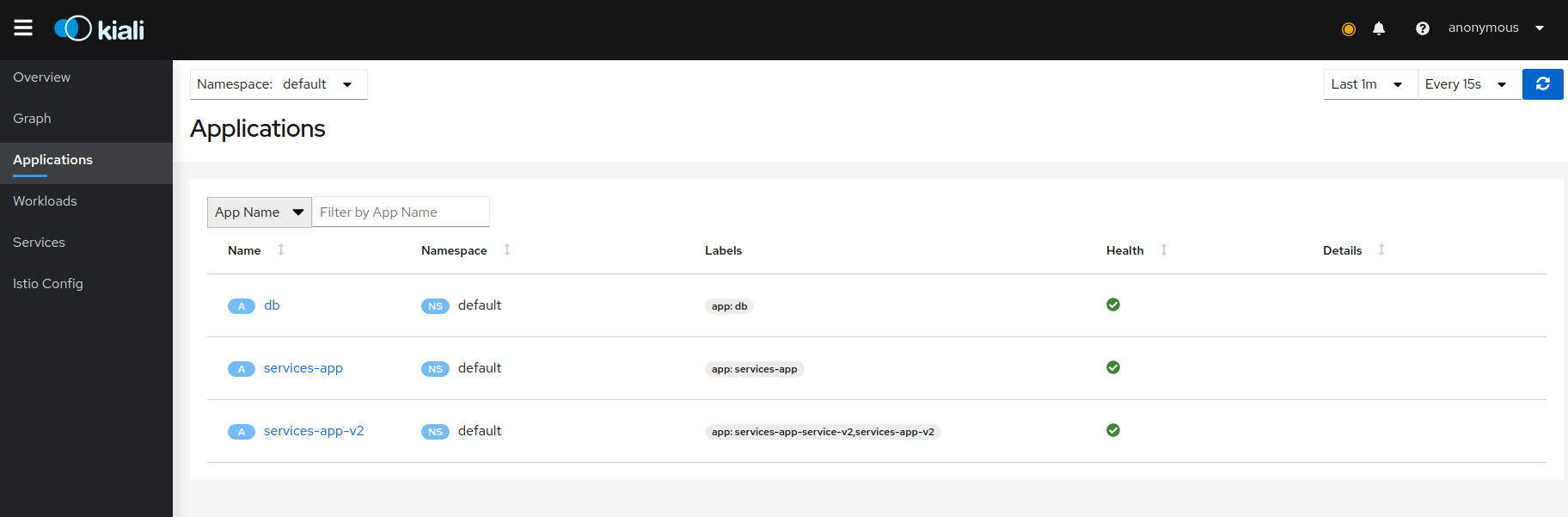
***TODO***

1. [***Explore other options***](https://istio.io/latest/docs/tasks/traffic-management/)

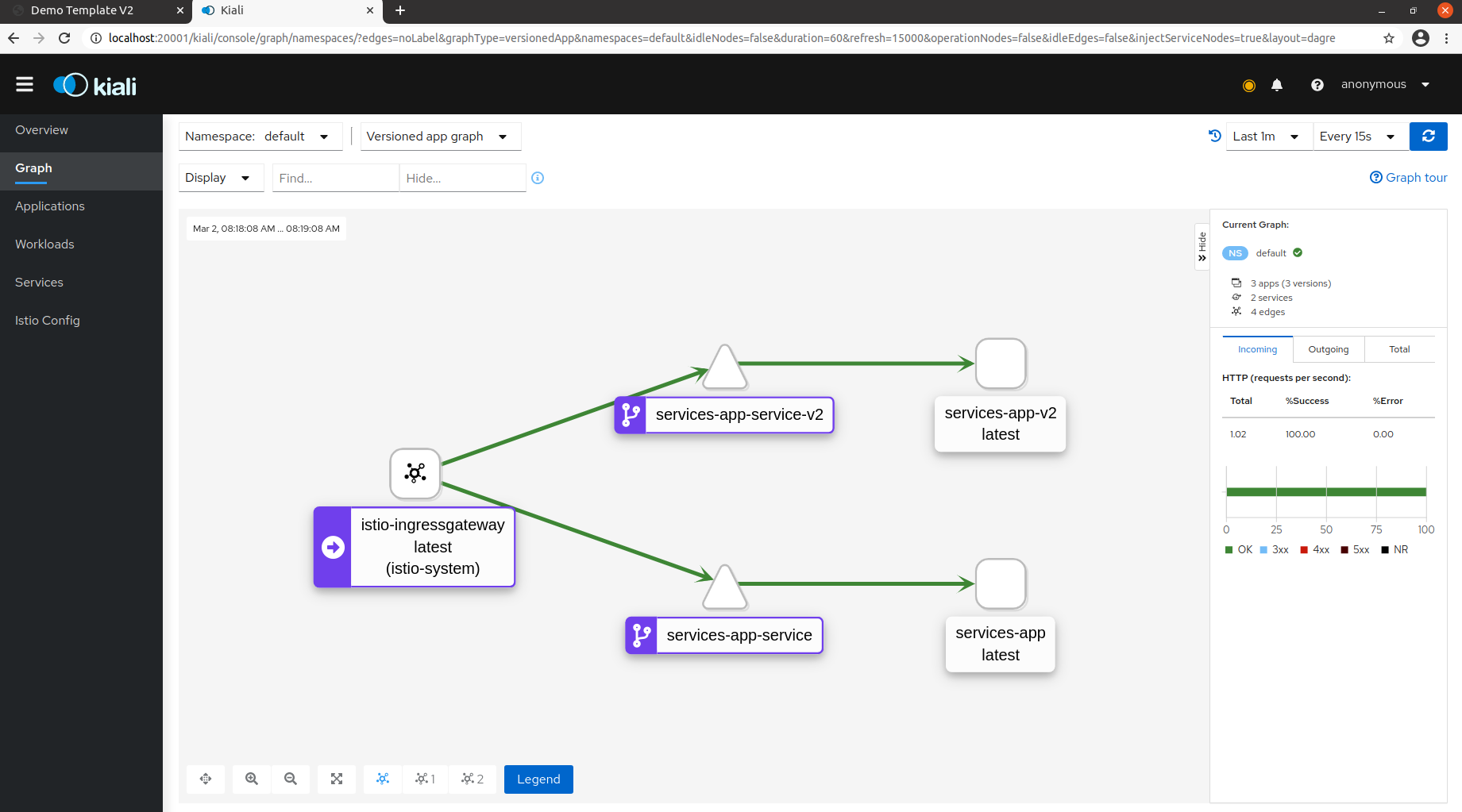
***Note: Demo on Minikube, Ubuntu***

# [Monitoring &Visualization](https://istio.io/latest/docs/tasks/observability/)

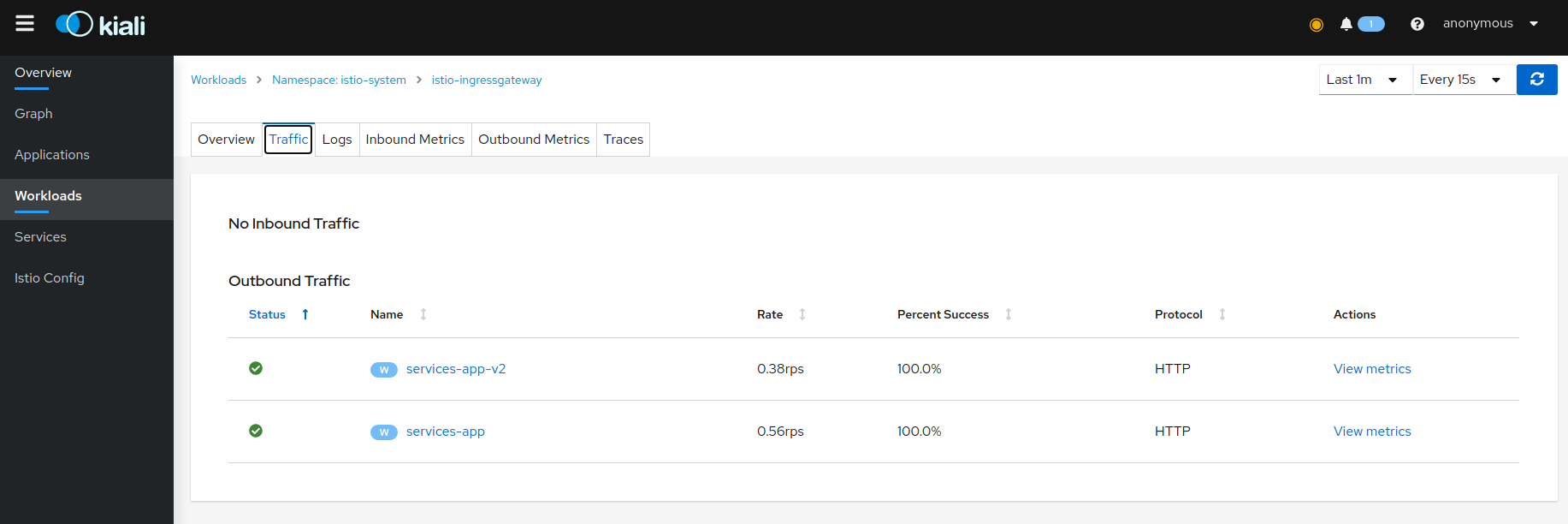
## 1: [***Kiali***](https://kiali.io/) (Visualization Tool) 🡪 Real Time



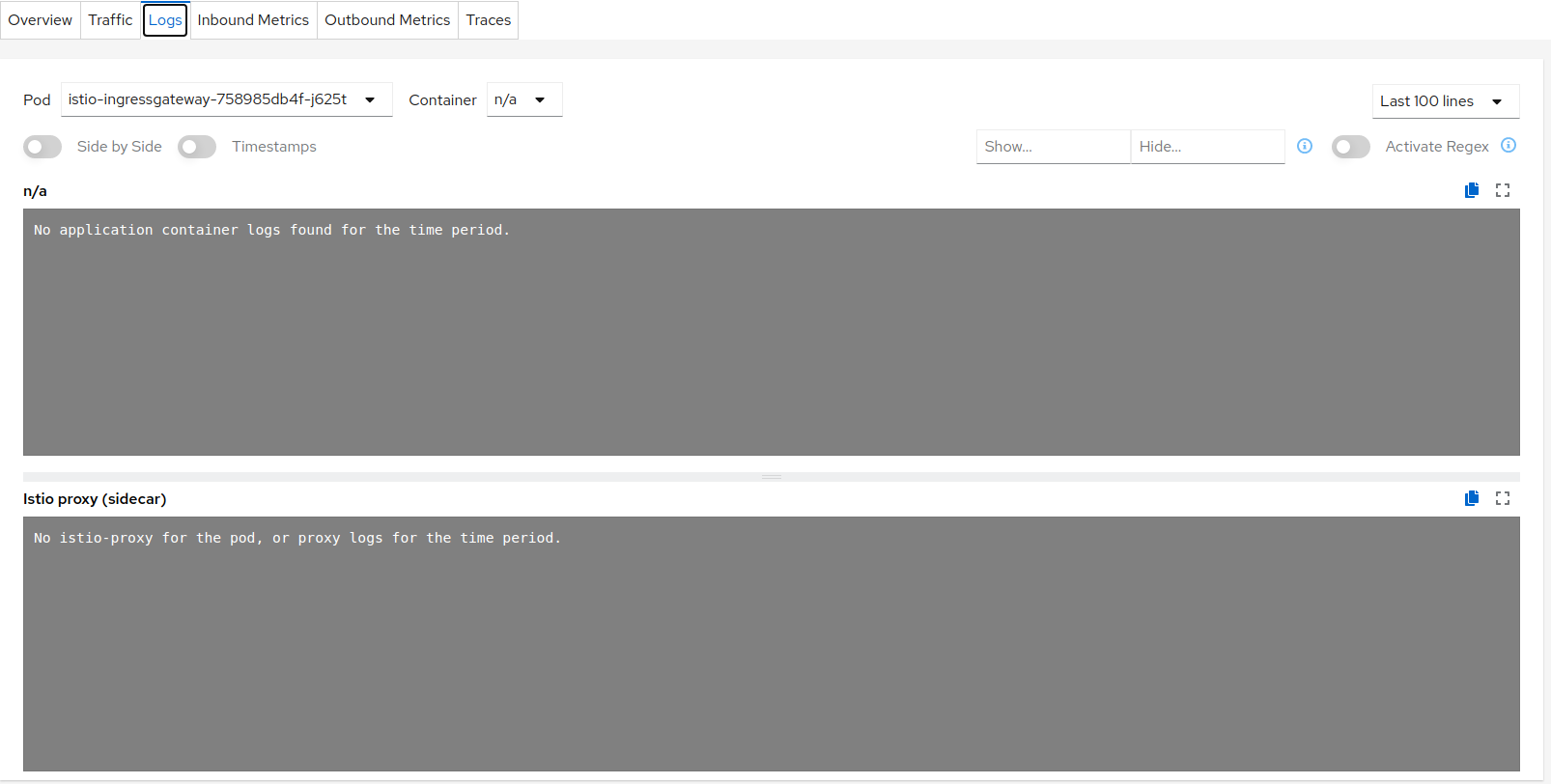
***Graph of request direction 50% to Service V1 and 50% to Service V2***



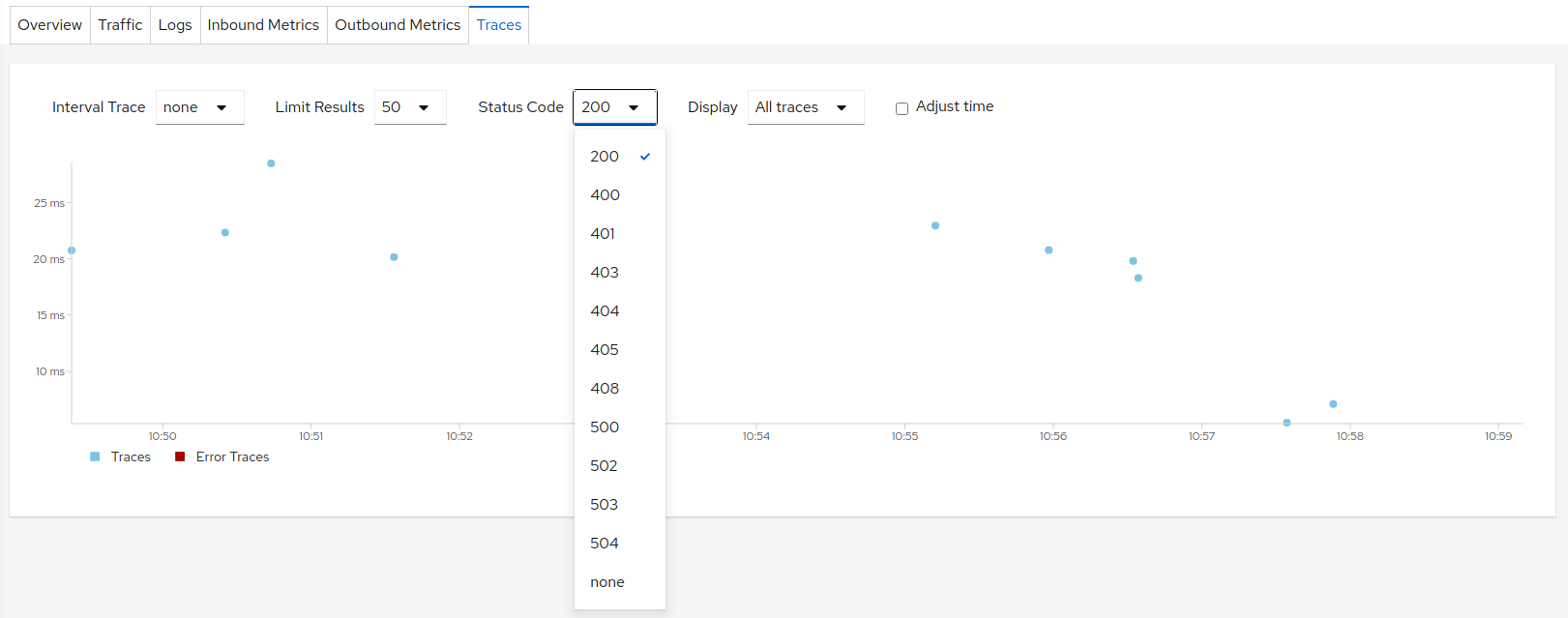
***Service Health***



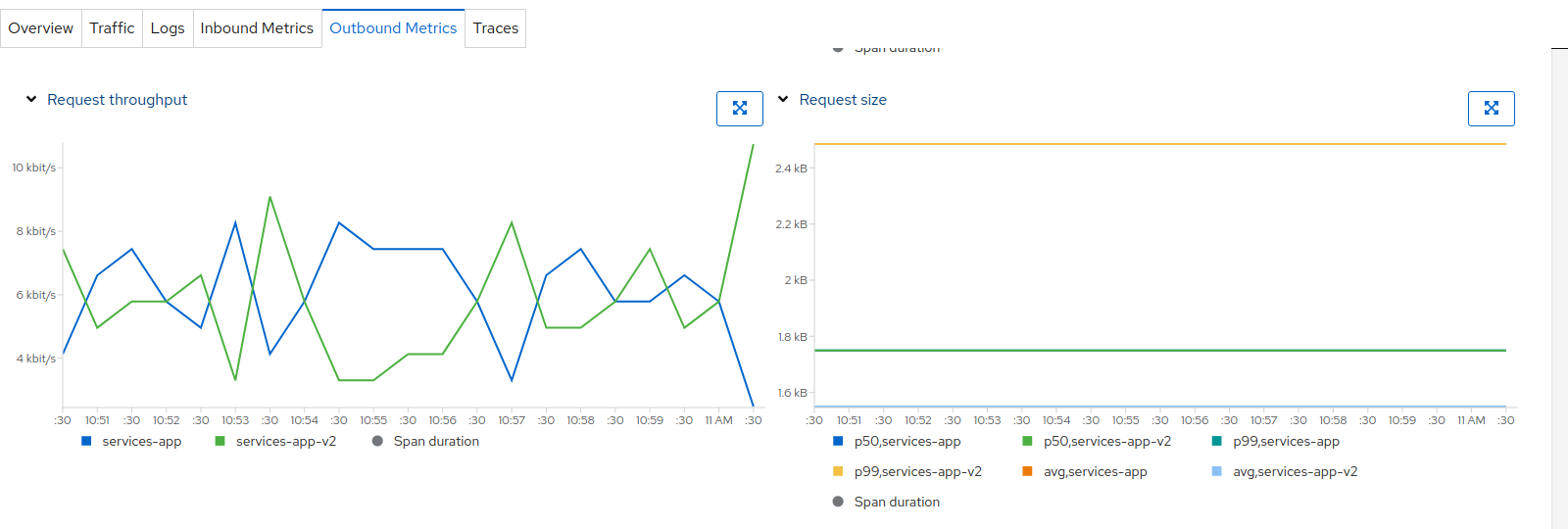
***Logs From Pods (Containers) (Not Configured Currently -TODO)***

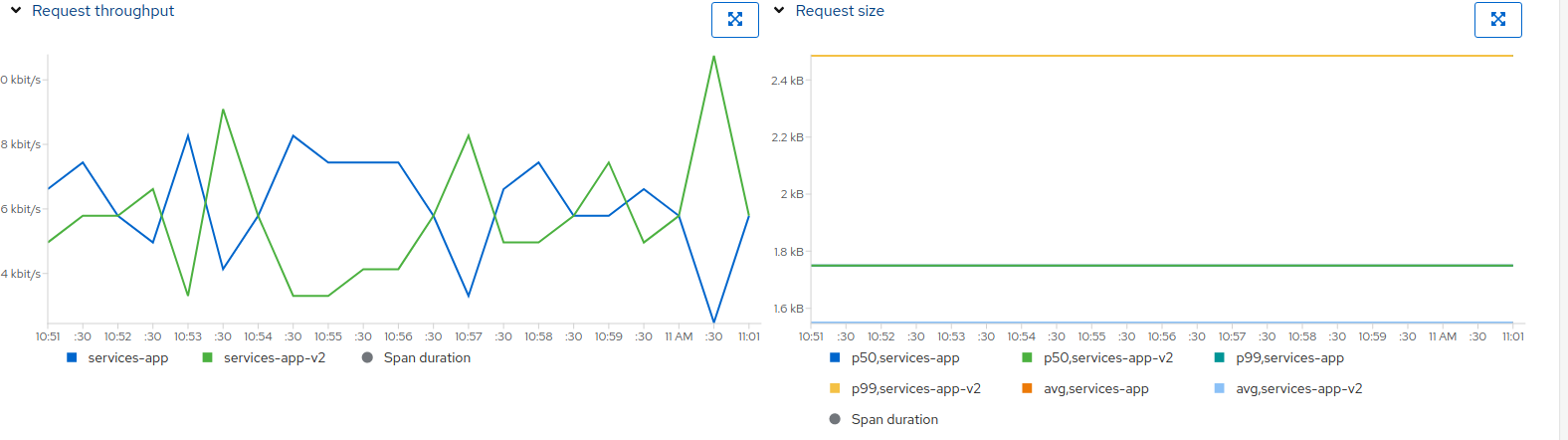


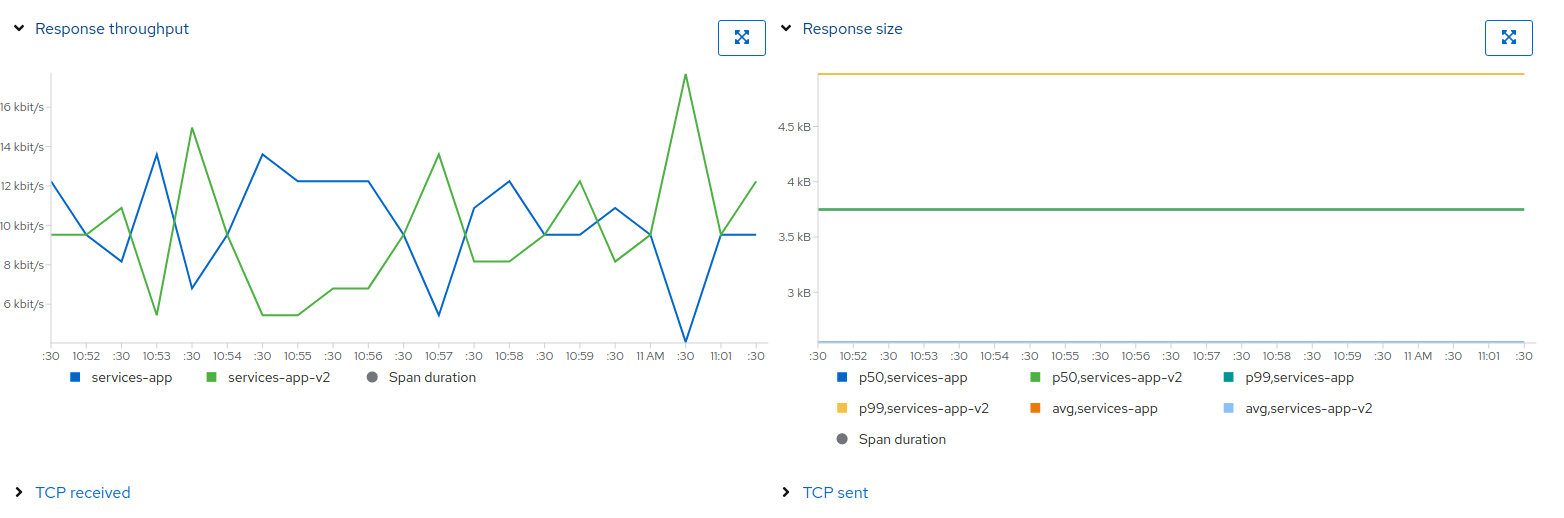
***Request Tracing (status codes and more)***



***Outbound Metrics:***



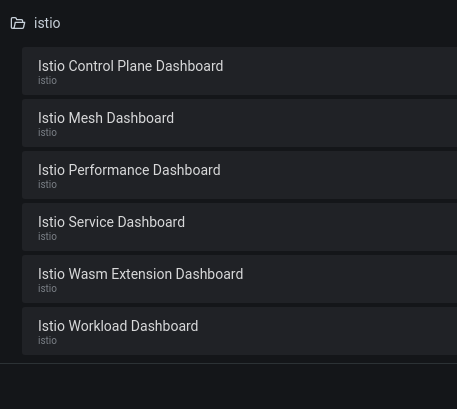




***Note: Demo on Minikub, Ubuntu***

## 2: [***Grafana***](https://grafana.com/) (Visualization Tool) 🡪 Real Time:

***List of available metrics and real time monitoring and visualization***

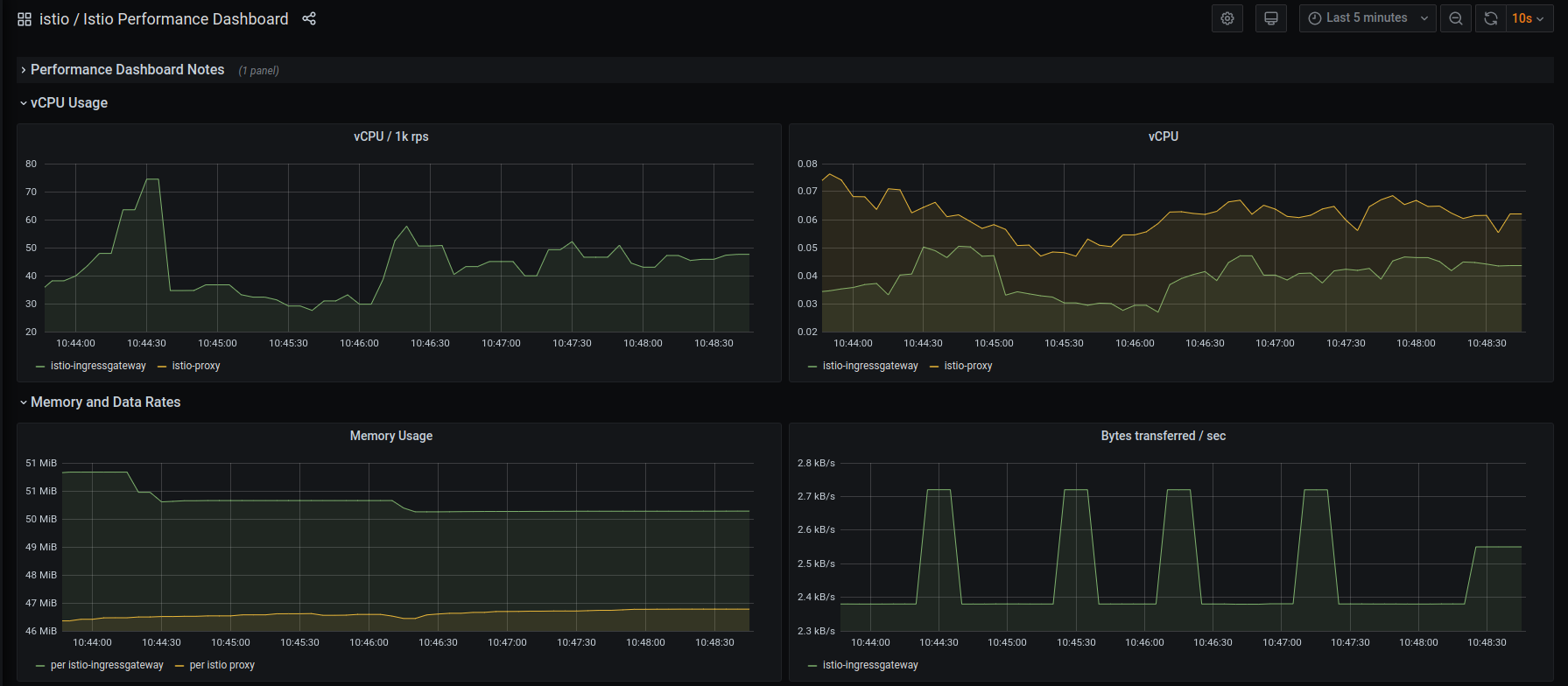


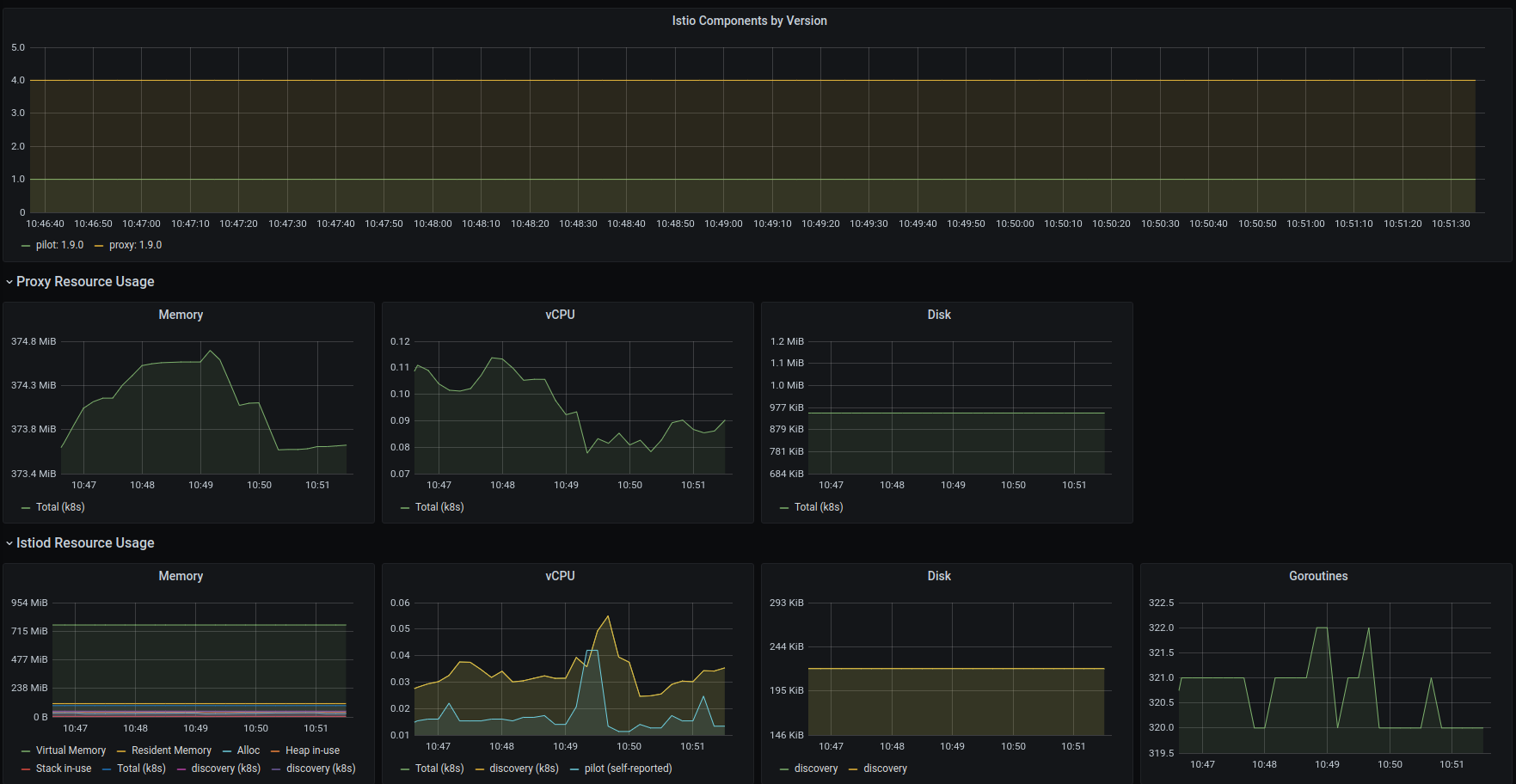
***Control Plane Dashboard***



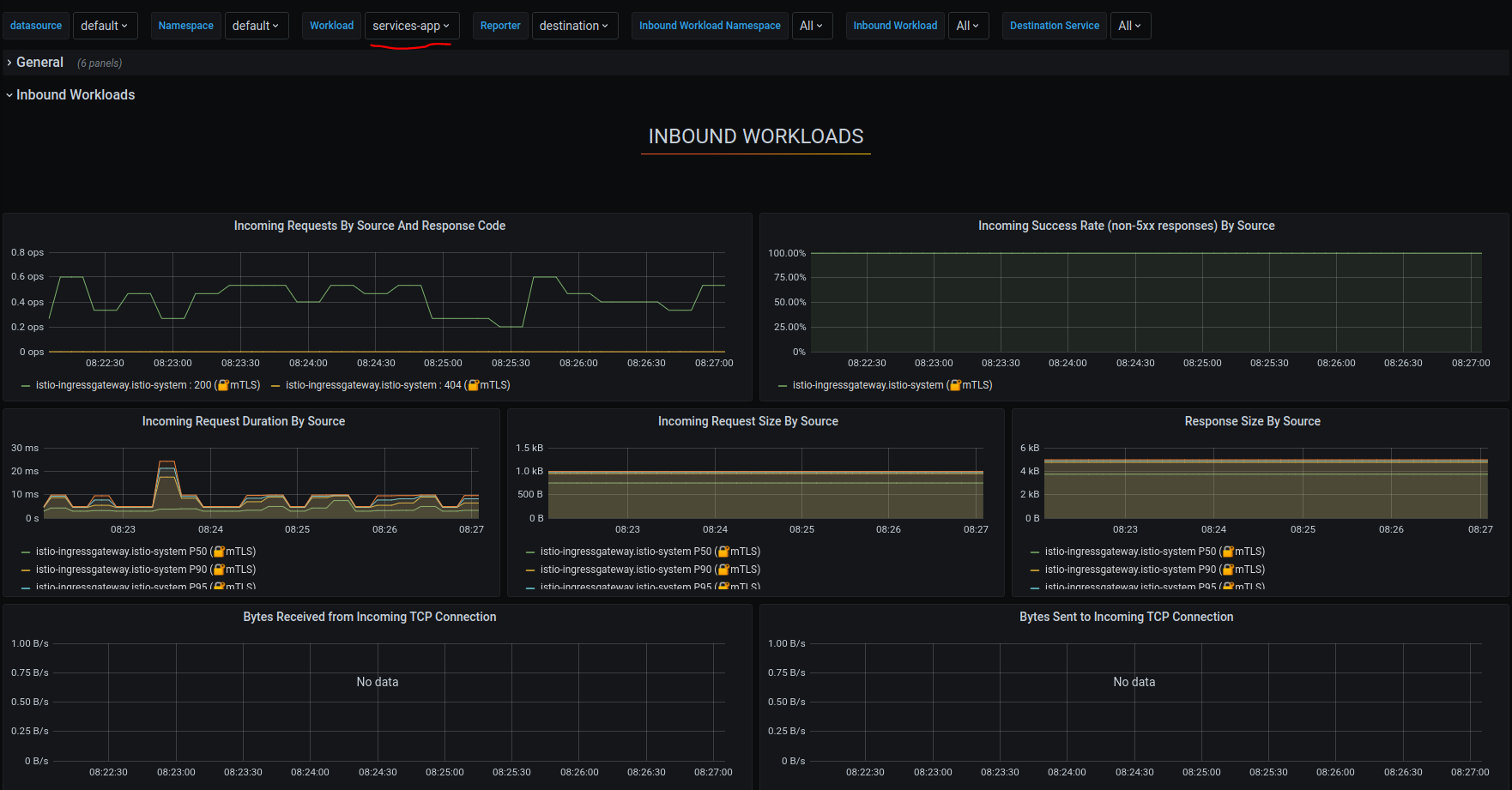


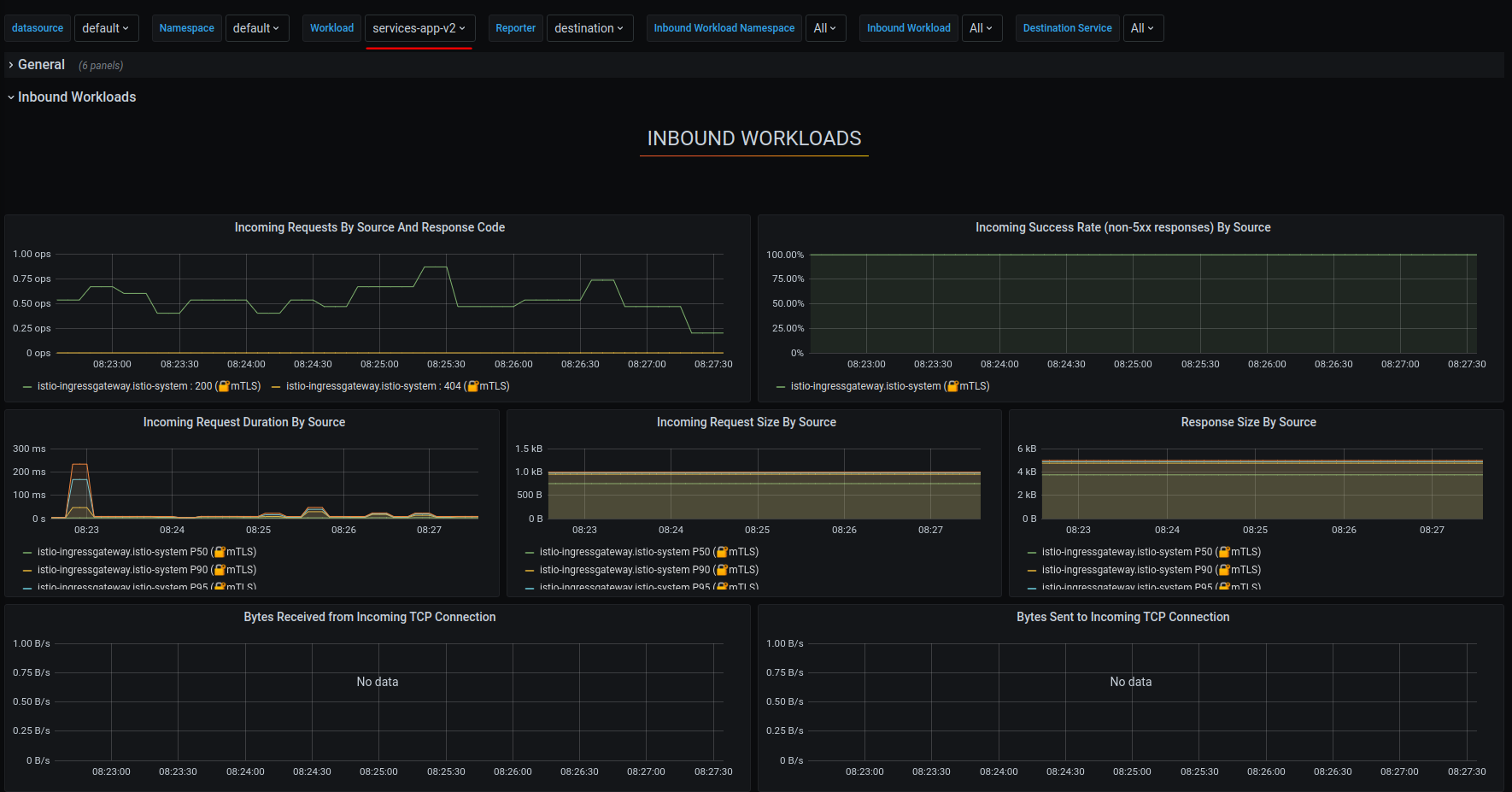
***Performance Dashboard (Metrics for CPU, Memory, IO utilization)***

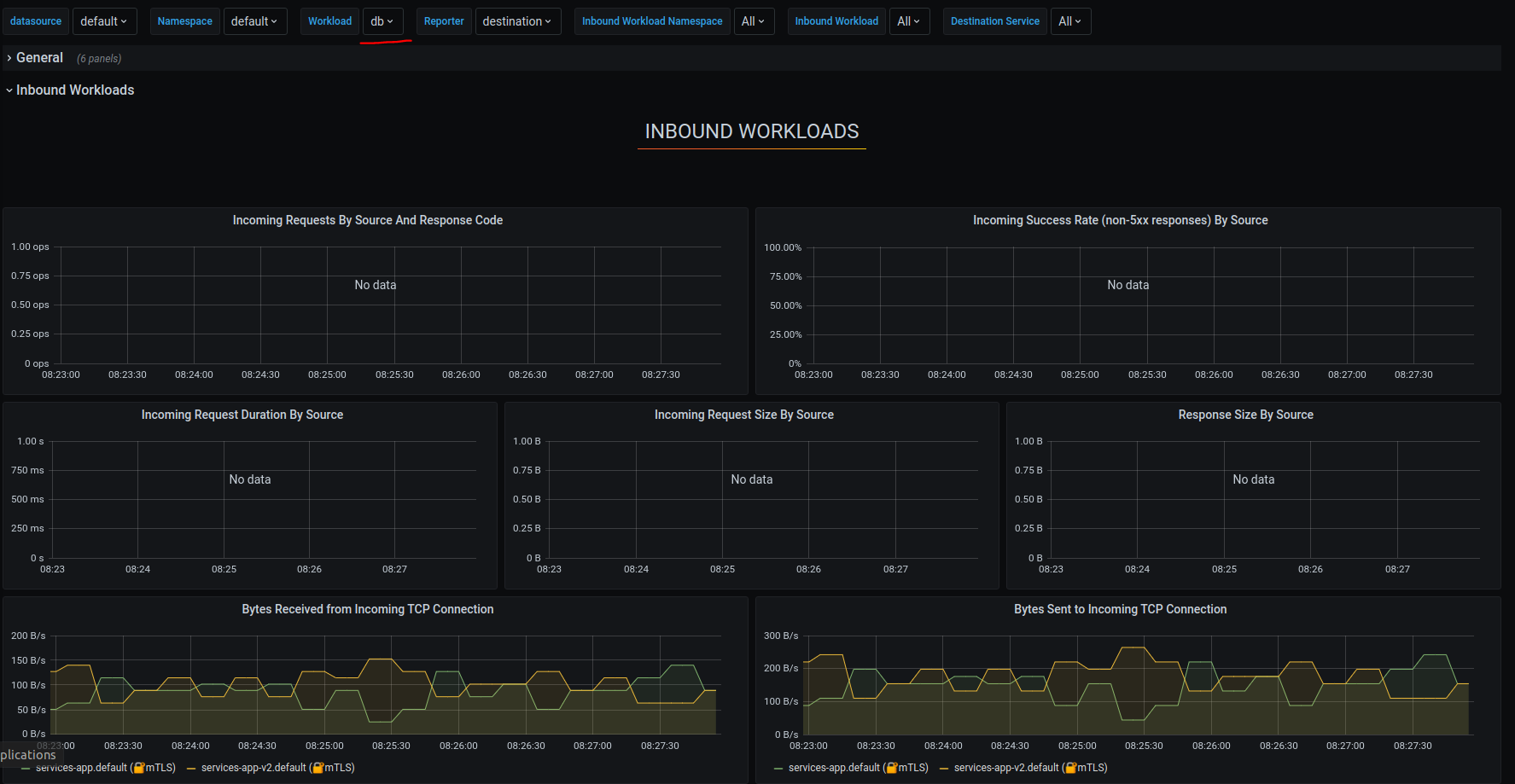




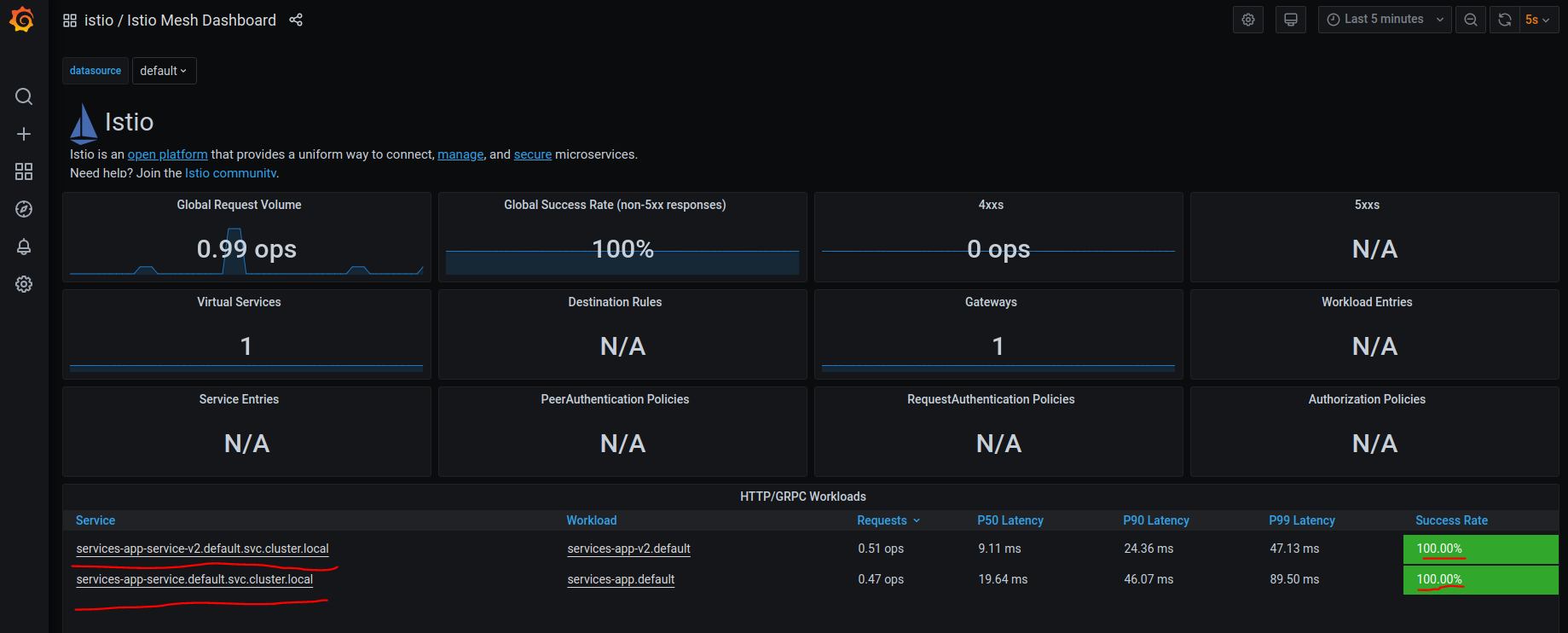
***Inbound Workload***





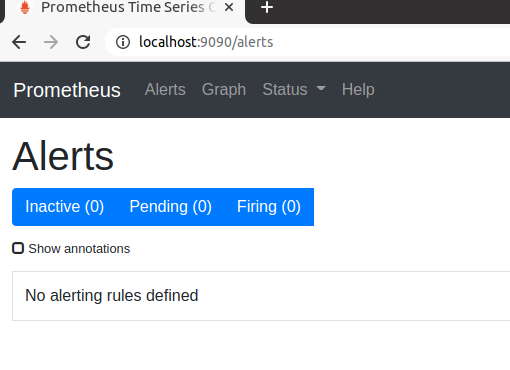


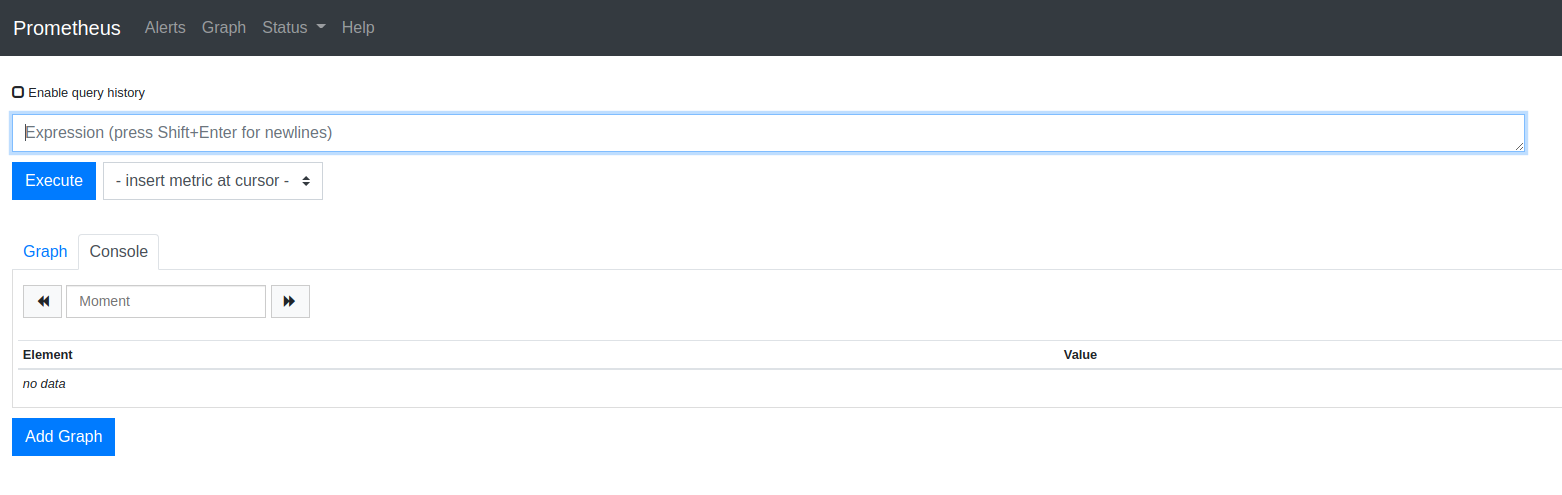
***Service Health***



***Note: Demo on Minikub, Ubuntu***

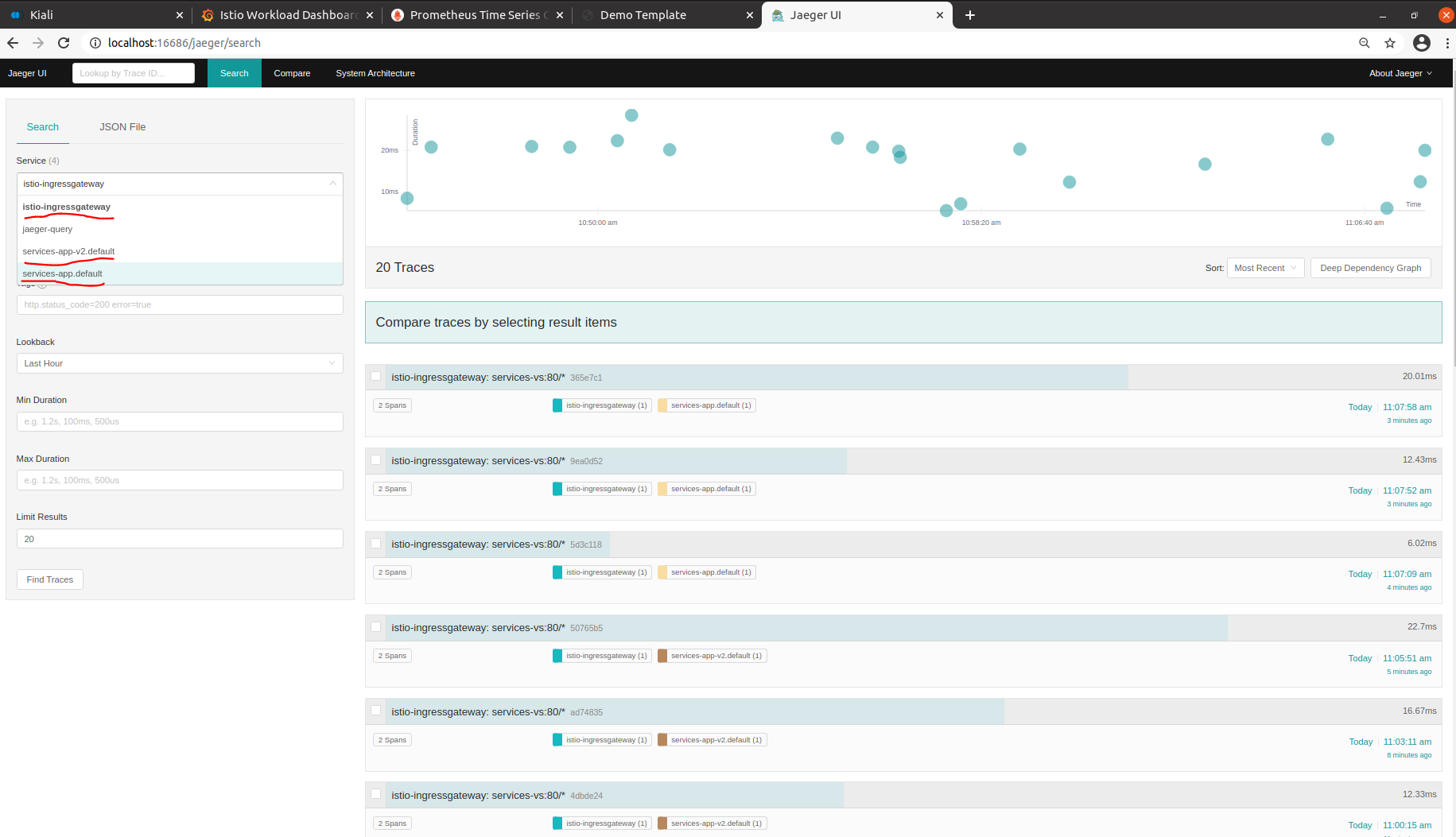
## 3: [***Prometheus***](https://prometheus.io/) (Monitoring): 🡪 (TODO:Explore Options, Alerts, Monitoring)





***Note: Demo on Minikub, Ubuntu***

## 4: [***Jaeger***](https://www.jaegertracing.io/) (Service Tracing)(TODO:Explore Options)



***Note: Demo on Minikub, Ubuntu***

***TODO***

1. ***Zipkin (Service Tracing)***
2. ***Service Discovery***
3. ***Log Management***