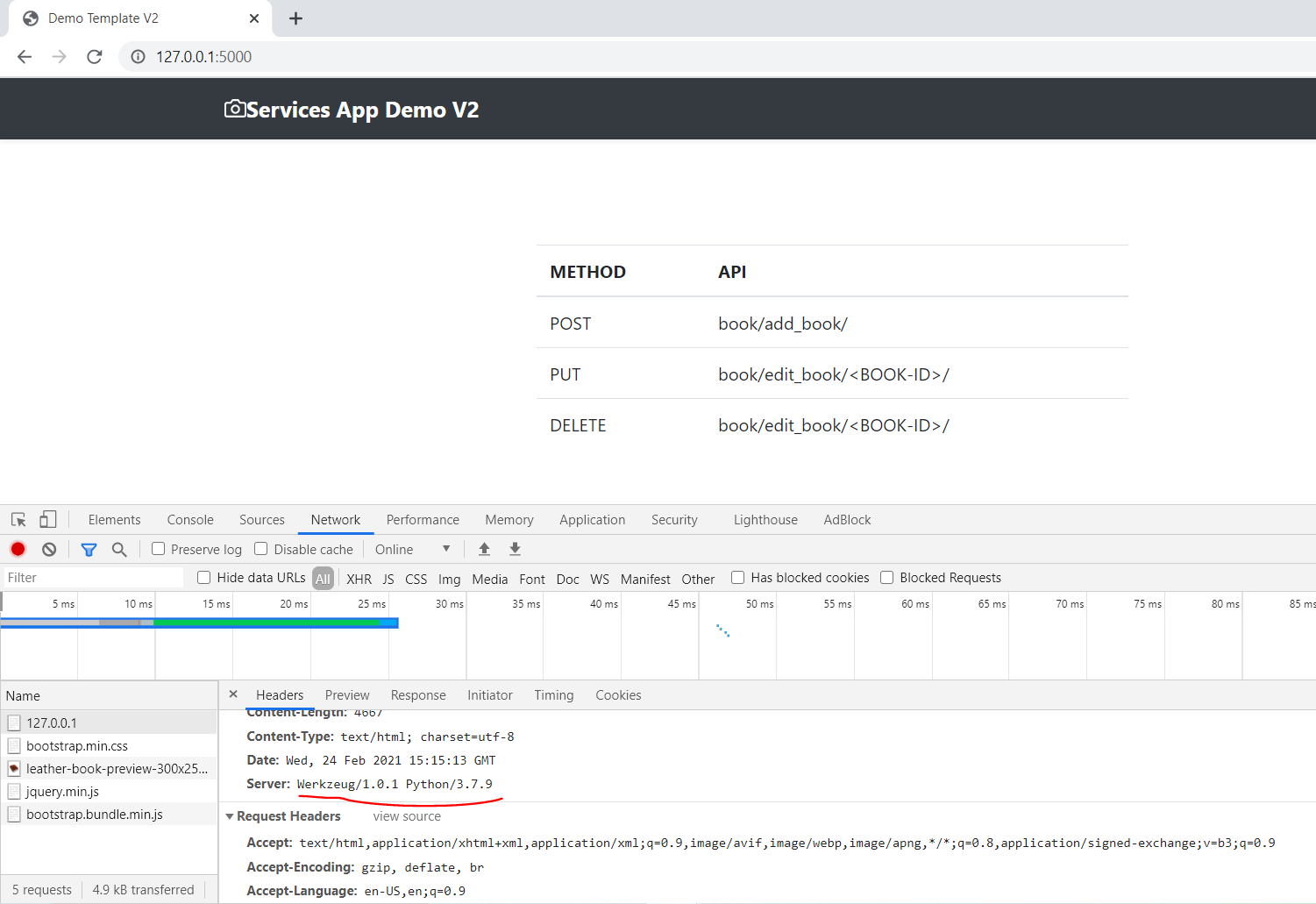
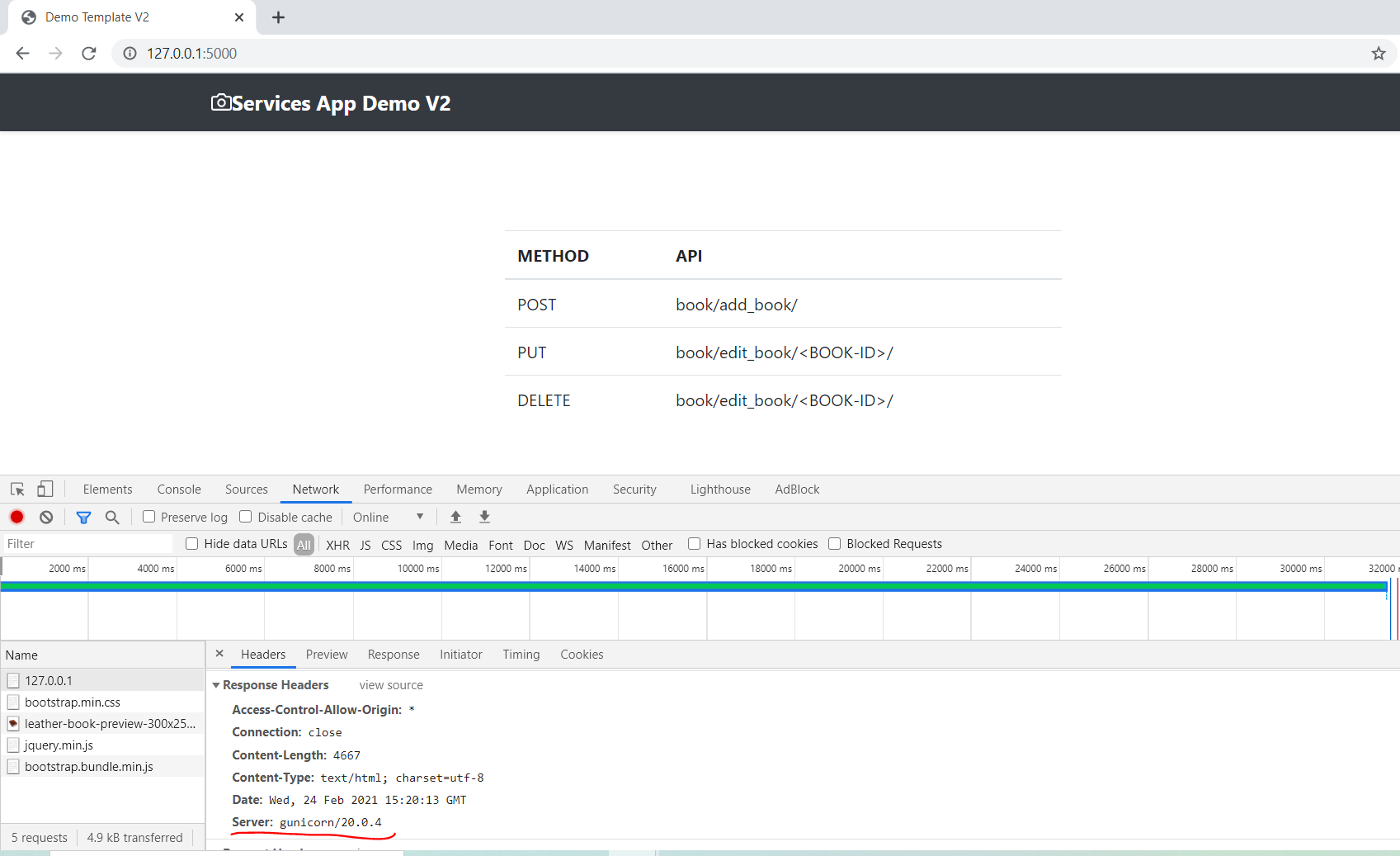
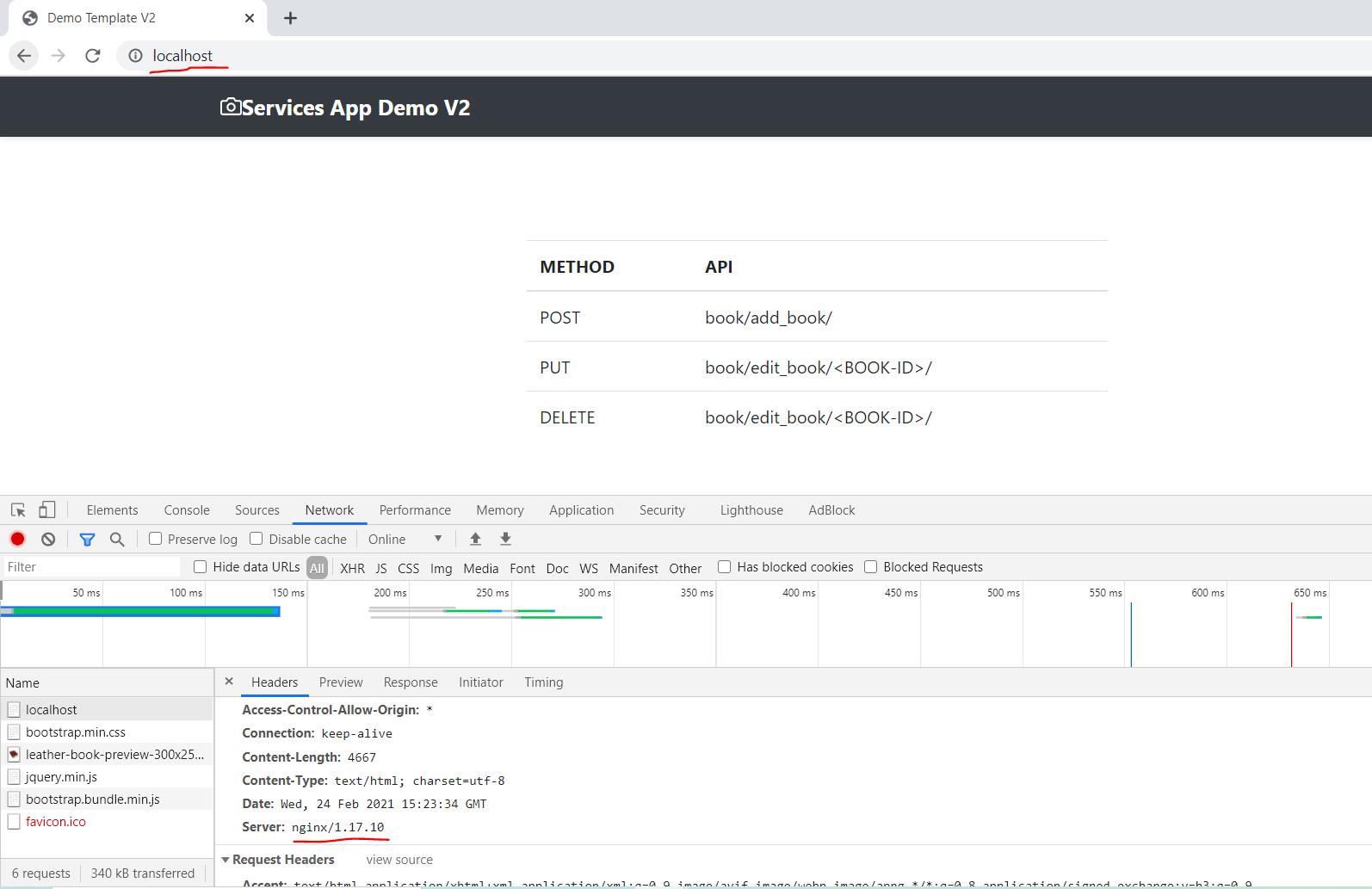
**Service Mesh POC**

**1: Default Inbuild Flask Server:**



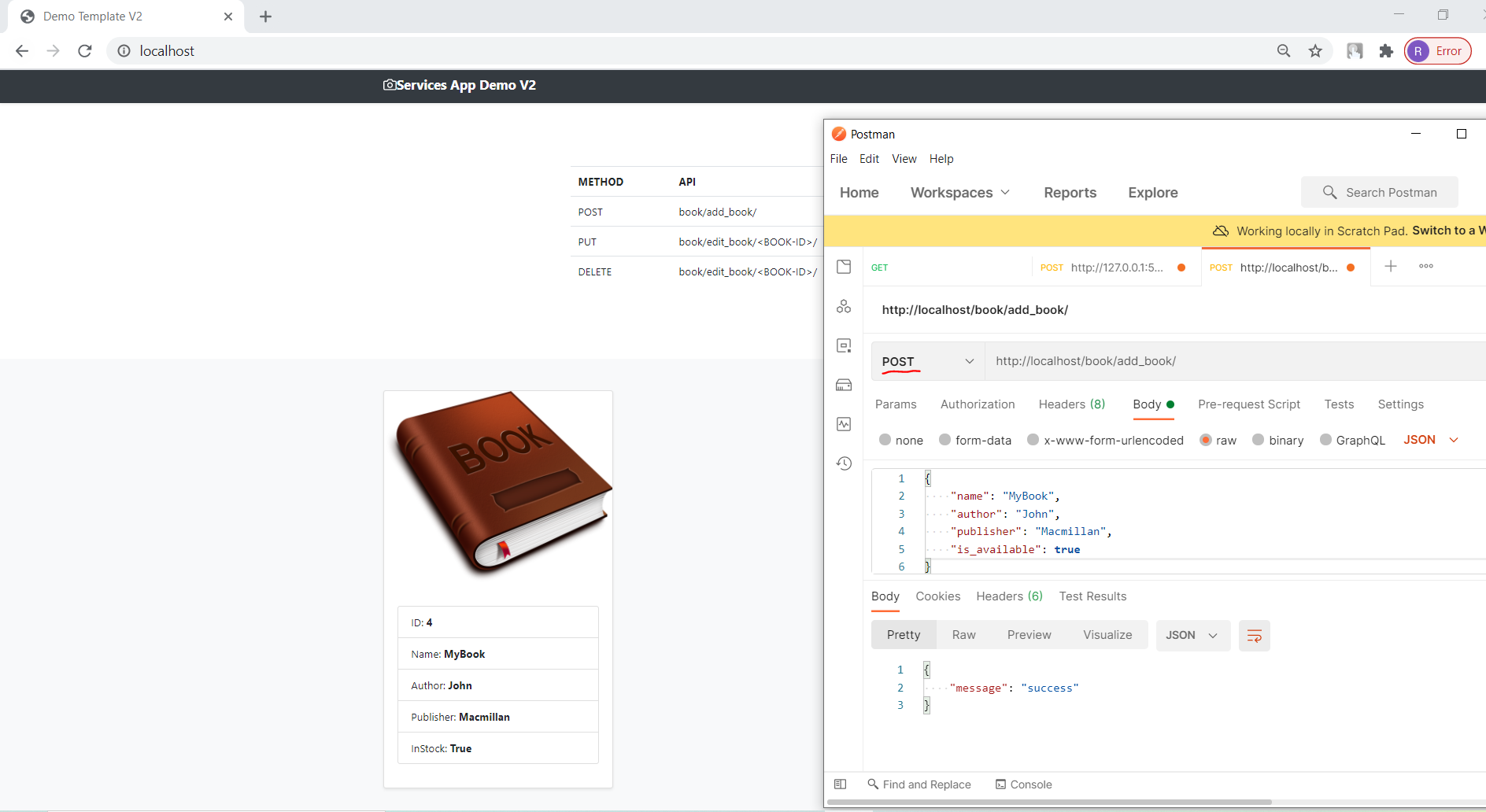
**2: Gunicorn Server:** 

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

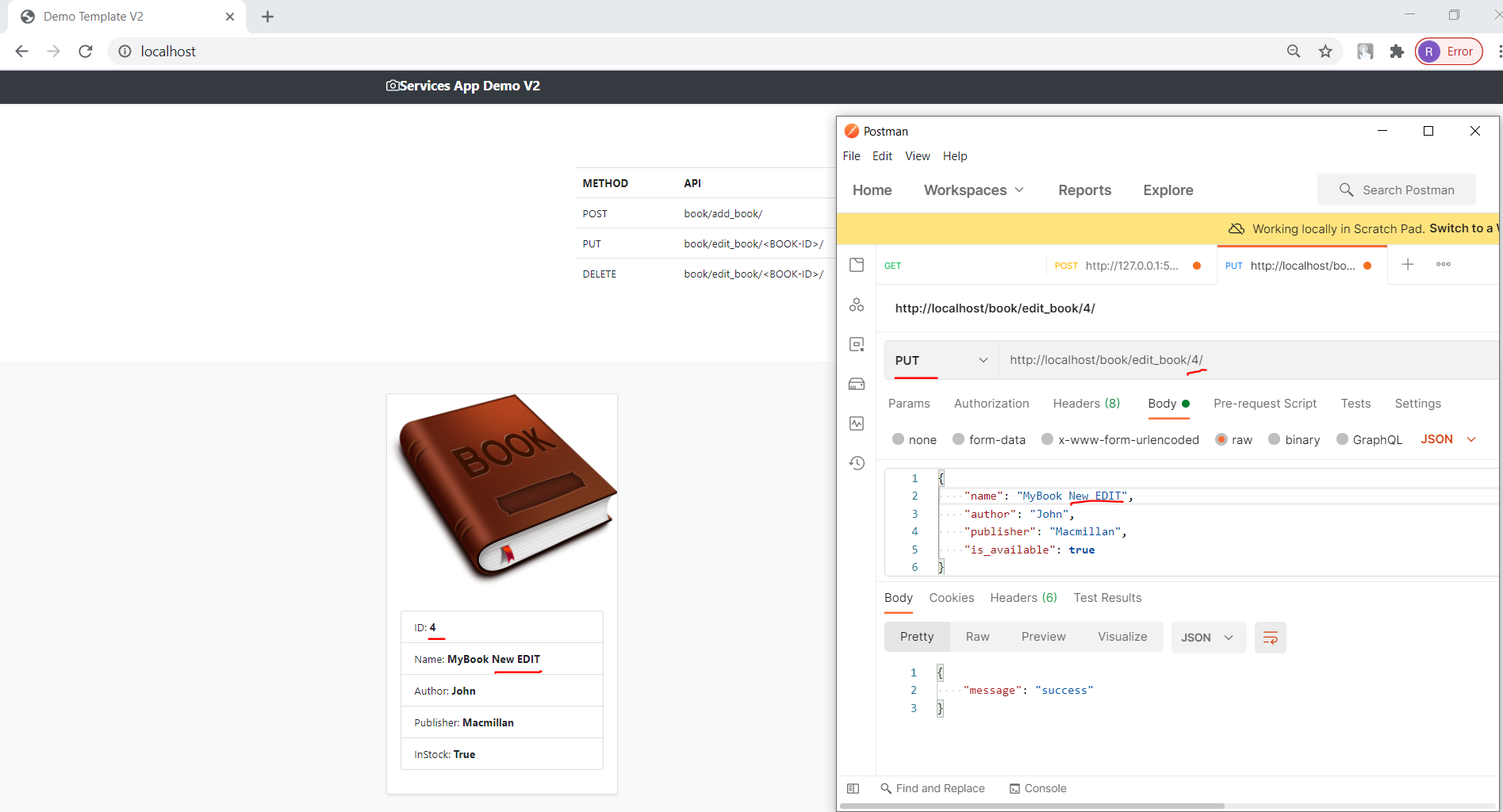


**REST Operations:**

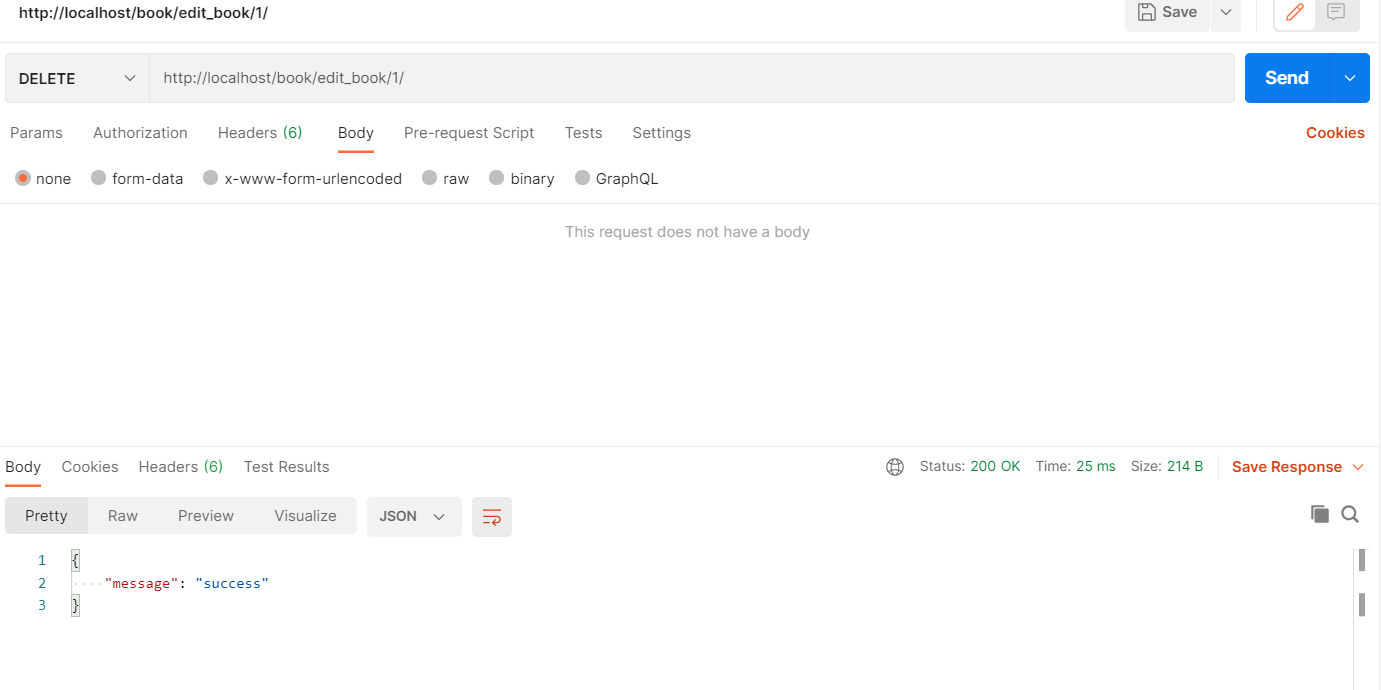
**1: POST:**

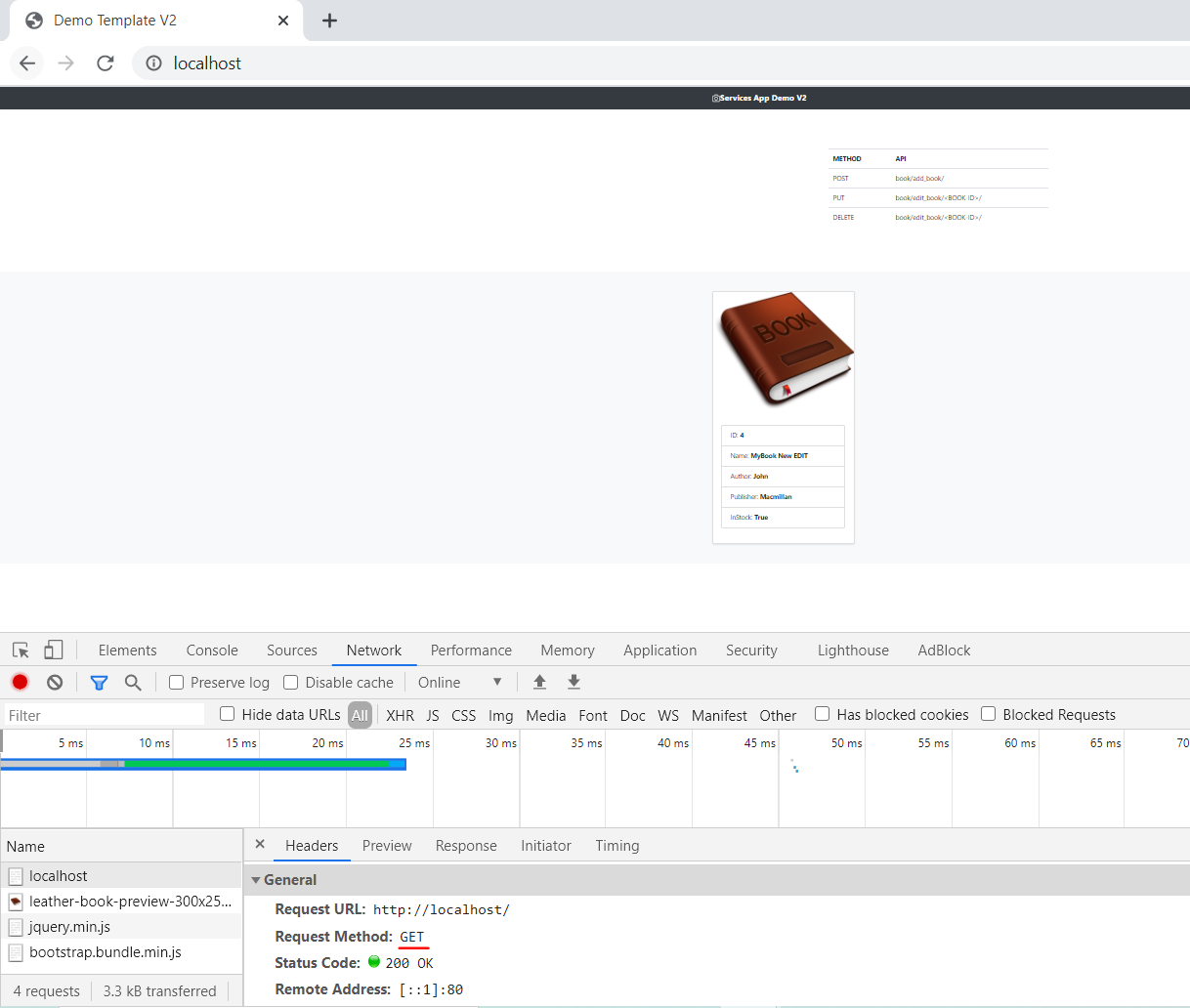


**2: PUT:**



**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;

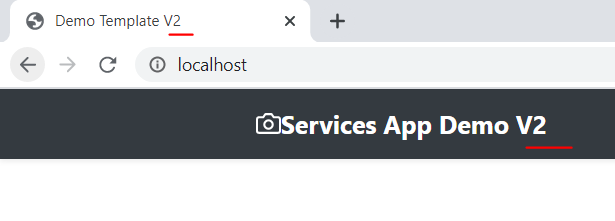
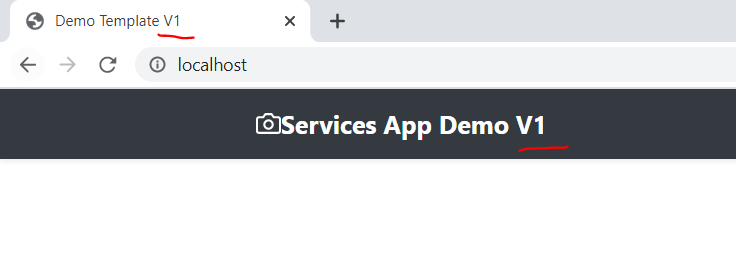
        proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

        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;

    }

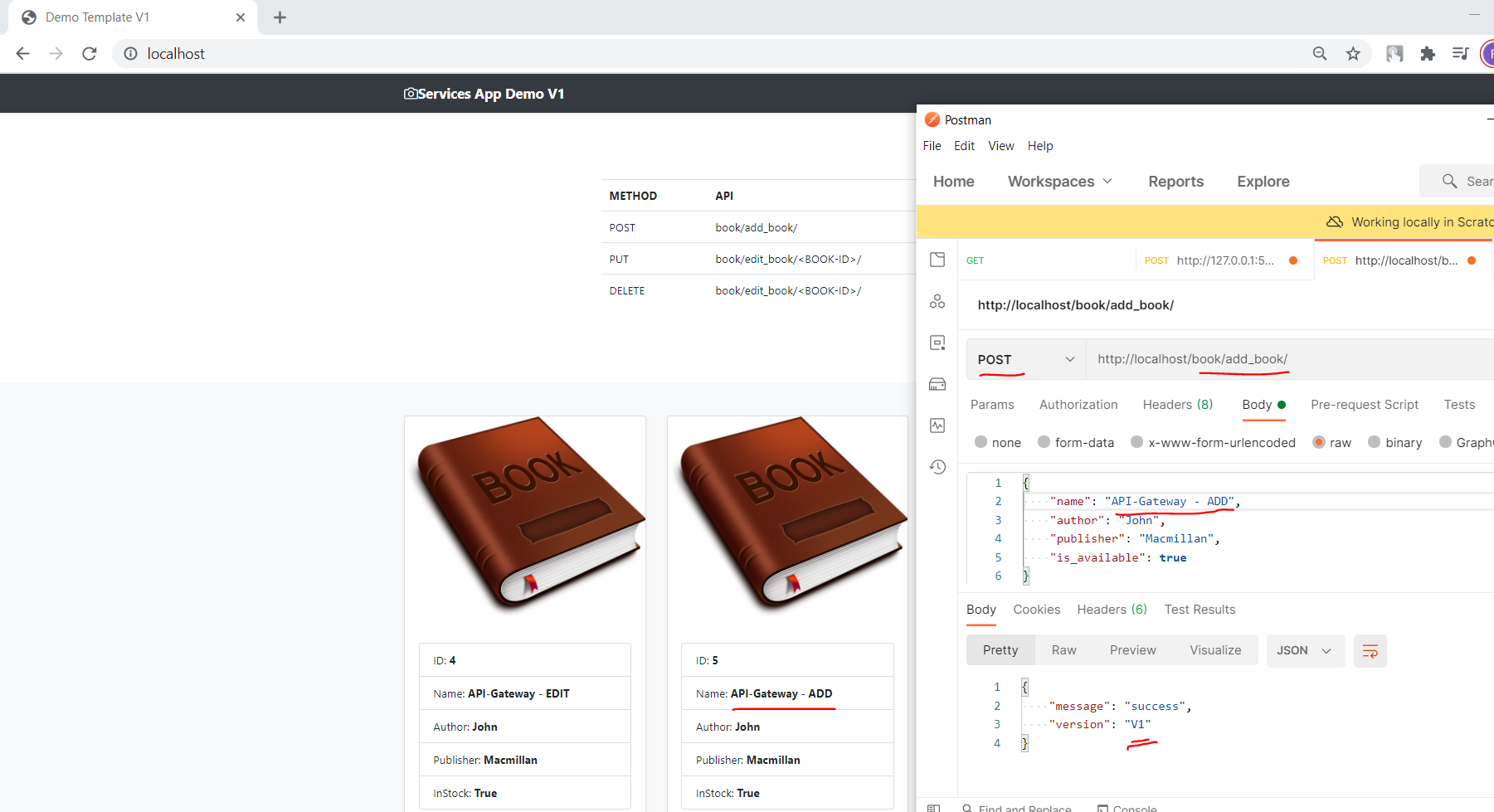
    # Edit & Delete book from DB

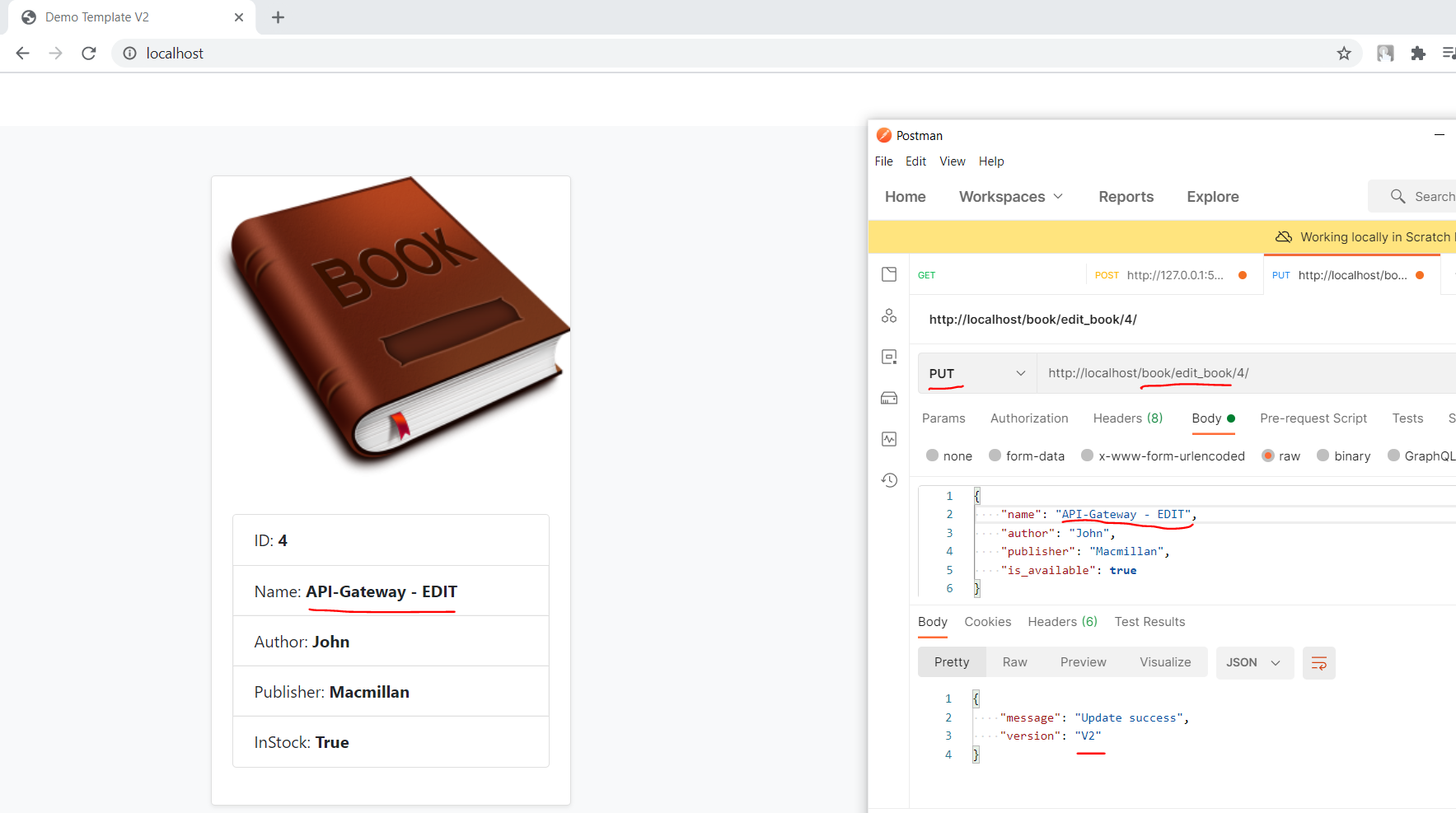
    location /book/edit\_book/ {

        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;

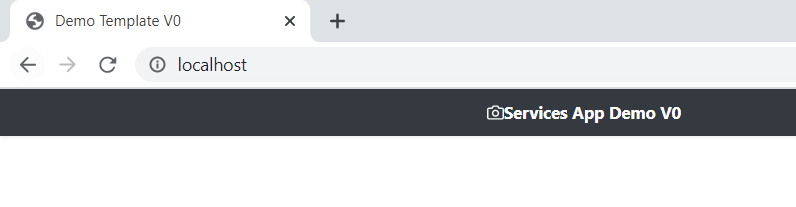
        proxy\_set\_header Host $host;

        proxy\_redirect off;

    }

}

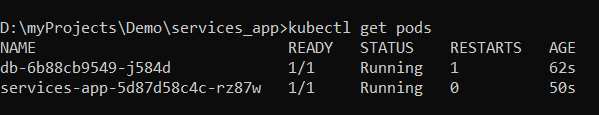
***Single Instance***



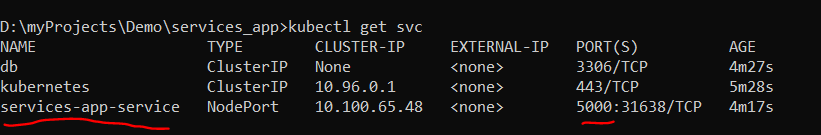
Note: Demo on Docker

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

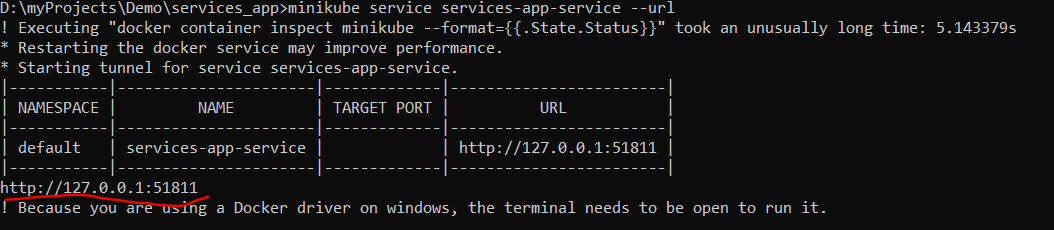
***1: Create Application and Database pods in K8s cluster***



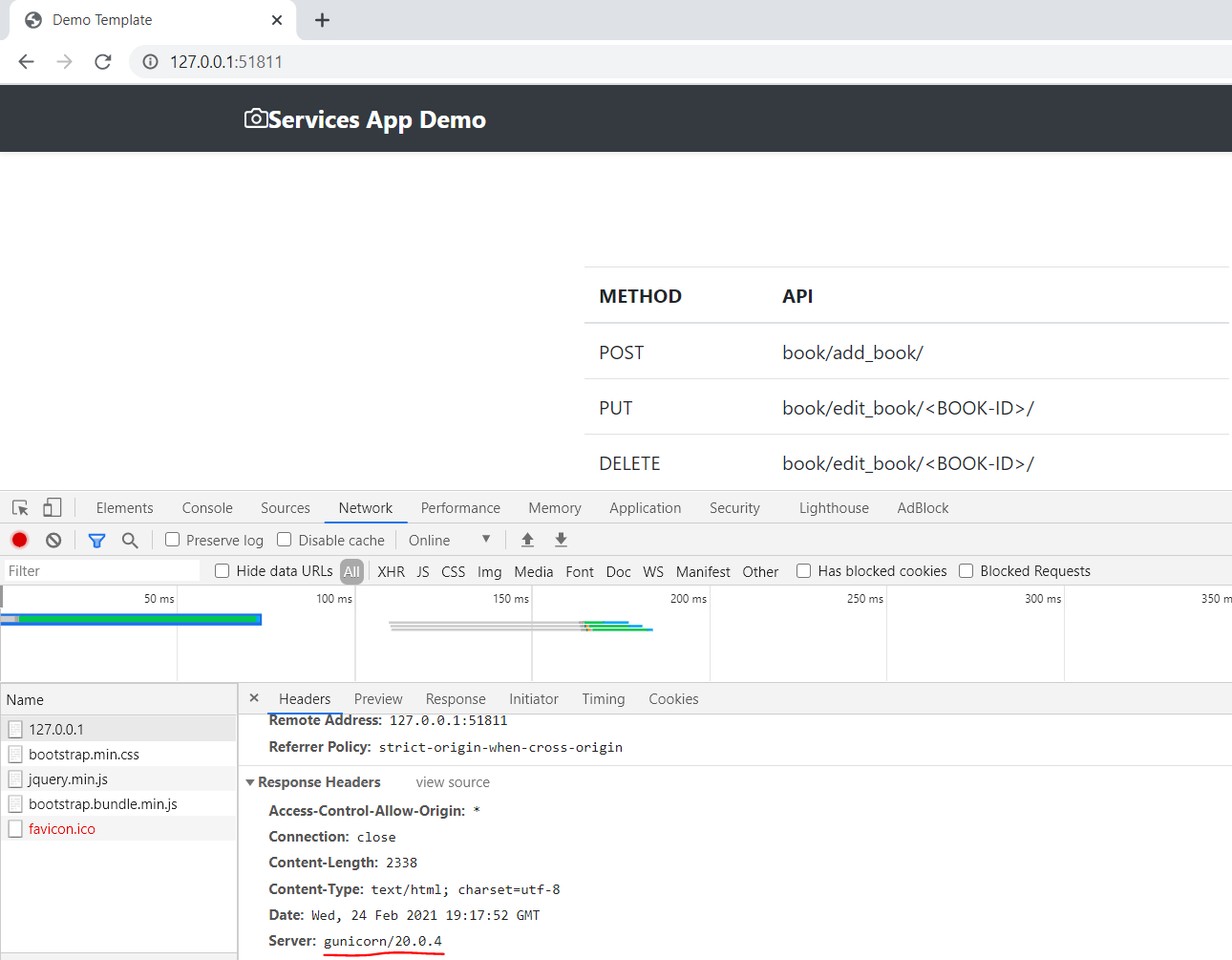
***2: Get Running Service for Application and Database***



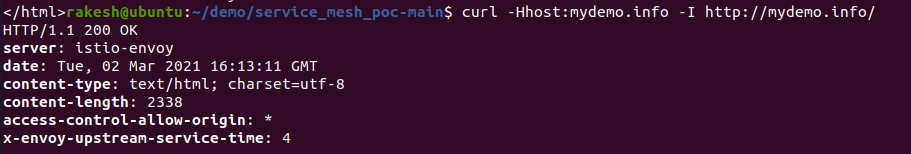
***3: Access Application inside K8s Cluster***



***Demo:***



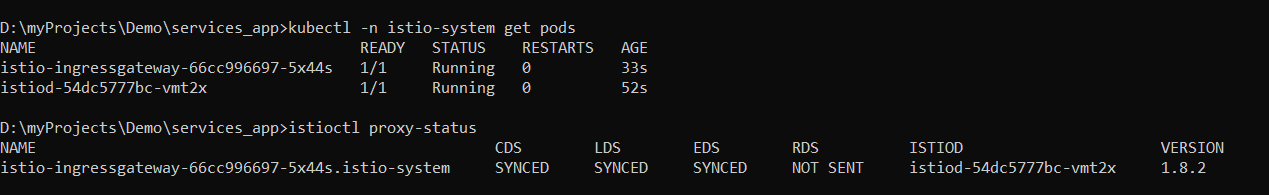
***4: Ingress controller:***



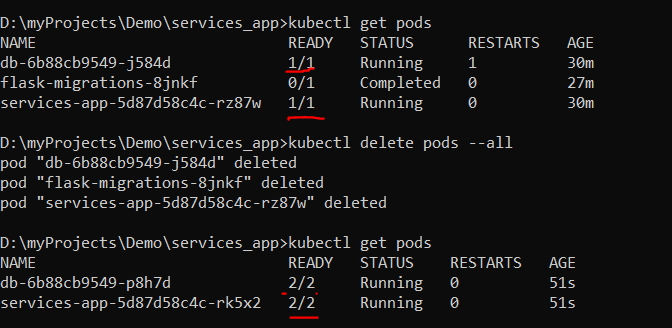
***Note: Demo on Minikub, Ubuntue***

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

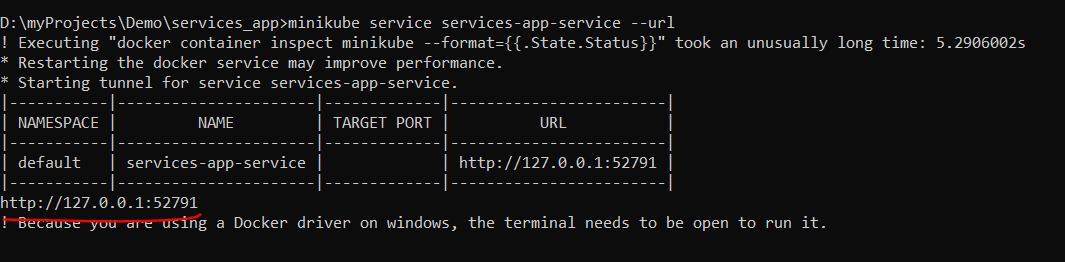
***1: Istio Pods and Service:***

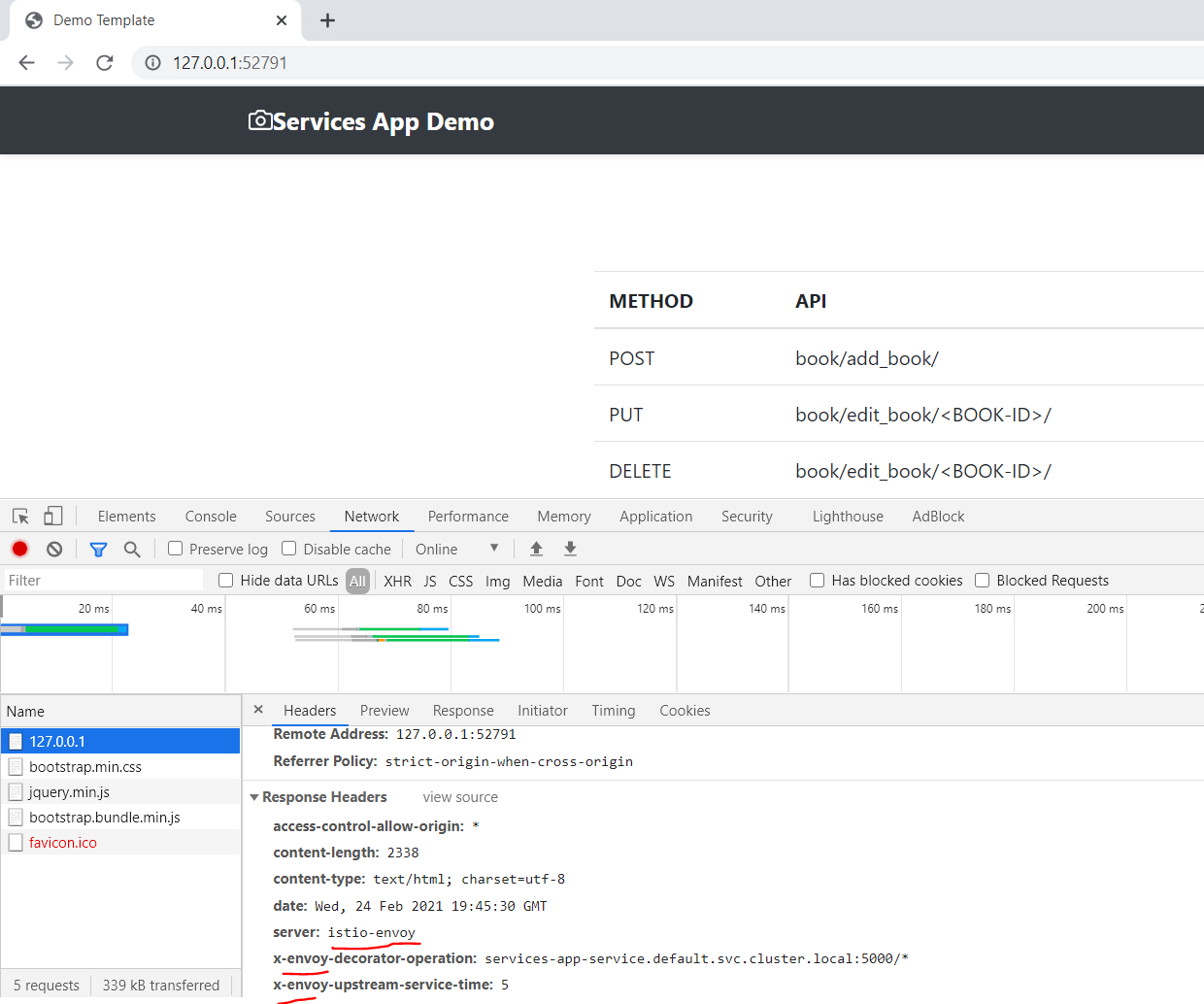


***2: Inject Istio side car (Envoy) to Application and DB pods.***

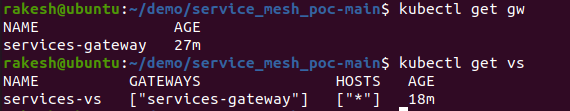


***3: Access Application inside K8s Cluster via Istio-envoy:***



***Demo:***

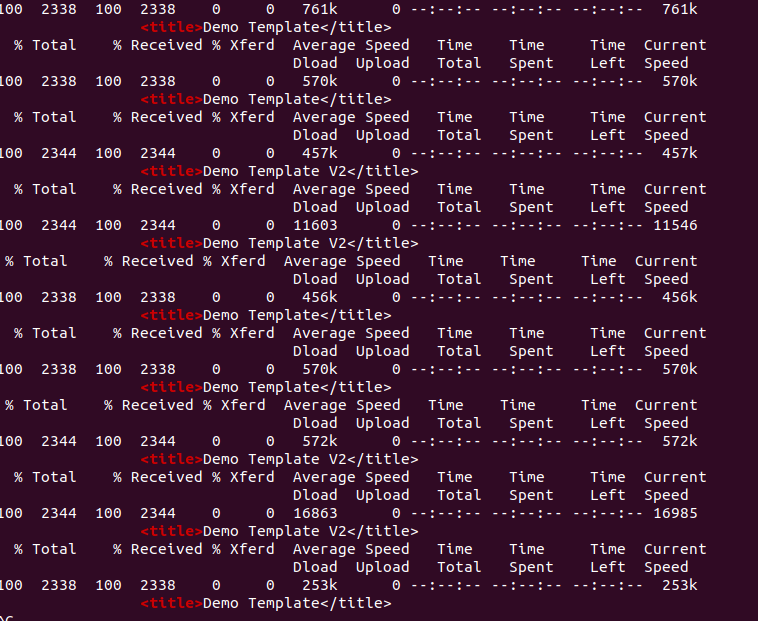
***4: Istio Gateway & VirtualService:***



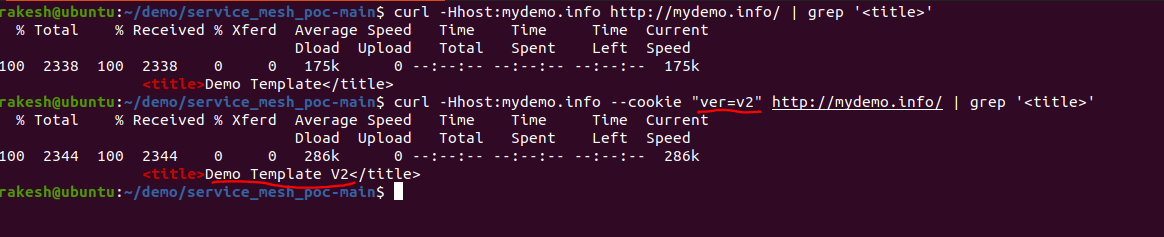
***5: Simple snippet to load test request and response***



***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)***



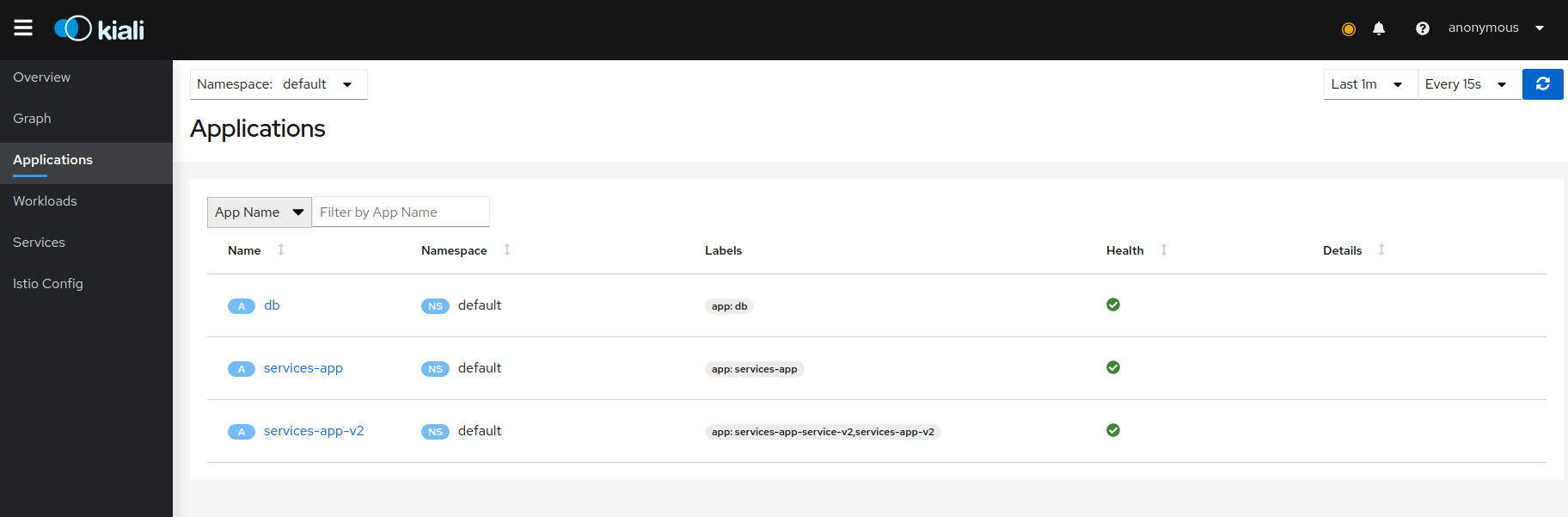
***TODO***

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

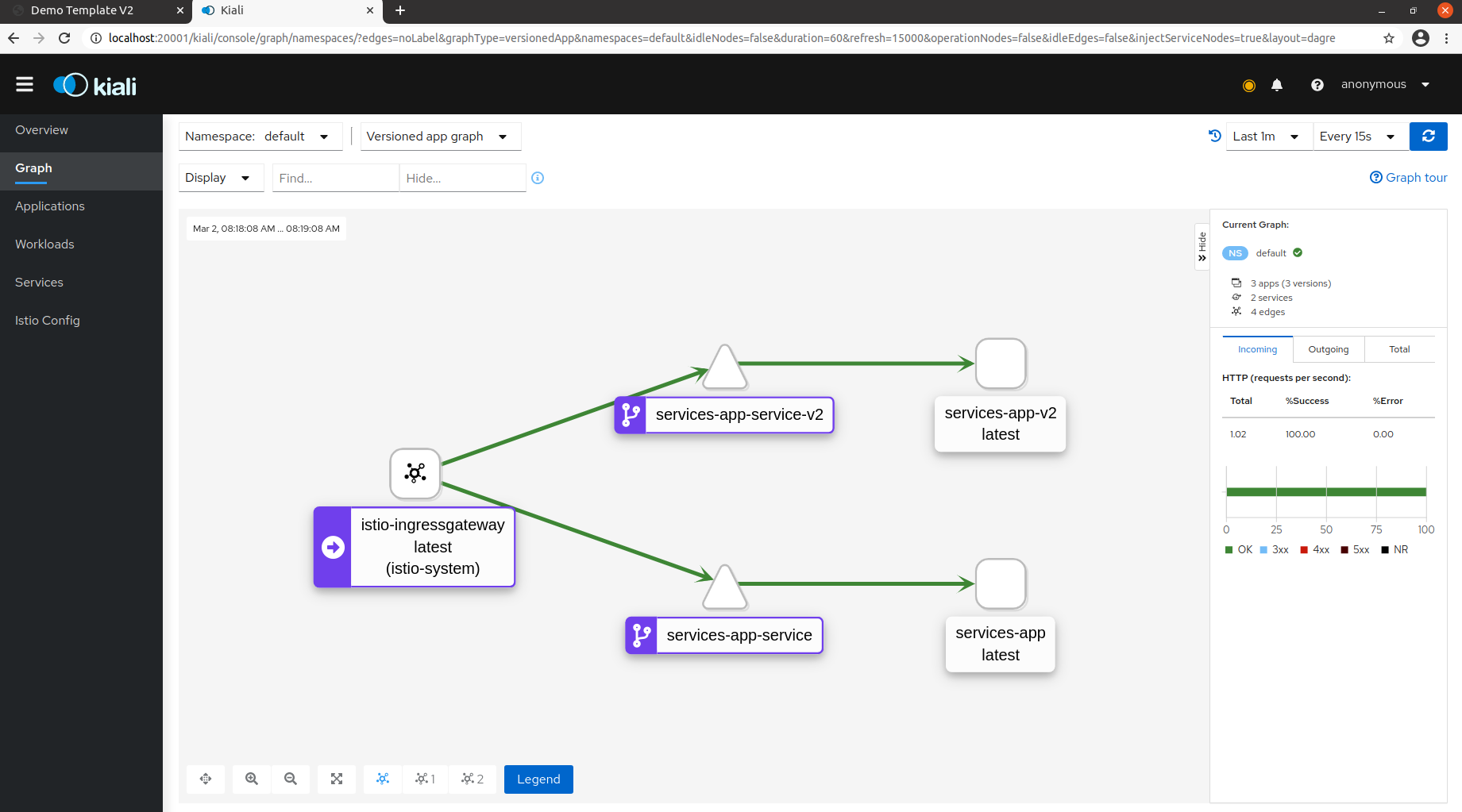
***Note: Demo on Minikube, Ubuntu***

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

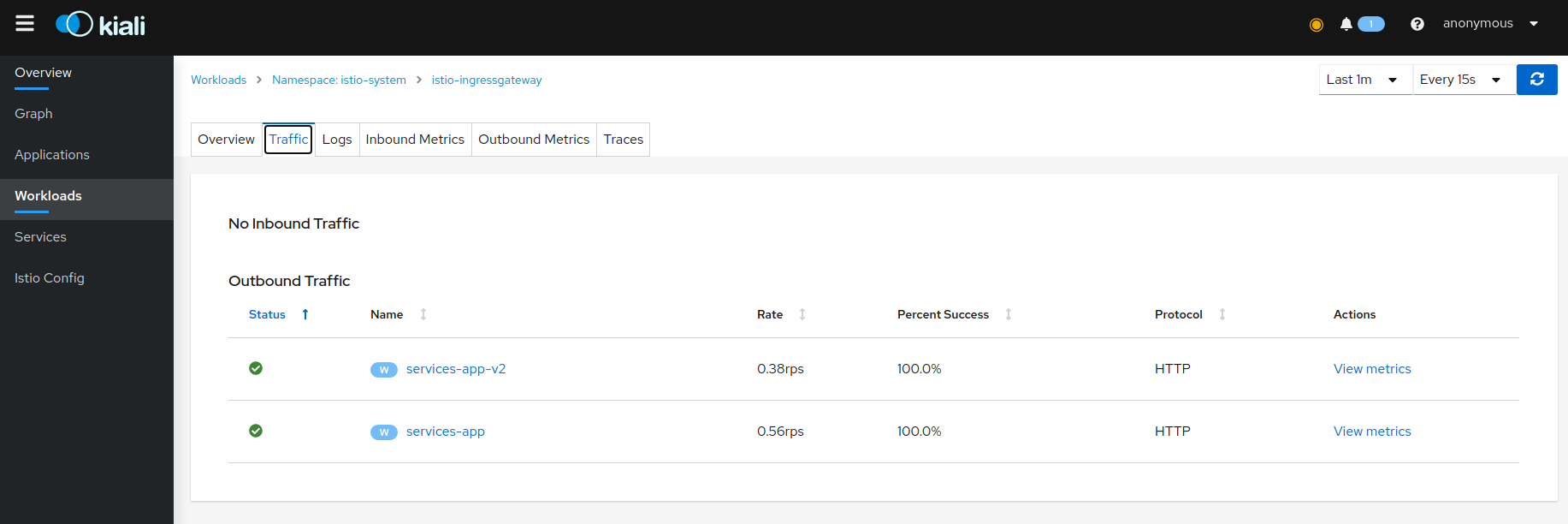
***1:*** [***Kiali***](https://kiali.io/) ***(Visualization Tool) 🡪 Real Time***



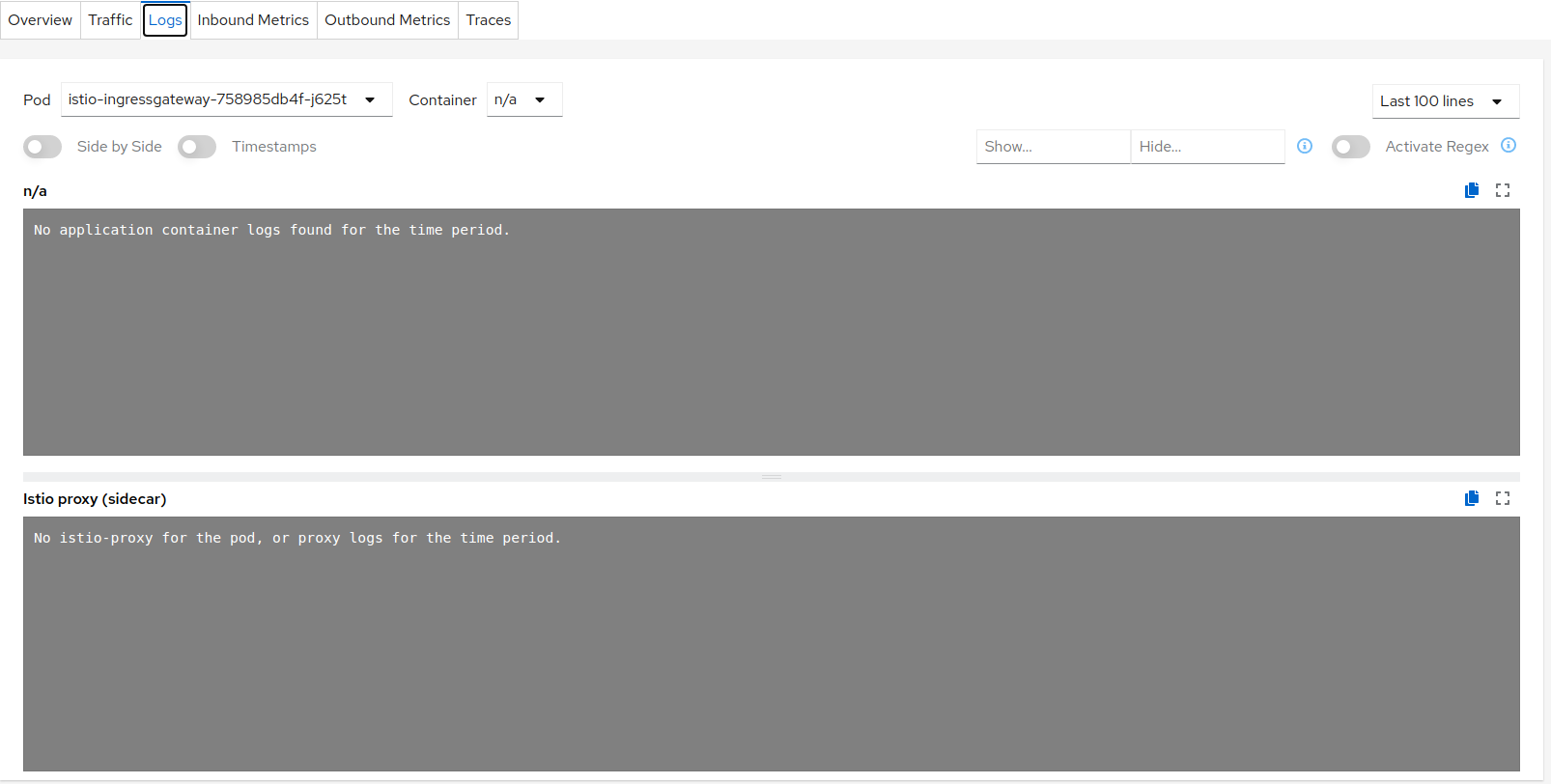
***Graph of request direction 50% to Service V1 and 50% to Service V2***



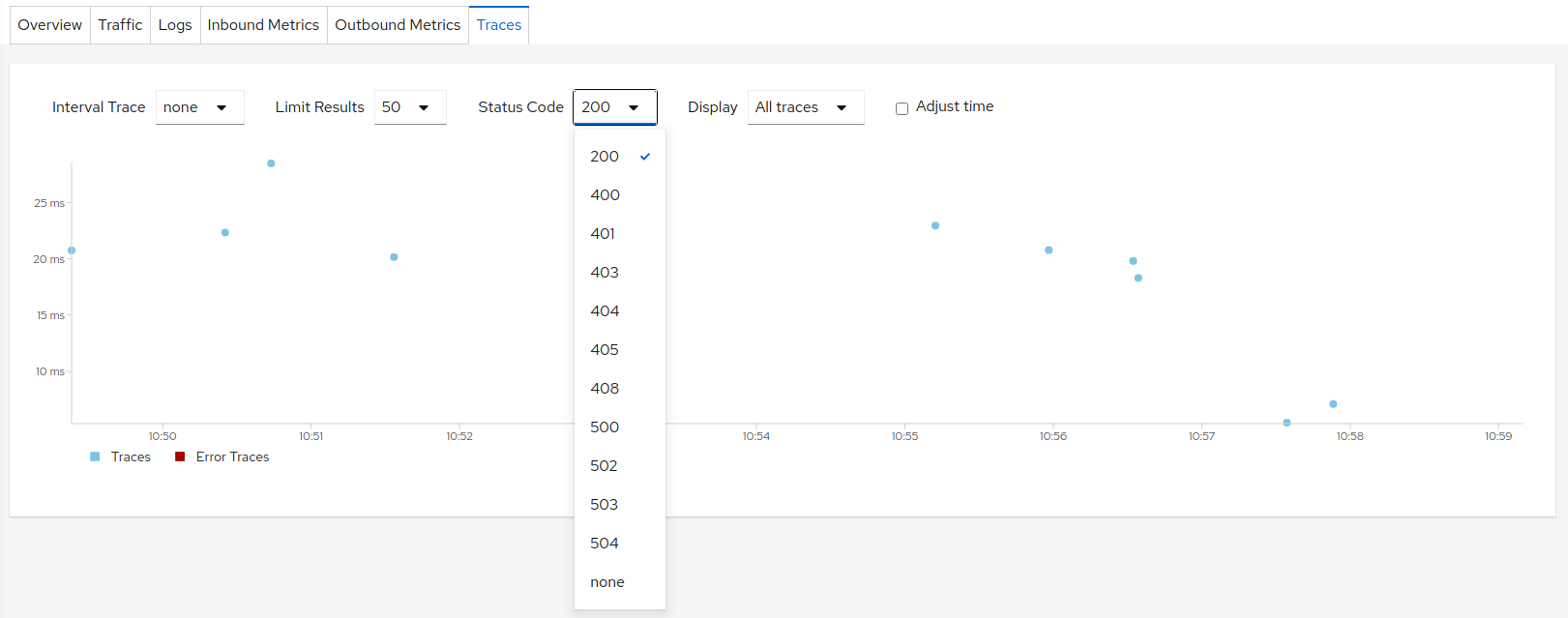
***Service Health***



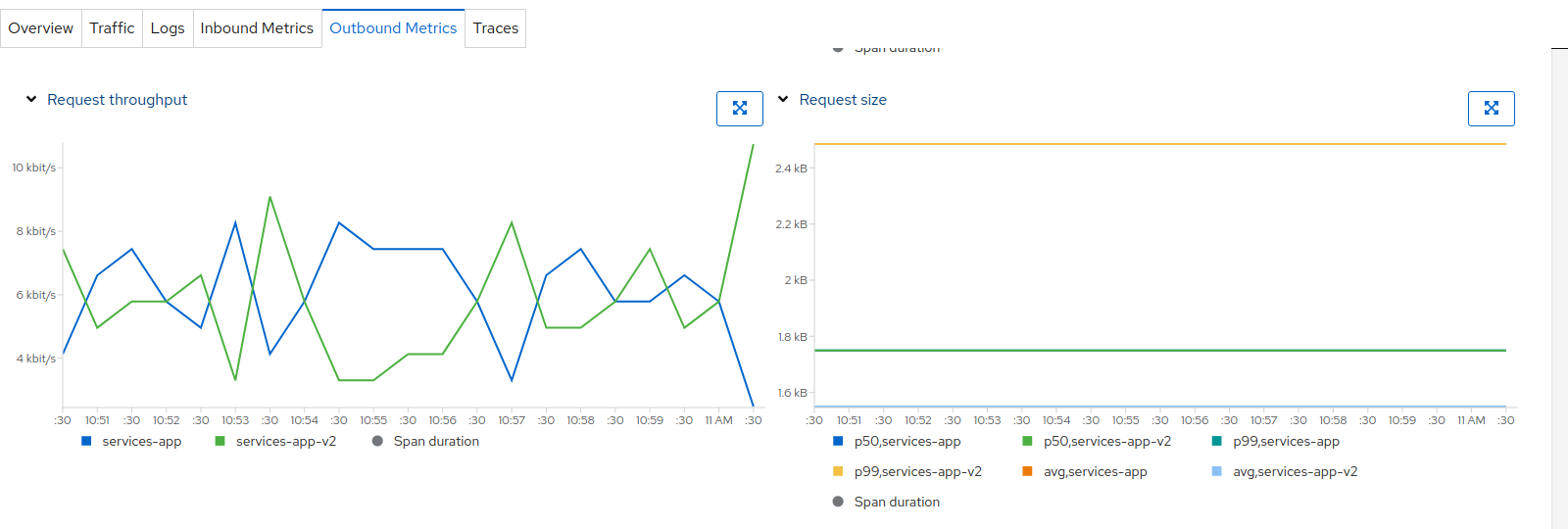
***Logs From Pods (Containers) (Not Configured Currently -TODO)***

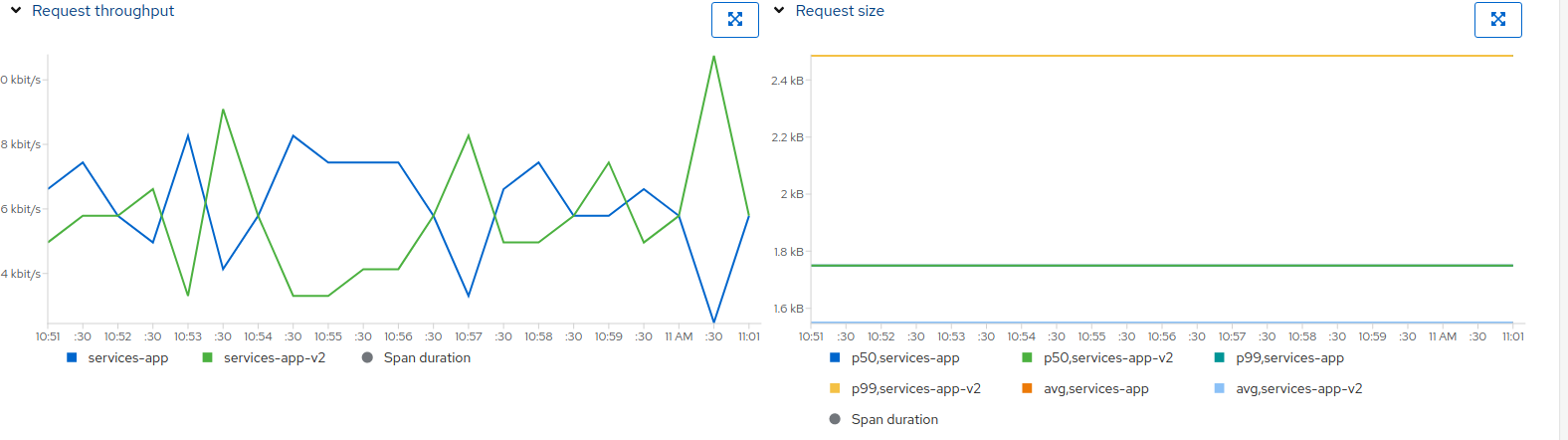


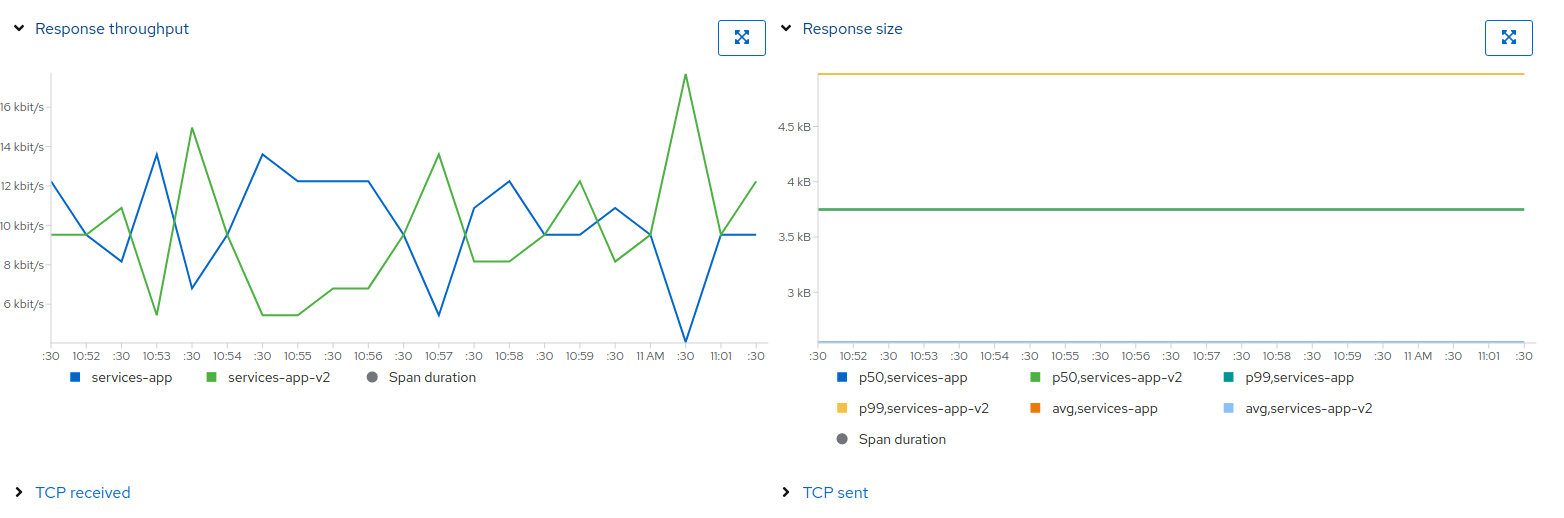
***Request Tracing (status codes and more)***



***Outbound Metrics:***



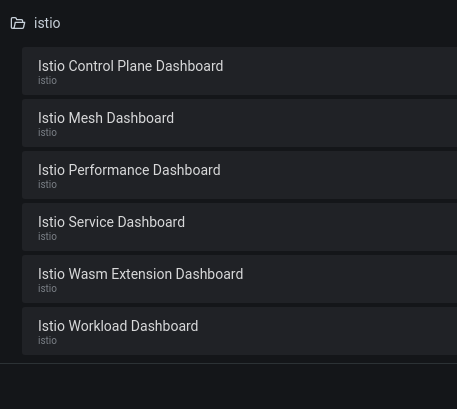




***Note: Demo on Minikub, Ubuntu***

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

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

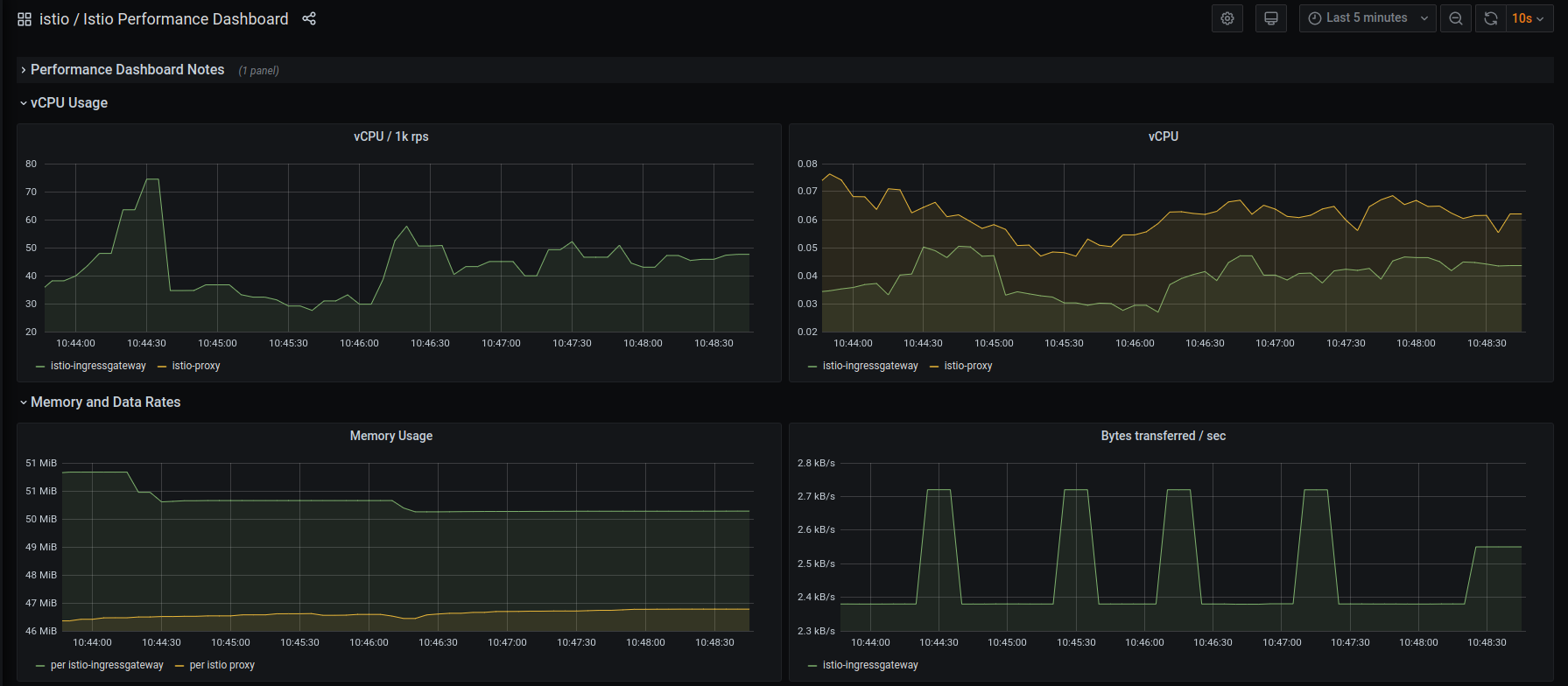


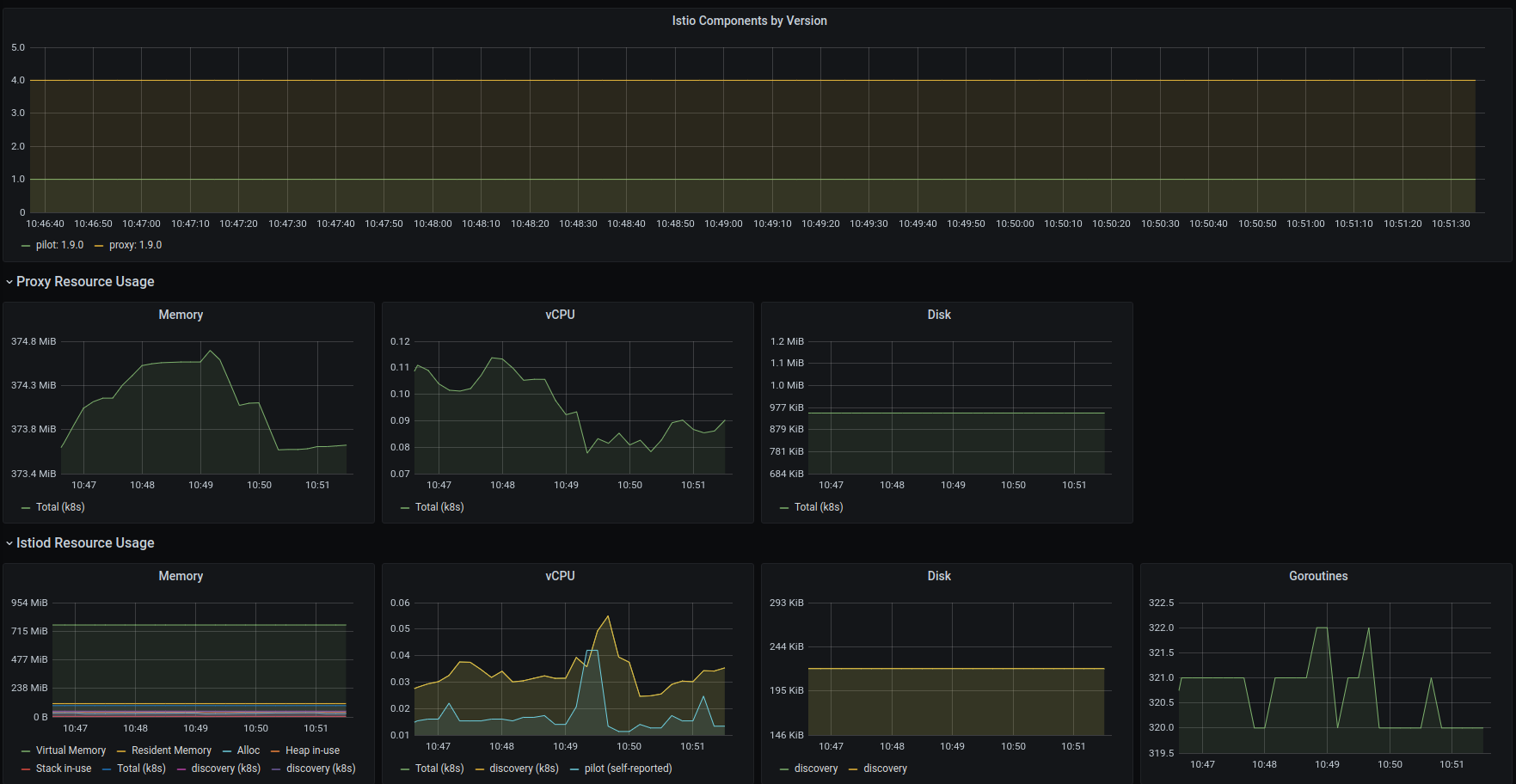
***Control Plane Dashboard***



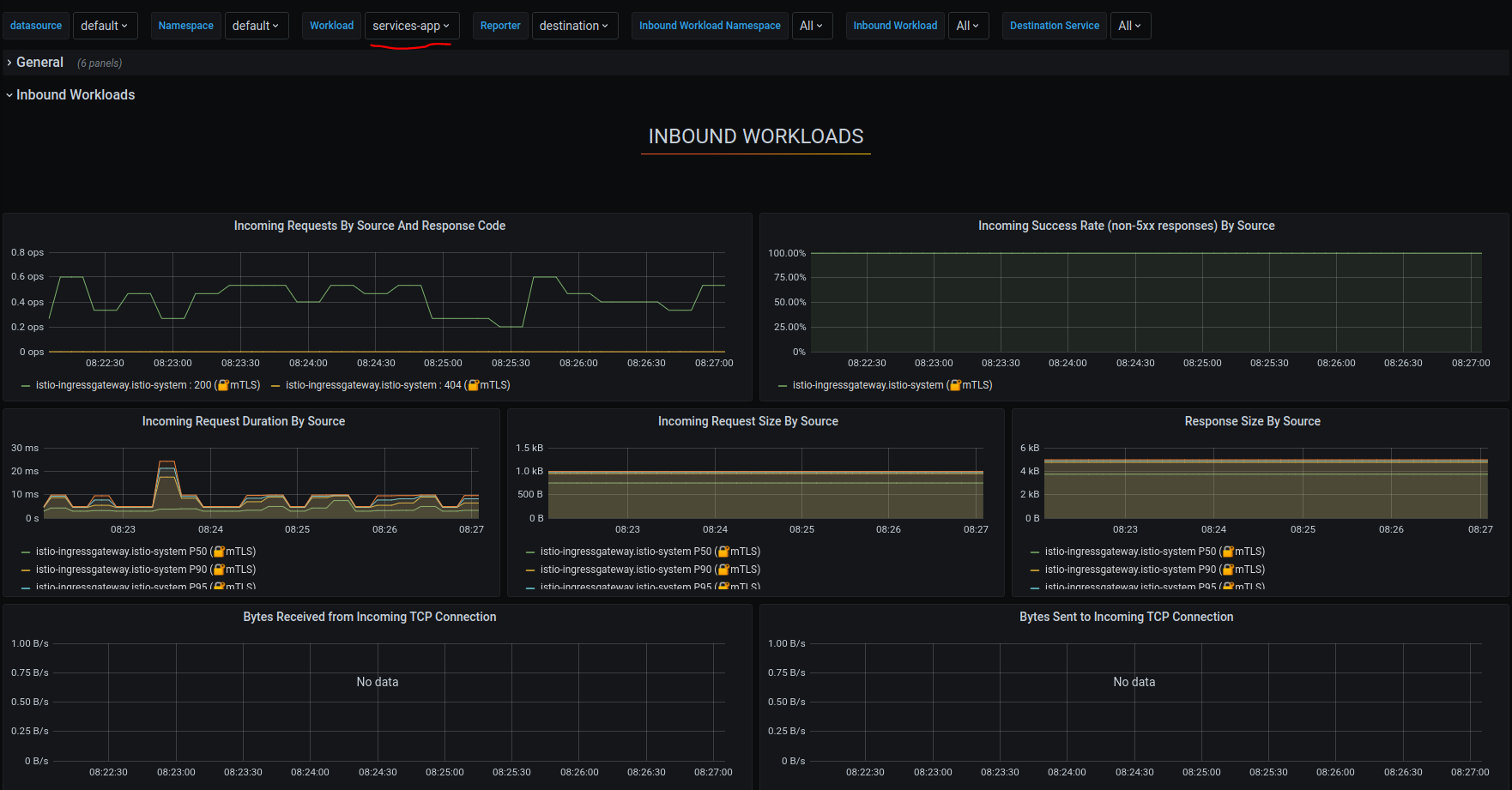


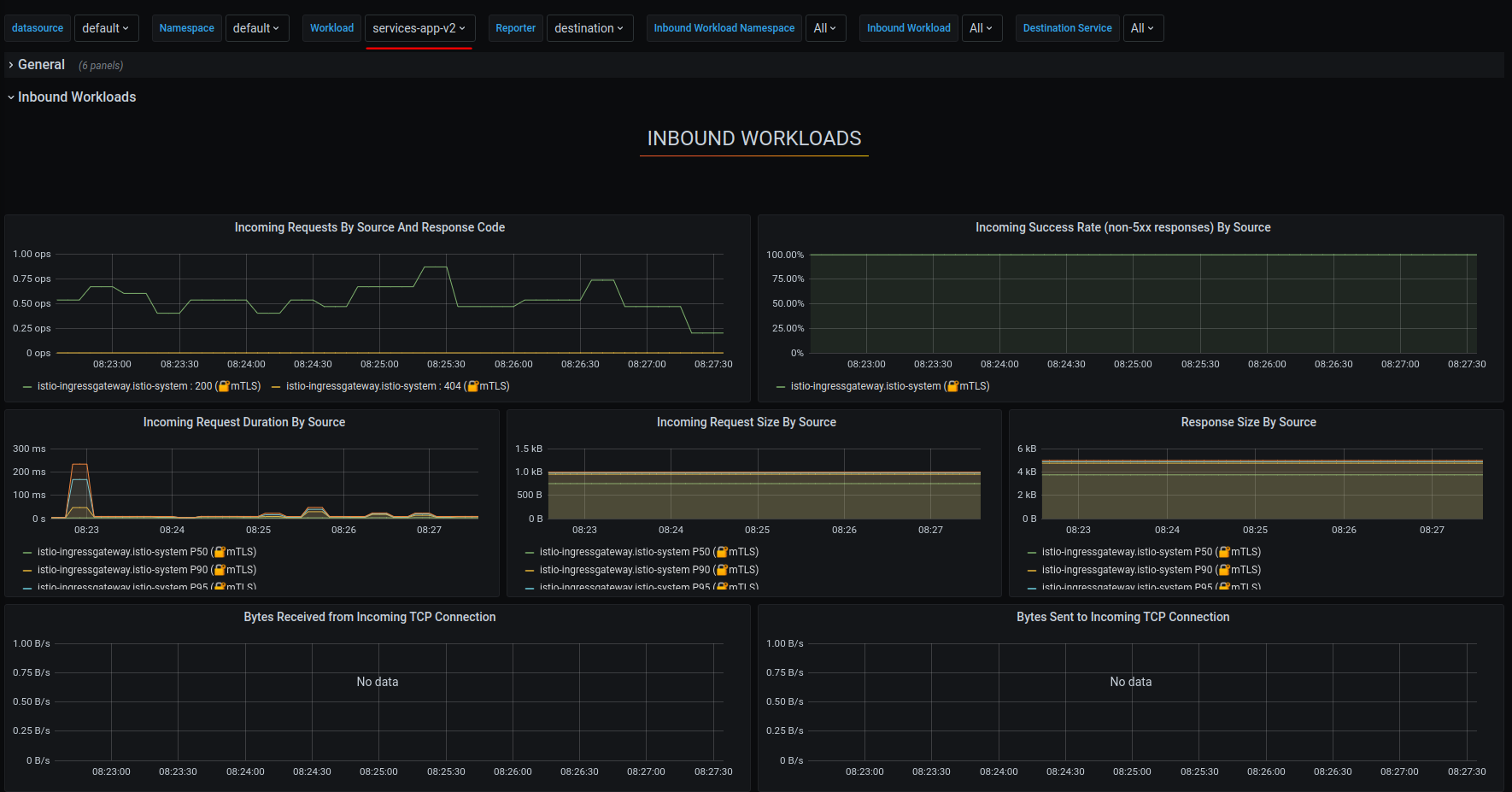
***Performance Dashboard (Metrics for CPU, Memory, IO utilization)***

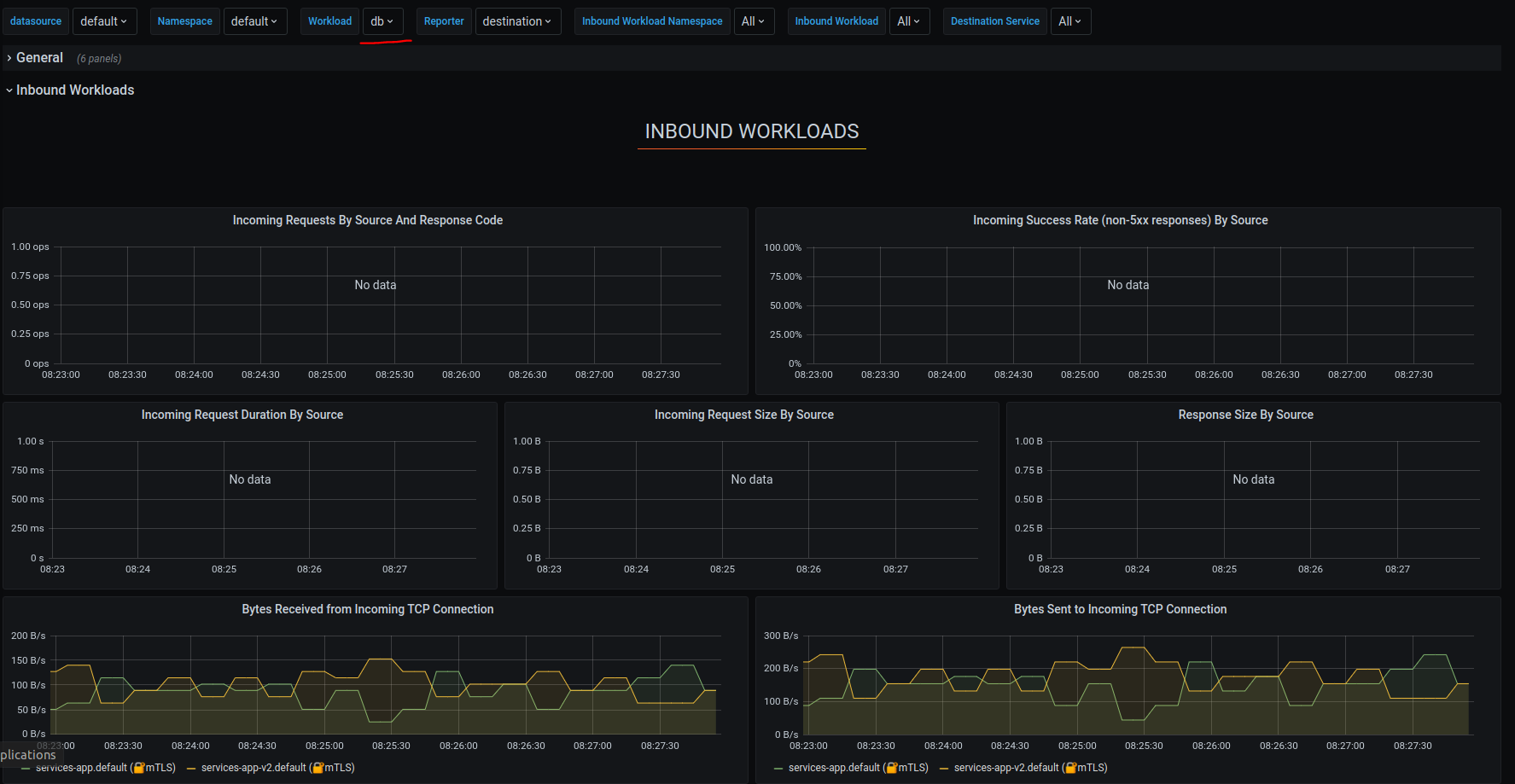




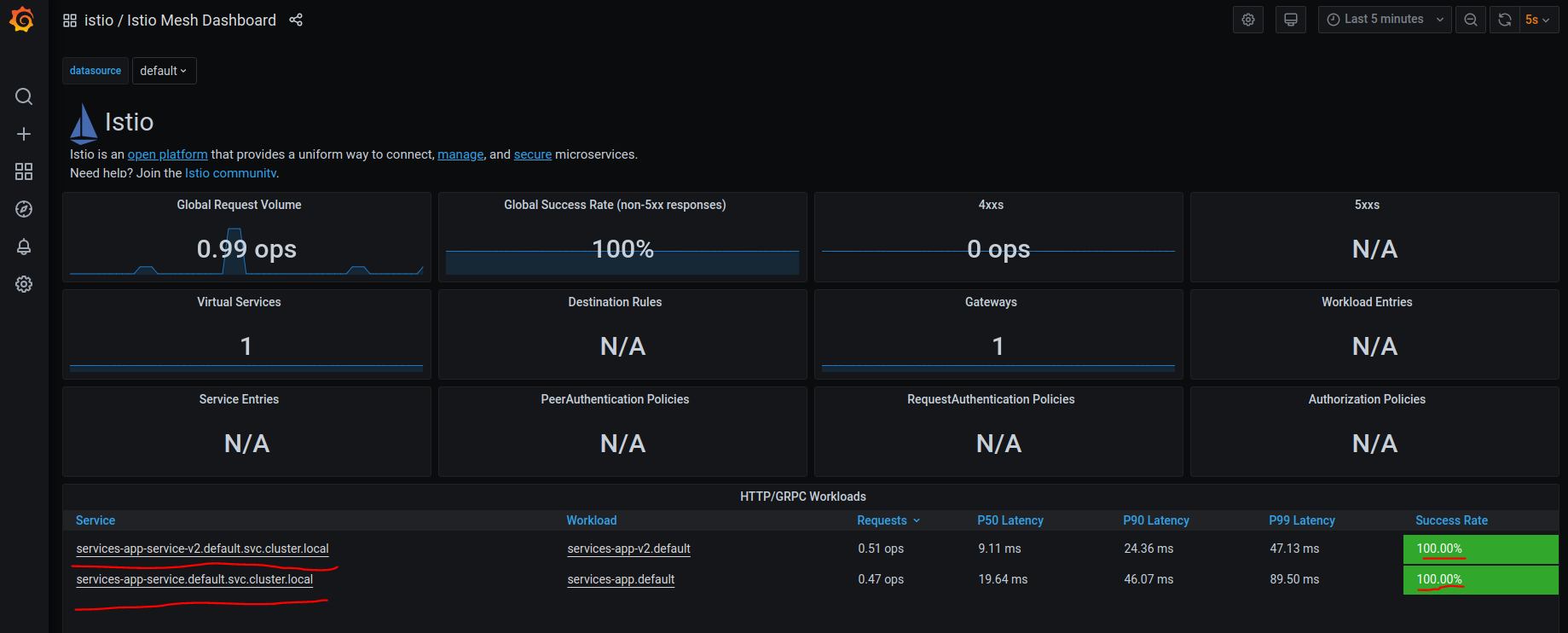
***Inbound Workload***





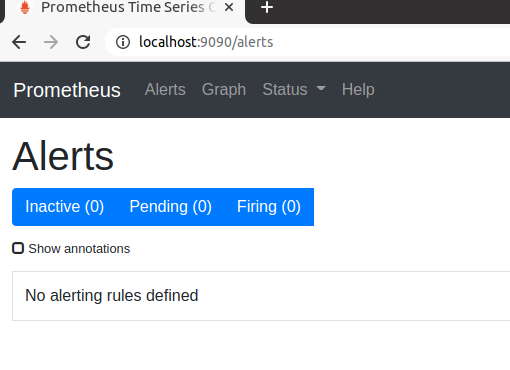


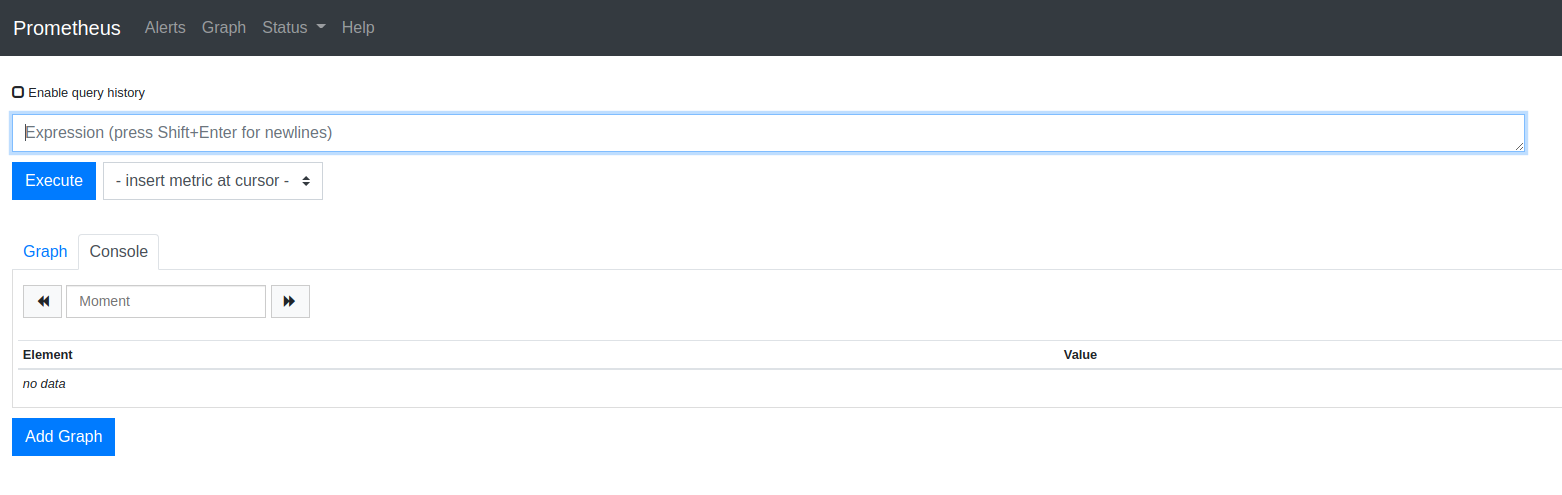
***Service Health***



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

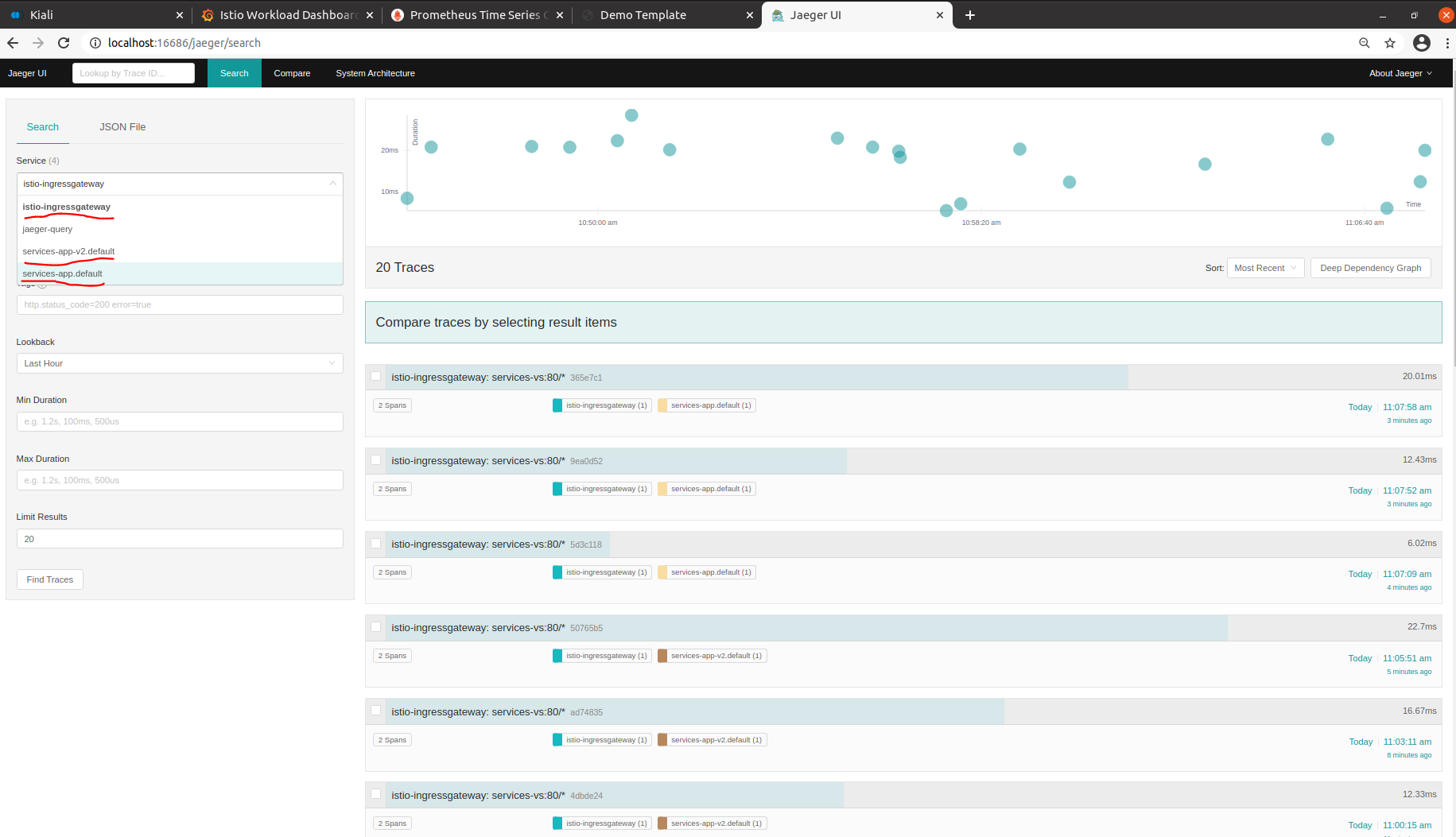
***3:*** [***Prometheus***](https://prometheus.io/) ***(Monitoring): 🡪 (TODO:Explore Options, Alerts, Monitoring)***





***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***