



Pramod Shehan

Follow

Jan 11 · 6 min read · Listen



Save



# Istio Traffic Shifting

## What is Istio

- Istio is an open source service mesh that layers transparently onto existing distributed applications.
- Istio is the path to load balancing, service-to-service authentication, and monitoring — with few or no service code changes.

## Deployment and Traffic shifting

- Traffic shifting makes it possible to gradually migrate traffic from one version of a micro service to another version. This usually happens when migrating from an older version of an app to a newer one. After doing some feature enhancement, first we should need to send small amount of traffic to the new version of service in the initial stage. After that gradually we can increase traffic percentage using service mesh.

## Prerequisite

- First we need to start **minikube**.

```
minikube start
```

```
zsh: command not found: minikube
pramodshehan@Pramods-MacBook-Pro ~ % minikube start
minikube v1.28.0 on Darwin 13.0.1 (arm64)
Using the docker driver based on existing profile
Starting control plane minikube in cluster minikube
Pulling base image ...
Restarting existing docker container for "minikube" ...
Preparing Kubernetes v1.25.3 on Docker 20.10.20 ...
Verifying Kubernetes components...
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
  ▪ Using image docker.io/kubernetes/metrics-scraper:v1.0.8
  ▪ Using image docker.io/kubernetes/dashboard:v2.7.0
💡 Some dashboard features require the metrics-server addon. To enable all features please run:

    minikube addons enable metrics-server

🌟 Enabled addons: storage-provisioner, default-storageclass, dashboard
🏁 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
pramodshehan@Pramods-MacBook-Pro ~ %
```

- Install Istio and **istioctl**.
- Here I am using simple spring boot rest api.

a) This is my old version.

```
1 package com.pramod.myapp;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RestController;
6
7 @RestController
8 @RequestMapping("/api/v1")
9 public class HelloController {
10
11     @GetMapping("hello")
12     public String getHello() {
13         return "Hello World, This is version 1!";
14     }
15 }
```

HelloController.java hosted with ❤ by GitHub

[view raw](#)

b) This is my new version file.

```
1 package com.pramod.myapp;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RestController;
6
7 @RestController
8 @RequestMapping("/api/v1")
9 public class HelloController {
10
11     @GetMapping("hello")
12     public String getHello() {
13         return "Hello World, This is version 2!";
14     }
15 }
```

HelloController.java hosted with ❤ by GitHub

[view raw](#)

- First we should need to create two docker images for old version and new version. Here I am using local docker images. We can push docker images into docker hub and we can pull those images from the docker hub or registry when deploying in Kubernetes.
- Provides instructions to point your terminal's docker-cli to minikube docker-env.

```
eval $(minikube docker-env)
```

- Build the jar file for each version and build the docker image.([how to dockerize SpringBoot application](#))
- Here we have 1.0 and 2.0 myapp docker images.

```
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
myapp	2.0	278bfe5018c2	22 seconds ago	520MB
myapp	1.0	2d0bcf24d286	About a minute ago	520MB

- We can use below mentioned **kubectl** command to add a namespace label to instruct Istio to automatically inject Envoy sidecar proxies.

```
kubectl label namespace default istio-injection=enabled
```

```
virtualservice.networking.istio.io/myapp configured
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl label namespace default istio-injection=enabled
namespace/default not labeled
```

- After enabled Istio in minikube, Istio-proxy(Envoy proxy) called docker container is running in each and every pods.

## Deployment

- Now we can do all the deployments.

1. First we should deploy myapp service and two pods for v1 and v2.

Here **imagePullPolicy** is **Never** because I am using local docker images. We have two deployment definition for myapp-v1 and myapp-v2. myapp:1.0 is defined as v1 and myapp:2.0 is defined as v2.

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: myapp
5    labels:
6      app: myapp
7      service: myapp
8  spec:
9    ports:
10     - port: 8080
11       name: http
12     selector:
13       app: myapp
14  ---
15  apiVersion: v1
16  kind: ServiceAccount
17  metadata:
18    name: demo-myapp
19    labels:
20      account: myapp
21  ---
22  apiVersion: apps/v1
23  kind: Deployment
24  metadata:
25    name: myapp-v1
26    labels:
27      app: myapp
28      version: v1
29  spec:
30    replicas: 1
31    selector:
32      matchLabels:
33        app: myapp
34        version: v1
35    template:
36      metadata:
37        labels:
38          app: myapp
39          version: v1
40      spec:
41        serviceAccountName: demo-myapp
42        containers:
43          - name: myapp
44            image: myapp:1.0
45            imagePullPolicy: Never
46            ports:
47              - containerPort: 8080
48            volumeMounts:
49              - name: tmp
50                mountPath: /tmp
51            securityContext:
52              runAsUser: 1000
53            volumes:
54              - name: tmp
55                emptyDir: {}
56  ---
57  apiVersion: apps/v1
58  kind: Deployment
59  metadata:
60    name: myapp-v2
61    labels:
62      app: myapp
63      version: v2
64  spec:
65    replicas: 1
66    selector:
67      matchLabels:
```

```
68     app: myapp
69     version: v2
70   template:
71     metadata:
72       labels:
73         app: myapp
74         version: v2
75     spec:
76       serviceAccountName: demo-myapp
77       containers:
78       - name: myapp
79         image: myapp:2.0
80         imagePullPolicy: Never
81         ports:
82         - containerPort: 8080
83         volumeMounts:
84         - name: tmp
85           mountPath: /tmp
86         securityContext:
87           runAsUser: 1000
88       volumes:
89       - name: tmp
```

```
kubectl apply -f myapp.yaml
```

```

zsh: command not found: kubectl
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl apply -f myapp.yaml
service/myapp created
serviceaccount/demo-myapp created
deployment.apps/myapp-v1 created
deployment.apps/myapp-v2 created
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$

```

- Check deployments , pods and services.

```

kubectl get svc
kubectl get pods
kubectl get deployments

```

```

pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1    <none>       443/TCP    4d23h
myapp        ClusterIP   10.97.58.17  <none>       8080/TCP   35m

pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
myapp-v1-6cc8bbccf6-svllg          2/2     Running   0           35m
myapp-v2-587b79c77d-nlglr          2/2     Running   0           35m

pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl get deployments
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
myapp-v1     1/1     1             1           35m
myapp-v2     1/1     1             1           35m
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$

```

- We have defined only one container in the deployment description. But two containers are running in the pod because each and every pods have its own Istio-proxy(Envoy proxy) and defined application container(myapp). After that all the communication between services, using that envoy proxy.

```

containers:
- name: myapp
  image: myapp:1.0
  imagePullPolicy: Never

```

```

pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
myapp-v1-6cc8bbccf6-svllg          2/2     Running   0           6h44m
myapp-v2-587b79c77d-nlglr          2/2     Running   0           6h44m
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$

```

```

kubectl describe pods myapp-v1-6cc8bbccf6-svllg

```

```
Containers:
  myapp:
    Container ID:   docker://1c5d77ed3c2a0b1fd764dc44c69020fd0d787a77ab9111761bf906855a912bcb
    Image:          myapp:1.0
    Image ID:       docker://sha256:2d0bcf24d28627ea6f64ab434e7fd8c5f34a871cba8f09dcdbcee5d55a97ea4b
    Port:          8080/TCP
    Host Port:     0/TCP
    State:         Running
      Started:     Wed, 11 Jan 2023 23:16:54 +0800
    Ready:         True
    Restart Count:  0
    Environment:   <none>
    Mounts:
      /tmp from tmp (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-l8bnf (ro)
  istio-proxy:
    Container ID:   docker://39640c8de738e8d727f4362e03105a7371d8deaa0c869017b489c789976daa7f
    Image:          docker.io/istio/proxyv2:1.16.1
    Image ID:       docker-pullable://istio/proxyv2@sha256:a861ee2ce3693ef85bbf0f96e715dde6f3fbd1546333d348993cc123a00a0290
    Port:          15090/TCP
    Host Port:     0/TCP
    Args:
      proxy
      sidecar
      --domain
      $(POD_NAMESPACE).svc.cluster.local
      --proxyLogLevel=warning
      --proxyComponentLogLevel=misc:error
      --log_output_level=default:info
      --concurrency
      2
    State:         Running
      Started:     Wed, 11 Jan 2023 23:16:55 +0800
```

docker containers in pod

- You can use minikube dashboard to check all the service, pods, deployments and etc.

minikube dashboard

Name	Labels	Type	Cluster IP	Internal Endpoints	External Endpoints	Created
myapp	app: myapp service: myapp	ClusterIP	10.97.58.17	myapp:8080 TCP myapp:0 TCP	-	36 minutes ago

services

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
myapp-v1-6cc8bbccf6-svlg	myapp:1.0 docker.io/istio/proxyv2:1.16.1	app: myapp pod-template-hash: 6cc8bbccf6 security.istio.io/tlsMode: istio	minikube	Running	0	-	-	4 minutes ago
myapp-v2-587b79c77d-nlgr	myapp:2.0 docker.io/istio/proxyv2:1.16.1	app: myapp pod-template-hash: 587b79c77d security.istio.io/tlsMode: istio	minikube	Running	0	-	-	4 minutes ago

pods

## 2. Now we should deploy destination rule.

- These rules specify configuration for load balancing, connection pool size from the sidecar, and outlier detection settings to detect and evict unhealthy hosts from the load balancing pool.
- Version specific policies can be specified by defining a named `subset`.
- Here we have two versions for myapp(v1 and v2). So we should define these two versions as two subsets.

```
1 apiVersion: networking.istio.io/v1alpha3
2 kind: DestinationRule
3 metadata:
4   name: myapp-destination-rule
5 spec:
6   host: myapp
7   trafficPolicy:
8     loadBalancer:
9       simple: ROUND_ROBIN
10  subsets:
11    - name: v1
12      labels:
13        version: v1
14    - name: v2
15      labels:
16        version: v2
17  #   trafficPolicy:
18  #     loadBalancer:
19  #       simple: ROUND_ROBIN
```

myapp-destination-rule.yaml hosted with ❤ by GitHub

[view raw](#)

```
kubectl apply -f myapp-dr.yaml
```

```
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl apply -f myapp-dr.yaml
destinationrule.networking.istio.io/myapp-destination-rule created
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$
```

deploy destination rule

- Check destination rules.

```
kubectl get dr
```



```
myapp-v2-587b79c77d-high-272-Running-0-0n44m
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples kubectl get dr
NAME HOST AGE
myapp-destination-rule myapp 6h45m
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples
```

- There are 6 different load balancing policies. We can use load balancing policy for subset level and service level.

## LoadBalancerSettings.SimpleLB

Standard load balancing algorithms that require no tuning.

Name	Description
UNSPECIFIED	No load balancing algorithm has been specified by the user. Istio will select an appropriate default.
RANDOM	The random load balancer selects a random healthy host. The random load balancer generally performs better than round robin if no health checking policy is configured.
PASSTHROUGH	This option will forward the connection to the original IP address requested by the caller without doing any form of load balancing. This option must be used with care. It is meant for advanced use cases. Refer to Original Destination load balancer in Envoy for further details.
ROUND_ROBIN	A basic round robin load balancing policy. This is generally unsafe for many scenarios (e.g. when endpoint weighting is used) as it can overburden endpoints. In general, prefer to use LEAST_REQUEST as a drop-in replacement for ROUND_ROBIN.
LEAST_REQUEST	The least request load balancer spreads load across endpoints, favoring endpoints with the least outstanding requests. This is generally safer and outperforms ROUND_ROBIN in nearly all cases. Prefer to use LEAST_REQUEST as a drop-in replacement for ROUND_ROBIN.
LEAST_CONN	Deprecated. Use LEAST_REQUEST instead.

### 3. Deploy Gateway and Virtual Services.

#### VirtualService

- VirtualService defines a set of traffic routing rules with matching criteria.
- If the traffic is matched, then it is sent to a named destination service (or subset/version of it) defined in the registry.

#### Example -

If uri is “api/v1/hello”, it is sent all the traffic to **myapp-v1 subset** and **myapp-v2**. Before deploying this one, we should deploy destination rules because we define all the subsets in the destination rule.

```
- match:
- uri:
  exact: /api/v1/hello
route:
- destination:
  host: myapp
  port:
    number: 8080
  subset: v1
  weight: 75
- destination:
  host: myapp
  port:
    number: 8080
```

```
subset: v2
weight: 25
```

- For **traffic shifting**, we are using **weight**. According to above example, we are routing 25% traffic to myapp-v2 and 75% traffic to myapp-v1.

#### Gateway

- Gateway describes a load balance operating at the edge of the mesh receiving incoming or outgoing HTTP/TCP connections.
- We can expose set of ports in Gateway definition which use the type of protocol.

```
1  apiVersion: networking.istio.io/v1alpha3
2  kind: Gateway
3  metadata:
4    name: myapp-gateway
5  spec:
6    selector:
7      istio: ingressgateway # use istio default controller
8    servers:
9      - port:
10          number: 80
11          name: http
12          protocol: HTTP
13          hosts:
14            - "*"
15    ---
16  apiVersion: networking.istio.io/v1alpha3
17  kind: VirtualService
18  metadata:
19    name: myapp
20  spec:
21    hosts:
22      - "*"
23    gateways:
24      - myapp-gateway
25    http:
26      - match:
27          - uri:
28              exact: /api/v1/hello
29          route:
30            - destination:
31                host: myapp
32                port:
33                  number: 8080
34                subset: v1
35              weight: 75
36            - destination:
37                host: myapp
38                port:
39                  number: 8080
40                subset: v2
41              weight: 25
```

```
kubectl apply -f myapp-gateway.yaml
```

```
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl apply -f myapp-gateway.yaml
gateway.networking.istio.io/myapp-gateway created
virtualservice.networking.istio.io/myapp created
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$
```

- Check virtual services and gateways.

```
kubectl get gw
kubectl get vs
```

```
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ kubectl get gw
NAME          AGE
myapp-gateway 6h50m
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ kubectl get vs
NAME    GATEWAYS          HOSTS    AGE
myapp   ["myapp-gateway"] ["*"]    6h50m
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$
```

After deployment we check the Istio errors using below command.

```
istioctl analyze
```

#### 4. Start minikube tunnel

Minikube tunnel that sends traffic to your Istio Ingress Gateway. This will provide an external load balancer, `EXTERNAL-IP`, for `service/istio-ingressgateway`.

```
minikube tunnel
```

```
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ minikube tunnel
✓ Tunnel successfully started
NOTE: Please do not close this terminal as this process must stay alive for the tunnel to be accessible ...
! The service/ingress istio-ingressgateway requires privileged ports to be exposed: [80 443]
sudo permission will be asked for it.
Starting tunnel for service istio-ingressgateway.
Password:
```

- Check the external IP of ingress gateway load balancer.

```
kubectl -n istio-system get service istio-ingressgateway
-o jsonpath='{.status.loadBalancer.ingress[0].ip}'
```

```
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}'
127.0.0.1
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$
```

- Check the external http2 port of ingress gateway load balancer.

```
kubectl -n istio-system get service istio-ingressgateway
-o jsonpath='{.spec.ports[?(@.name=="http2")].port}'
```

```
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="http2")].port}'
8080
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$
```

- These are the istio-system services which running in minikube after installed istio.

```
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$ kubectl get svc -n istio-system
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)                                     AGE
istio-ingressgateway LoadBalancer  10.106.194.179 <pending>      15021:30329/TCP,80:30733/TCP,443:30638/TCP 5d
istiod               ClusterIP     10.104.149.38 <none>         15010/TCP,15012/TCP,443/TCP,15014/TCP      5d
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp$
```

## 5. Testing

```
curl http://kubectl -n istio-system get service istio-ingressgateway
-o jsonpath='{.status.loadBalancer.ingress[0].ip}':kubectl -n istio-system get service istio-ingressgateway
-o jsonpath='{.spec.ports[?(@.name=="http2")].port}'/api/v1/hello

curl http://127.0.0.1/api/v1/hello
```

- Here you can see, we have two different output for same url.

```
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ curl http://127.0.0.1/api/v1/hello
Hello World, This is version 1!
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ curl http://127.0.0.1/api/v1/hello
Hello World, This is version 1!
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ curl http://127.0.0.1/api/v1/hello
Hello World, This is version 1!
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ curl http://127.0.0.1/api/v1/hello
Hello World, This is version 2!
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ curl http://127.0.0.1/api/v1/hello
Hello World, This is version 1!
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ curl http://127.0.0.1/api/v1/hello
Hello World, This is version 1!
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$ curl http://127.0.0.1/api/v1/hello
Hello World, This is version 2!
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples$
```

## Kiali Console

- Kiali is a console for Istio service mesh. Kiali can be quickly installed as an Istio add-on, or trusted as a part of your production environment.
- There are several addons for Istio.

```
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples/addons ls
README.md      extras          grafana.yaml    jaeger.yaml     kiali.yaml      prometheus.yaml
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples/addons
```

- Deploy Kiali, prometheus using 'kubectl apply -f addons'.

```
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples kubectl apply -f addons
serviceaccount/grafana created
configmap/grafana created
service/grafana created
deployment.apps/grafana created
configmap/istio-grafana-dashboards created
configmap/istio-services-grafana-dashboards created
deployment.apps/jaeger created
service/tracing created
service/zipkin created
service/jaeger-collector created
serviceaccount/kiali created
configmap/kiali created
clusterrole.rbac.authorization.k8s.io/kiali-viewer created
clusterrole.rbac.authorization.k8s.io/kiali created
clusterrolebinding.rbac.authorization.k8s.io/kiali created
role.rbac.authorization.k8s.io/kiali-controlplane created
rolebinding.rbac.authorization.k8s.io/kiali-controlplane created
service/kiali created
deployment.apps/kiali created
serviceaccount/prometheus created
configmap/prometheus created
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
service/prometheus created
deployment.apps/prometheus created
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples
```

```
deployment.apps/prometheus created
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples kubectl get svc -n istio-system
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
grafana	ClusterIP	10.103.84.114	<none>	3000/TCP	60s
istio-ingressgateway	LoadBalancer	10.106.194.179	127.0.0.1	15021:30329/TCP,80:30733/TCP,443:30638/TCP	5d
istiod	ClusterIP	10.104.149.38	<none>	15010/TCP,15012/TCP,443/TCP,15014/TCP	5d
jaeger-collector	ClusterIP	10.104.58.102	<none>	14268/TCP,14250/TCP,9411/TCP	60s
kiali	ClusterIP	10.101.90.253	<none>	20001/TCP,9090/TCP	60s
prometheus	ClusterIP	10.100.161.92	<none>	9090/TCP	59s
tracing	ClusterIP	10.105.106.116	<none>	80/TCP,16685/TCP	60s
zipkin	ClusterIP	10.103.24.74	<none>	9411/TCP	60s

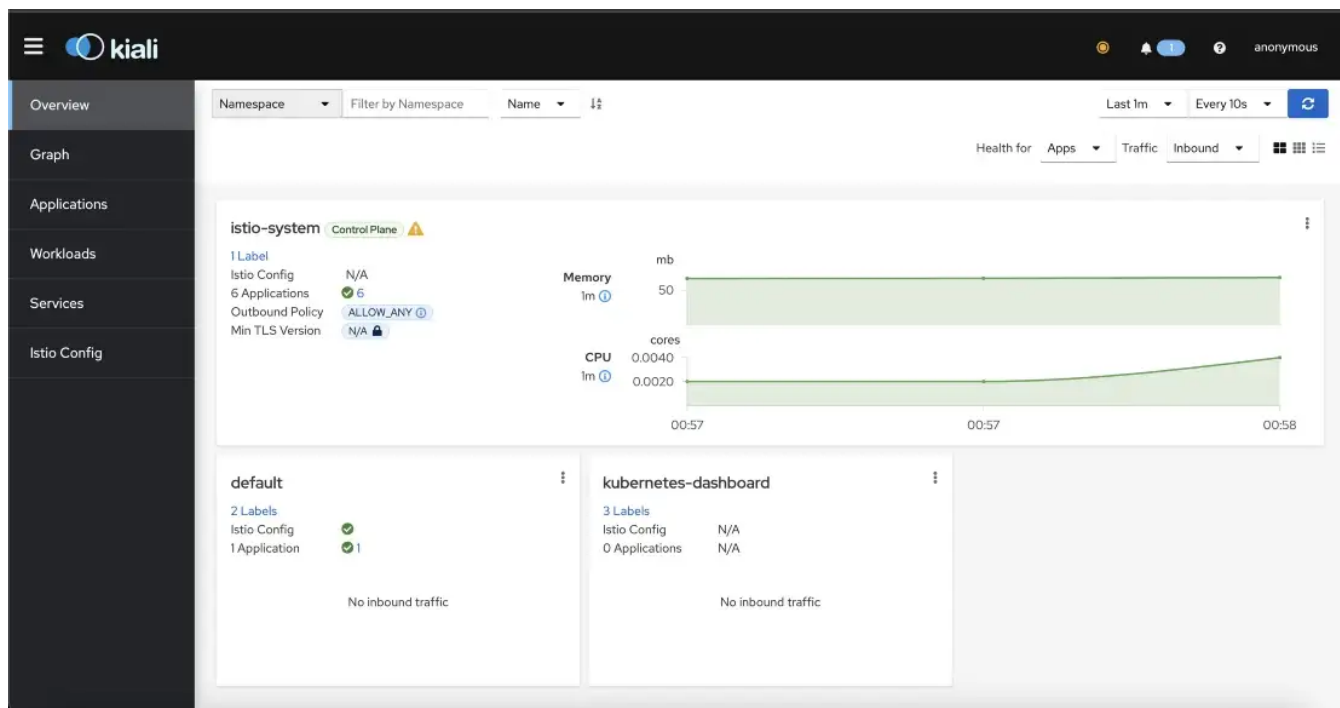
```
pramodshehan@Pramods-MacBook-Pro ~/Work/Istio-installation/istio-1.16.1/samples
```

- Open Kiali dashboard using below command.

```
istioctl dashboard kiali
```

```
Error: no Kiali pods found
pramodshehan@Pramods-MacBook-Pro ~/Work/SpringBoot/myapp istioctl dashboard kiali
http://localhost:20001/kiali
```





Kiali dashboard

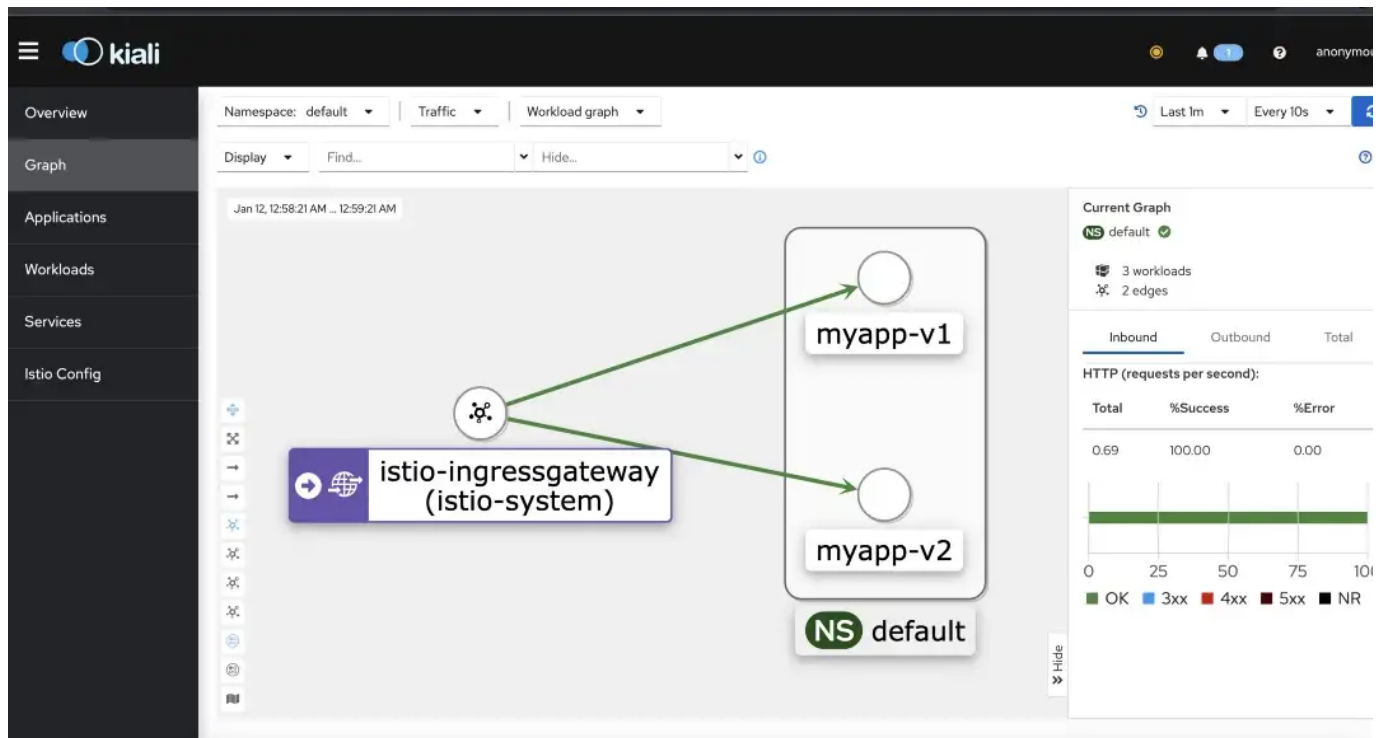
- We can view all the deployment yaml file here which we used for services, deployments, virtual services and etc.

The Kiali dashboard **Istio Config** view shows a table of Istio configurations in the **default** namespace. The table has columns for Name, Namespace, Type, and Configuration.

Name	Namespace	Type	Configuration
<b>VS</b> myapp	<b>NS</b> default	VirtualService	✓
<b>DR</b> myapp-destination-rule	<b>NS</b> default	DestinationRule	✓
<b>G</b> myapp-gateway	<b>NS</b> default	Gateway	✓

- This is how istio-ingressgateway is routing data to myapp-v1 and myapp-v2

Istio    Service Mesh    Microservices    Minikube    Kubernetes



Services

Service Name Filter by Service Name

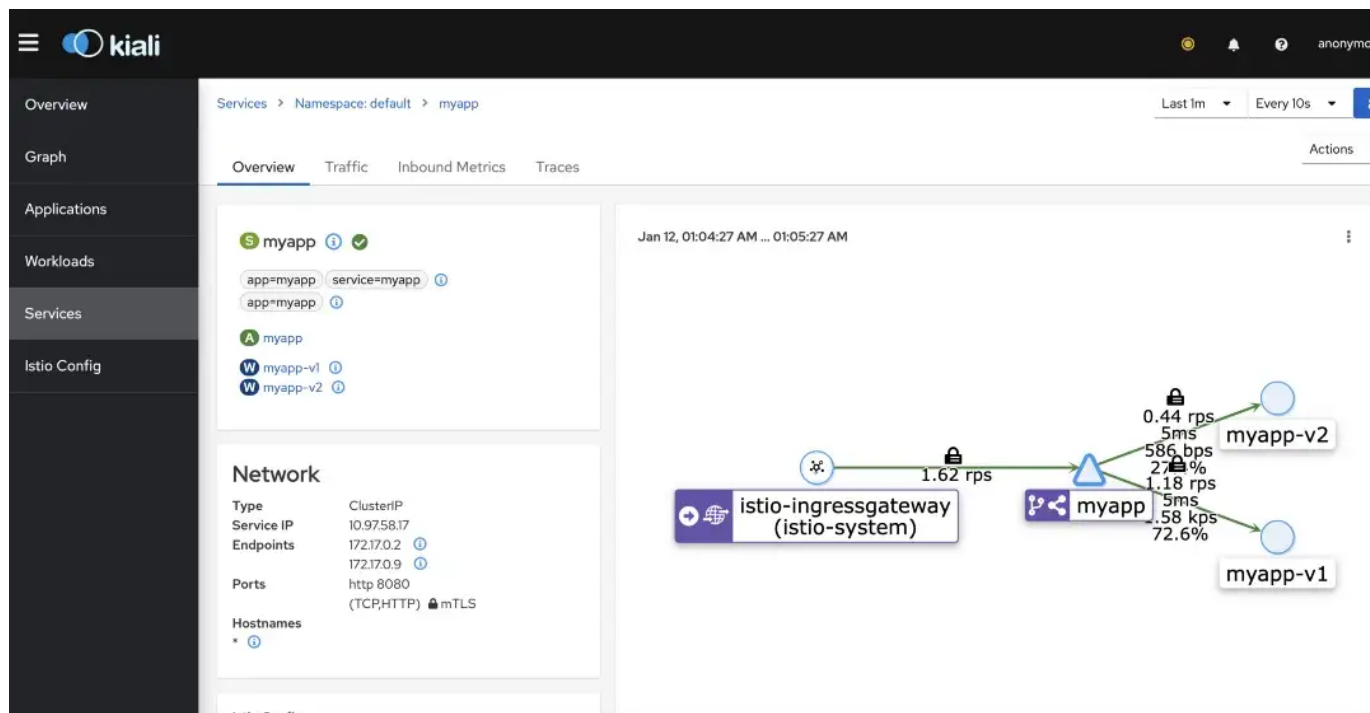
Health	Name	Namespace	Labels	Configurati...	Details
⚠	kubernetes	NS default	component=apiserver provider=kubernetes	✓	
✓	myapp	NS default	app=myapp service=myapp	✓	DR myapp-destination-rule G myapp-gateway VS myapp

- Send some traffic to the application and check the Kiali dashboard.

```
for ((i=1;i<=100;i++)); do sleep 0.5; curl -v --header  
"Connection: keep-alive" "http://127.0.0.1/api/v1/hello"; done
```

- Here you can see, traffic shifting has been worked according to our definition. Traffic has been routed to myapp-v1 around 72.6. for myapp-v2, it is around 27.4





github — <https://github.com/pramodShehan5/istio-traffic-shifting-demo>

## References

<https://istio.io/latest/docs/setup/getting-started/>