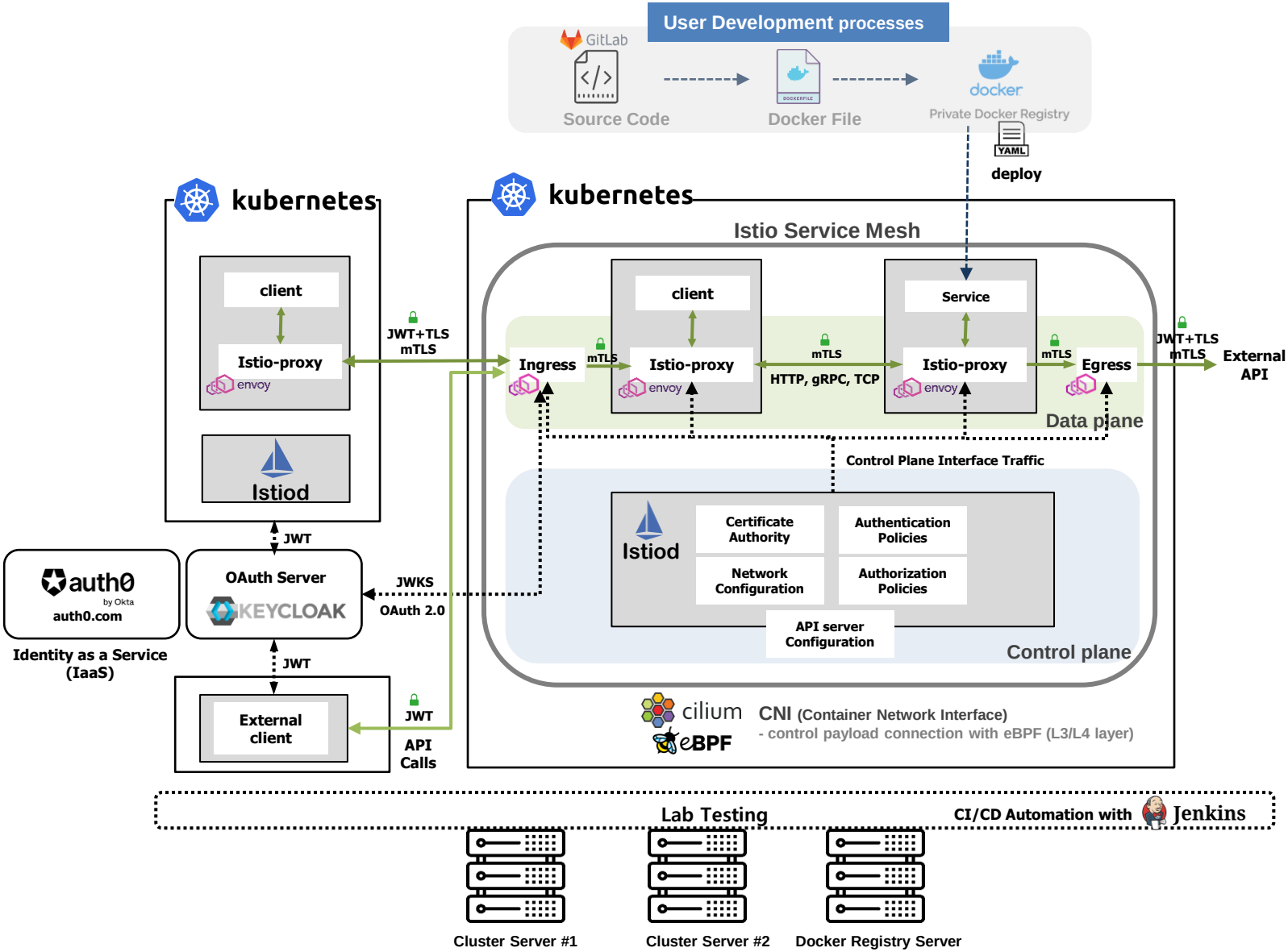
Zero Trust Security Architecture with Istio Service Mesh and OAuth 2.0 in Kubernetes



Kyehwan Lee

Edge-to-Core Service Mesh Testbed with Zero Trust Security Stack

Dell PowerEdge R640

Spec:
Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz
64GB system Memory
32GB DDR4 DIMM, speed up to 3200 MT/s
SCSI Disk 931GiB (999GB)
Embedded NIC: 4× 1GbE ports

Registry

- Docker Registry
- OAuth server
- Storage (NFS) & ETC
- Jenkins CI/CD tool

docker Private Docker Registry · KEYCLOAK · auth0 by Okta · Jenkins

Node 1

- Kubernetes 1.32
- Istio v1.24 + Cilium v1.17
- MetalLB
- Grafana, Kiali for monitoring

kubernetes · Istio · envoy · cilium · Grafana · kiali · O-RAN ALLIANCE

Node 2

- Kubernetes 1.32
- Istio v1.24 + Cilium v1.17
- MetalLB
- External Client app.
- ETC

kubernetes · Istio · envoy · cilium · O-RAN ALLIANCE

Source Code & Manifest files

GitLab · YAML

GitLab Source Code
https://gitlab.nist.gov/gitlab/kyehwanl/zta-testbed

Server Access Info

ssh to Server1

```
[kyehwanl@portal ~]$ ssh onfadmin@5g1-comp1.antd.nist.gov
```

```
vmware-011 [2003]{~}$ ssh onfadmin@5g1-comp2.antd.nist.gov
```

Access Info

Server 1: 5g1-comp1 (10.5.0.2)
Server 2: 5g1-comp2 (10.5.0.3)
Server 3: 5g1-comp3 (10.5.0.4)

ID: onfadmin, PW: 5Gtb@ctl

- **ssh session to each server**
  - From Portal or vm farm
  - Server 1 has Docker Registry, OAuth server (Keycloak), Jenkins server

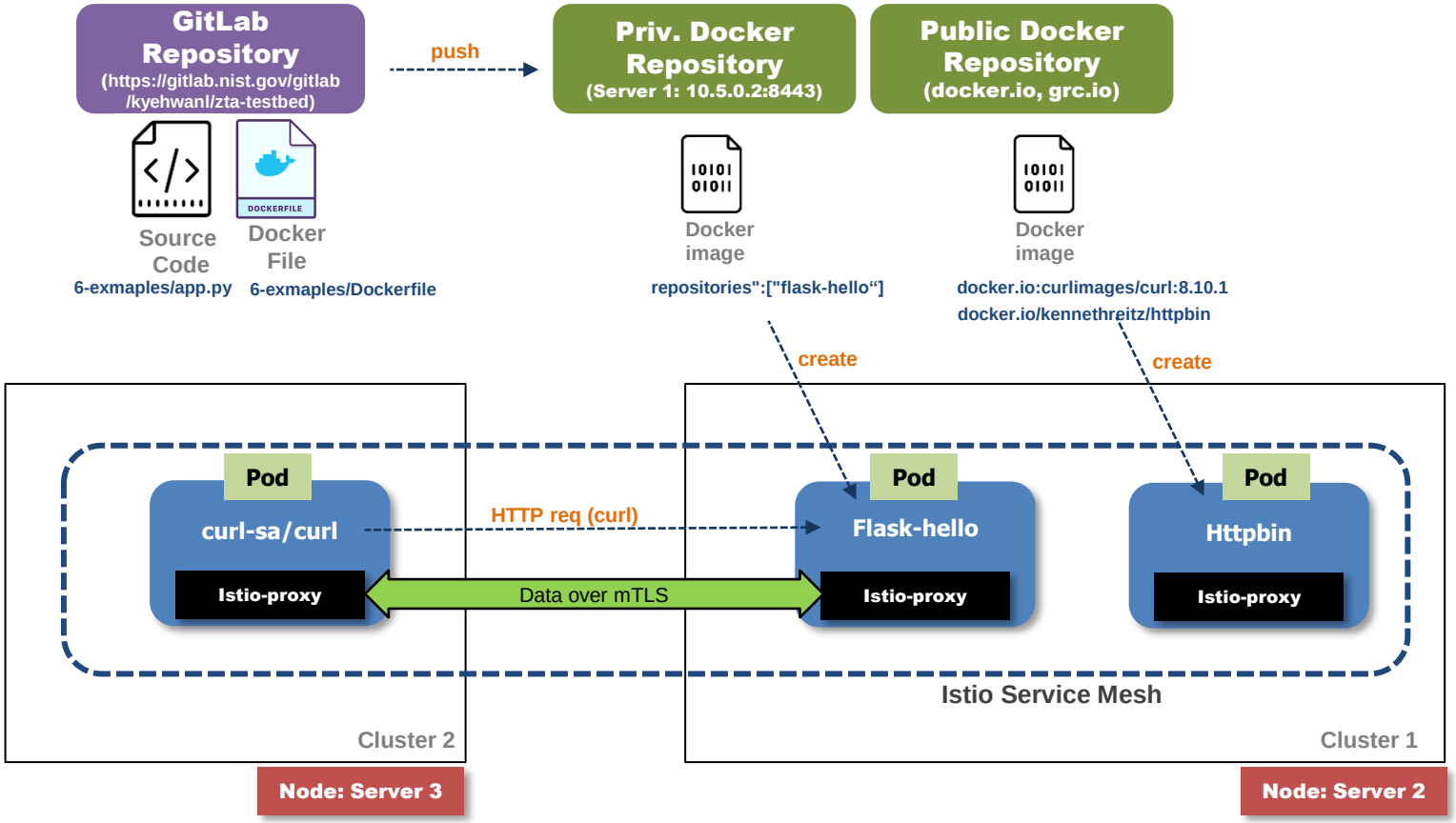
Oauth server (Keycloak) on Server 1

```
CONTAINER ID   IMAGE                                COMMAND                  CREATED
9f9a90c0086d   quay.io/keycloak/keycloak:26.4.0     "/opt/keycloak/bin/k…"   31 hours ago
 STATUS                 PORTS                                                NAMES
 Up 31 hours            8443/tcp, 0.0.0.0:8080->8080/tcp, 9000/tcp           keycloak
```

Docker Private Registry on Server 1

```
onfadmin@5g1-comp1:~$ docker ps
f7ebdc789796   registry:2                                "/entrypoint.sh /etc…"
 PORTS                                                NAMES
5000/tcp, 0.0.0.0:8443->443/tcp, :::8443->443/tcp     registry
```

Sample Test case



For installation : https://gitlab.nist.gov/gitlab/kyehwanl/zta-testbed/-/blob/main/Installation_Guide_Testbed-KyeHwanLee-v1.1.pdf

Private Docker Registry - setup

Pre-requisites for credentials (with Docker & Kubernetes)

- Place TLS credential for docker service
  - In order to access the docker registry running on 5g1-comp1 server (10.5.0.2) from another host, locate TLS certificate (/etc/docker/certs.d/)

```
onfadmin@5g1-comp2:~/Downloads/zta-testbed$ mkdir -p  /etc/docker/certs.d/10.5.0.2\:8443/
onfadmin@5g1-comp2:~/Downloads/zta-testbed$ cp 6-exmaples/tls.crt /etc/docker/certs.d/10.5.0.2\:8443/
```

- Place TLS credential for Kubernetes service
  - To access the docker registry with Kubernetes,
    - update ca certificate with the following command so that the system may recognize and update its ca certs pool

```
onfadmin@5g1-comp2:~/Downloads/zta-testbed$ sudo cp 6-exmaples/tls.crt /usr/local/share/ca-certificates/
onfadmin@5g1-comp2:~/Downloads/zta-testbed$ sudo update-ca-certificates
onfadmin@5g1-comp2:~/Downloads/zta-testbed$ sudo systemctl restart containerd
```

Private Docker Registry – how to upload (1)

Docker Image repository

- Running on Server 1 (10.5.0.2) with TLS port **8443**
- How to Push images into the repo
  - Tag: docker tag <user-image>[:ver] <RegistryIP>:8443/<image-name>[:version]

```
onfadmin@5g1-comp1:~$ docker tag flask-hello:latest 10.5.0.2:8443/flask-hello
```

  - git push/pull <RegistryIP>:8443/<image-name>[:version]

```
onfadmin@5g1-comp1:~$ docker push 10.5.0.2:8443/flask-hello
```

  - If there is no tls certs in previous prerequisite setup, TLS error occurs on git push/pull

- List the images and check the version info

```
onfadmin@5g1-comp2:~$ curl -k https://10.5.0.2:8443/v2/_catalog
{"repositories":["flask-hello","nginx"]}
```

```
onfadmin@5g1-comp2:~$ curl -k https://10.5.0.2:8443/v2/flask-hello/tags/list
{"name":"flask-hello","tags":["latest"]}
```

Private Docker Registry – how to upload (2)

How to use in Kubernetes cluster

1. Deploy registry secret

   ```
   kubectl apply –f 6-exmaples/registry-secret.yaml
   ```

2. Deploy deployment manifest with the image info

   ```
   kubectl apply –f 6-exmaples/flask-hello-service-deploy-tb.yaml
   ```

3. Check out the deploy/pod is available

   ```
   kubectl get pod [namespace] <pod name>
   ```

6-exmaples/flask-hello-service-deploy-tb.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-hello
  namespace: sample
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flask-hello
  template:
    metadata:
      labels:
        app: flask-hello
    spec:
      containers:
      - name: flask-hello
        image: 10.5.0.2:8443/flask-hello:latest
        ports:
        - containerPort: 80
      imagePullSecrets:
      - name: registry-ca
```

Image from
Private Docker Registry

```
$ kubectl -f 6-examples/flask-hello-service-deploy-tb.yaml
```

6-exmaples/registry-secret.yaml

registry-secret.yaml 2.65 KiB

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: registry-ca
  namespace: kube-system
type: Opaque
data:
  registry-ca: LS0tLS1CRUdJTiBDRVJ
```

Credential
(TLS certs hex codes)

```
$ kubectl apply -f 6-exmaples/registry-secret.yaml
```

Exploring Clusters

- **Pods and Services are shown by kubectl command**
  - kubectl get [-n namespce] <pod|service> <pod|service name> [options]
    - Kubectl get pod –n sample <curl-pod>
    - Kubectl get service –n sample <curl-service>
  - Using Kubernetes CLI managing tool: k9s (https://k9scli.io/)

Cluster 1

```
onfadmin@5g1-comp2:~$ kubectl get po --all-namespaces
NAMESPACE      NAME                                       READY   STATUS    RESTARTS        AGE
default        client-575688f76-9dmtv                     1/1     Running   0               84d
ingress-nginx  ingress-nginx-controller-dcf9c89b4-7xgfg   1/1     Running   0               32h
istio-system   grafana-6b45c49476-72phw                   1/1     Running   0               134d
istio-system   istio-eastwestgateway-84d87d448d-m6v47     1/1     Running   0               138d
istio-system   istio-ingressgateway-748855649b-5fs4f      1/1     Running   0               138d
istio-system   istiod-896cc8f4b-xh9mq                     1/1     Running   0               138d
istio-system   kiali-79b6d98d5d-nckwp                     1/1     Running   0               134d
istio-system   prometheus-6dd9fd5446-ctwlm                2/2     Running   0               134d
jwt-test       httpbin-5d76c6469b-zbj54                   2/2     Running   0               133d
kube-system    cilium-operator-ddb9b866-xvcqk             1/1     Running   1 (138d ago)    139d
kube-system    cilium-t6zrx                               1/1     Running   1 (138d ago)    139d
kube-system    coredns-7c65d6cfc9-5zjlg                   1/1     Running   0               139d
kube-system    coredns-7c65d6cfc9-trqjd                   1/1     Running   0               139d
kube-system    etcd-5g1-comp2                             1/1     Running   7 (138d ago)    139d
kube-system    kube-apiserver-5g1-comp2                   1/1     Running   2 (138d ago)    139d
kube-system    kube-controller-manager-5g1-comp2          1/1     Running   2 (138d ago)    139d
kube-system    kube-proxy-s652v                           1/1     Running   1 (138d ago)    139d
kube-system    kube-scheduler-5g1-comp2                   1/1     Running   2 (138d ago)    139d
metallb-system controller-5456bd6d98-xkzfm                1/1     Running   0               138d
metallb-system speaker-5x8k6                              1/1     Running   0               138d
sample         curl-5b549b49b8-5jfmf                      2/2     Running   0               138d
sample         flask-hello-64576d5bf-x9g5w                2/2     Running   0               133d
sample         helloworld-v1-6d65866976-g52xg             2/2     Running   0               138d
sample         httpbin-569988df8-jfcth                    2/2     Running   0               134d
```

Cluster 2

```
onfadmin@5g1-comp2:~$ sudo k9s
```

```
Context: cluster2                    <0> all       <a>      Attach      <ctrl-k>  Kill
Cluster: cluster2                    <1> default   <ctrl-d> Delete      <l>       Logs
User:    cluster2-admin                            <d>      Describe    <p>       Logs Pre
K9s Rev: v0.32.5 ⚡ v0.50.15                       <e>      Edit        <shift-f> Port-For
K8s Rev: v1.31.9                                   <?>      Help        <z>       Sanitize
CPU:     n/a                                       <shift-j> Jump Owner <s>       Shell
MEM:     n/a

                                  Pods(all)[18]
NAMESPACE↑   NAME                                  PF  READY  STATUS   RESTARTS  IP              NODE       AGE
default      nginx-676b6c5bbc-dp4vg                ●   1/1    Running  0         192.168.0.193   5g1-comp3  140d
istio-system istio-eastwestgateway-5b59c747d6-7t6rx ●  1/1    Running  0         192.168.0.106   5g1-comp3  140d
istio-system istio-ingressgateway-5b8fbcfb9b-5xtsj ●   1/1    Running  0         192.168.0.94    5g1-comp3  140d
istio-system istiod-86897b9fd9-lkd4s               ●   1/1    Running  0         192.168.0.211   5g1-comp3  140d
kube-system  cilium-operator-ddb9b866-p6f2m        ●   1/1    Running  0         10.5.0.4        5g1-comp3  140d
kube-system  cilium-t9hxl                          ●   1/1    Running  0         10.5.0.4        5g1-comp3  140d
kube-system  coredns-7c65d6cfc9-blwmc              ●   1/1    Running  0         192.168.0.205   5g1-comp3  142d
kube-system  coredns-7c65d6cfc9-f9qsn              ●   1/1    Running  0         192.168.0.186   5g1-comp3  142d
kube-system  etcd-5g1-comp3                        ●   1/1    Running  2         10.5.0.4        5g1-comp3  142d
kube-system  kube-apiserver-5g1-comp3              ●   1/1    Running  2         10.5.0.4        5g1-comp3  142d
kube-system  kube-controller-manager-5g1-comp3     ●   1/1    Running  2         10.5.0.4        5g1-comp3  142d
kube-system  kube-proxy-tqsvn                      ●   1/1    Running  2         10.5.0.4        5g1-comp3  142d
kube-system  kube-scheduler-5g1-comp3              ●   1/1    Running  2         10.5.0.4        5g1-comp3  142d
metallb-system controller-5456bd6d98-4rhx5         ●   1/1    Running  0         192.168.0.93    5g1-comp3  140d
metallb-system speaker-bdqrs                       ●   1/1    Running  0         10.5.0.4        5g1-comp3  140d
sample       curl-75bf7fcf64-8l2lb                 ●   2/2    Running  0         192.168.0.198   5g1-comp3  45h
sample       curl-bash                             ●   2/2    Running  0         192.168.0.119   5g1-comp3  2d16h
sample       helloworld-v2-7dcd9b496d-dcnbm        ●   2/2    Running  0         192.168.0.84    5g1-comp3  140d
```

Example-Deploy services in the cloud (1)

- **3 major resources for deploying the service**
  - **Service account** – for identity its account (also, used in SPIFFE ID)
  - **Deployment** – actual user image, application deployment
  - **Service** – expose user's application to outside of cluster
- **6-exmaples/curl-service-deploy.yaml on cluster 2 (5g1-comp3, Server 3)**

```
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ kubectl apply -f 6-exmaples/curl-service-deploy.yaml
```

```yaml
# 1) ServiceAccount
apiVersion: v1
kind: ServiceAccount
metadata:
  name: curl-sa
  namespace: sample
```

```yaml
# 2) Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: curl
  namespace: sample
spec:
  replicas: 1
  selector:
    matchLabels:
      app: curl
  template:
    metadata:
      labels:
        app: curl
      annotations:
        proxy.istio.io/config: |
          proxyMetadata:
            ISTIO_META_DNS_CAPTURE: "true"
            ISTIO_META_DNS_AUTO_ALLOCATE: "true"
    spec:
      serviceAccountName: curl-sa
      containers:
      - name: curl
        image: curlimages/curl:8.10.1
        imagePullPolicy: IfNotPresent
        command: ["sh","-c","sleep 3650d"]
```

```yaml
# 3) Service (for internal use; optional)
apiVersion: v1
kind: Service
metadata:
  name: curl
  namespace: sample
spec:
  selector:
    app: curl
  ports:
  - name: tcp
    port: 80
    targetPort: 80
```

Example-Deploy services in the cloud (2)

- Creation – flask-hello service on cluster1 (5g1-comp2, Server 2)

```
onfadmin@5g1-comp2:~/Downloads/zta-testbed$ kubectl apply -f 6-exmaples/flask-hello-service-deploy-tb.yaml
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: flask-hello
  namespace: sample
spec:
  selector:
    app: flask-hello
  ports:
  - name: http      #  Istio port naming convention  (for HTTP)
    protocol: TCP
    port: 80
    targetPort: 80

```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-hello
  namespace: sample
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flask-hello
  template:
    metadata:
      labels:
        app: flask-hello
    spec:
      containers:
      - name: flask-hello
        image: 10.5.0.2:8443/flask-hello:latest
        ports:
        - containerPort: 80
      imagePullSecrets:
      - name: registry-ca
```

Example-Deploy services in the cloud (3)

- Creation – httpbin service on cluster1 (5g1-comp2, Server 2)

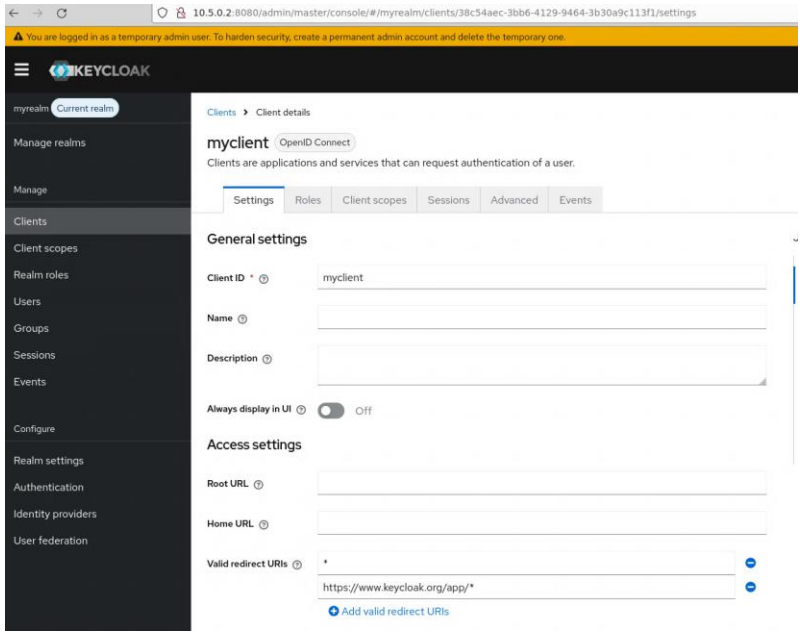
```
onfadmin@5g1-comp2:~/Downloads/zta-testbed$ kubectl apply -f 6-exmaples/httpbin-service-deploy.yaml
```

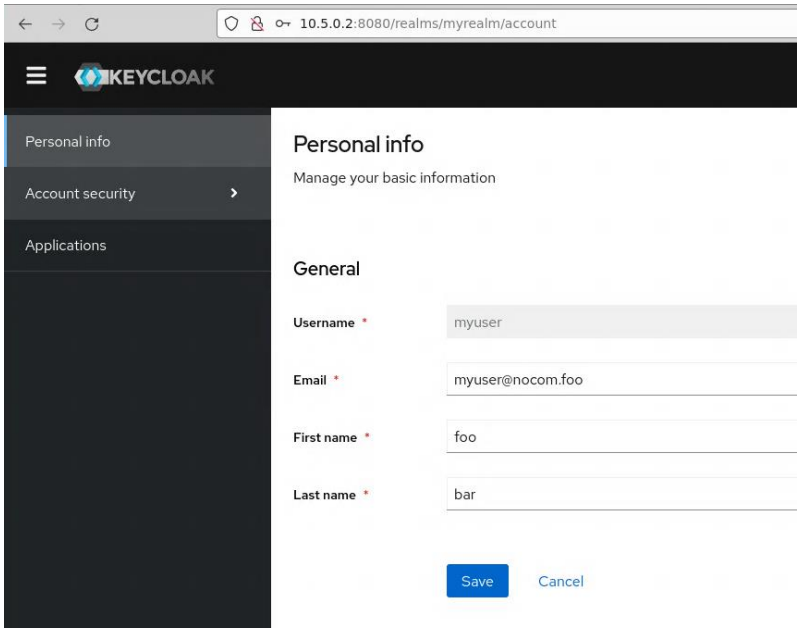
```yaml
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  namespace: sample
  labels:
    app: httpbin
spec:
  ports:
  - port: 80
    targetPort: 80
    name: http
  selector:
    app: httpbin
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpbin
  namespace: sample
  labels:
    app: httpbin
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpbin
  template:
    metadata:
      labels:
        app: httpbin
    spec:
      containers:
      - name: httpbin
        image: docker.io/kennethreitz/httpbin
        ports:
        - containerPort: 80
```

OAuth Server Setting (Keycloak)

- **For OIDC, Keycloak uses realm, users, clients**
  - Official page: https://www.keycloak.org/getting-started/
  - Those info will be used for ***issuer*** and ***jwksurl*** in RequestAuthenication resource
- **Access to Keycloak server on Server1 (5g1-comp1, 10.5.0.2)**
  - http://10.5.0.2:8080/admin for realm, client creation/update/delete
  - http://10.5.0.2:8080/realms/<realm-name>/myaccount for user setting





Authentication, Authorization for OIDC
(applied only to httpbin service)

- **Set up Authentication to httpbin service on Cluster 1**

```
onfadmin@5g1-comp2:~/Downloads/zta-testbed$ kubectl apply -f Keycloak/ra-docker.yaml
```

ra-docker.yaml 341 B

```yaml
# httpbin-requestauth.yaml
apiVersion: security.istio.io/v1beta1
kind: RequestAuthentication
metadata:
  name: httpbin-jwt
  namespace: sample
spec:
  selector:
    matchLabels:
      app: httpbin
  jwtRules:
  - issuer: "http://10.5.0.2:8080/realms/myrealm"
    jwksUri: "http://10.5.0.2:8080/realms/myrealm/protocol/openid-connect/certs"
```

- **Set up Authorization to httpbin service**

```
onfadmin@5g1-comp2:~/Downloads/zta-testbed$ kubectl apply -f Keycloak/ap-docker.yaml
```

ap-docker.yaml 278 B

```yaml
# httpbin-authz.yaml
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: httpbin-allow-jwt
  namespace: sample
spec:
  selector:
    matchLabels:
      app: httpbin
  action: ALLOW
  rules:
  - from:
    - source:
        requestPrincipals: ["*"]
```

Example-Deploy services in the cloud (2)

- curl request (from cluster2) to the httpbin and flaks-hello services on cluster1 (without Access Token)

```
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ kubectl exec -it -n sample deploy/curl
-- curl -sS http://flask-hello.sample.svc.cluster.local/hello
hello world
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ kubectl exec -it -n sample deploy/curl
 -- curl -sS http://httpbin.sample.svc.cluster.local/get
RBAC: access denied
```

Access Denied with 403 Forbidden Error on HTTP response due to Auth error

curl -v http:// …

```
* Request completely sent off
< HTTP/1.1 403 Forbidden
< content-length: 19
< content-type: text/plain
< date: Mon, 20 Oct 2025 01:23:24 GMT
< server: envoy
< x-envoy-upstream-service-time: 5
<
* Connection #0 to host httpbin.sample.svc.cluster.local left intact
RBAC: access denied
```

- curl request to the httpbin with Access Token
  - First, obtain access token from OAuth server with the following parameters
    - realm: myrealm
    - Client ID: myclient
    - User credential (type: password): myuser/myuser

```
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ KC=http://10.5.0.2:8080
REALM=myrealm
CLIENT_ID=myclient
USERNAME=myuser
PASSWORD=myuser
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ ACCESS_TOKEN=$(curl -s \
  -d "client_id=$CLIENT_ID" \
  -d "grant_type=password" \
  -d "username=$USERNAME" \
  -d "password=$PASSWORD" \
  "$KC/realms/$REALM/protocol/openid-connect/token" \
  | sed -E 's/.*"access_token":"([^"]+)".*/\1/')
```

Example-Deploy services in the cloud (3)

Send the curl request with jwt info

- httpbin(cluster1) response to the curl (cluster2) request with ACCESS Token
  - Attaching access token info into the header using -H “Authorization:Bearer <token>”

```
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ kubectl exec -n sample deployments/curl -- curl -sS -H "Authorization: Bearer $ACCESS_TOKEN
" http://httpbin.sample.svc.cluster.local/get
{
  "args": {},
  "headers": {
    "Accept": [
      "*/*"
    ],
    "Host": [
      "httpbin.sample.svc.cluster.local"
    ],
    "User-Agent": [
      "curl/8.10.1"
    ],
    "X-Envoy-Attempt-Count": [
      "1"
    ],
    "X-Forwarded-Client-Cert": [
      "By=spiffe://cluster.local/ns/sample/sa/default;Hash=fbf82a0bb655fb351e
";URI=spiffe://cluster.local/ns/sample/sa/curl-sa"
    ],
    "X-Forwarded-Proto": [
      "http"
    ],
    "X-Request-Id": [
      "a8157e9d-5335-44d2-b7cd-907d97b97c19"
    ]
  },
  "method": "GET",
  "origin": "127.0.0.6:51301",
  "url": "http://httpbin.sample.svc.cluster.local/get"
}
```

Access Token info

```
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ echo $ACCESS_TOKEN | cut -d. -f2 | base64 -d | jq
{
  "exp": 1760733012,
  "iat": 1760732712,
  "jti": "onrtro:79450d84-ca67-efe9-de83-ca41bf7c6976",
  "iss": "http://10.5.0.2:8080/realms/myrealm",
  "aud": "account",
  "sub": "e9c17aa2-9519-46cc-8528-b9127aad9b98",
  "typ": "Bearer",
  "azp": "myclient",
  "sid": "6e44840c-b210-2cfa-bef5-5239111233e1",
  "acr": "1",
  "allowed-origins": [
    "https://www.keycloak.org"
  ],
  "realm_access": {
    "roles": [
      "default-roles-myrealm",
      "offline_access",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "scope": "email profile",
  "email_verified": false,
  "name": "foo bar",
  "preferred_username": "myuser",
  "given_name": "foo",
  "family_name": "bar",
  "email": "myuser@nocom.foo"
}
```

Using sample traffic generator w/wo token

- curl request to both httpbin and flask-hello services on Cluster1
- sends curl without token info, traffic-gen-curl-fixpos.sh
- sends curl with token info, traffic-gen-curl-token-fixpos.sh (token is obtained inside the script every 300s)

```
$ cat Keycloak/traffic-gen-curl-fixpos.sh | \
  kubectl exec -it -n sample deploy/curl -- /bin/sh
```

```
#00002 | flask-hello: hello world 200 0.006272
       | httpbin:     RBAC: access denied 403 0.003220

flask-hello body: hello world
httpbin body:     RBAC: access denied
```

HTTP 200 ok

HTTP 403 Forbidden

```
$ cat Keycloak/traffic-gen-curl-token-fixpos.sh | \
  kubectl exec -it -n sample deploy/curl -- /bin/sh
```

```
#00004 | token_age=  4s (refresh=300s)
       | flask-hello: 200 0.006097
       | httpbin:     200 0.004664

flask-hello body: hello world
httpbin body:     {
  "args": {},
  "headers": {
    "Accept": [
      "*/*"
    ],
    "Host": [
      "httpbin.sample.svc.cluster.local"
    ],
    "User-Agent": [
      "curl/8.10.1"
    ],
    "X-Envoy-Attempt-Count": [
      "1"
    ],
    "X-Forwarded-Client-Cert": [
      "By=spiffe://cluster.local/ns/sample/sa/default;Hash=b2eee53299d7
    ],
    "X-Forwarded-Proto": [
      "http"
    ],
    "X-Request-Id": [
      "2484b6d6-772d-4991-b93d-8d4ae15b38cb"
    ]
  },
  "method": "GET",
  "origin": "127.0.0.6:49583",
  "url": "http://httpbin.sample.svc.cluster.local/get"
}
```

Observability - Kiali

Using Remote Desktop Access for GUI

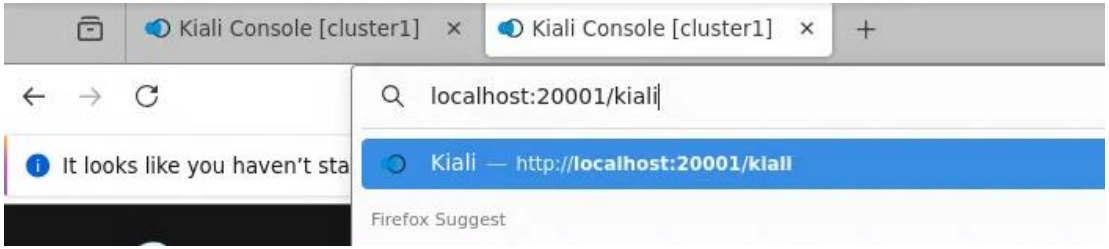
- Terminal client app, “APACHE GUACAMOLE”
  - http://docker.antd.nist.gov:8080/guacamole/#/
- Remote Desktop connection with port-forwarding
- Direct connection using ssh proxy jump

Enable command (if not enabled)

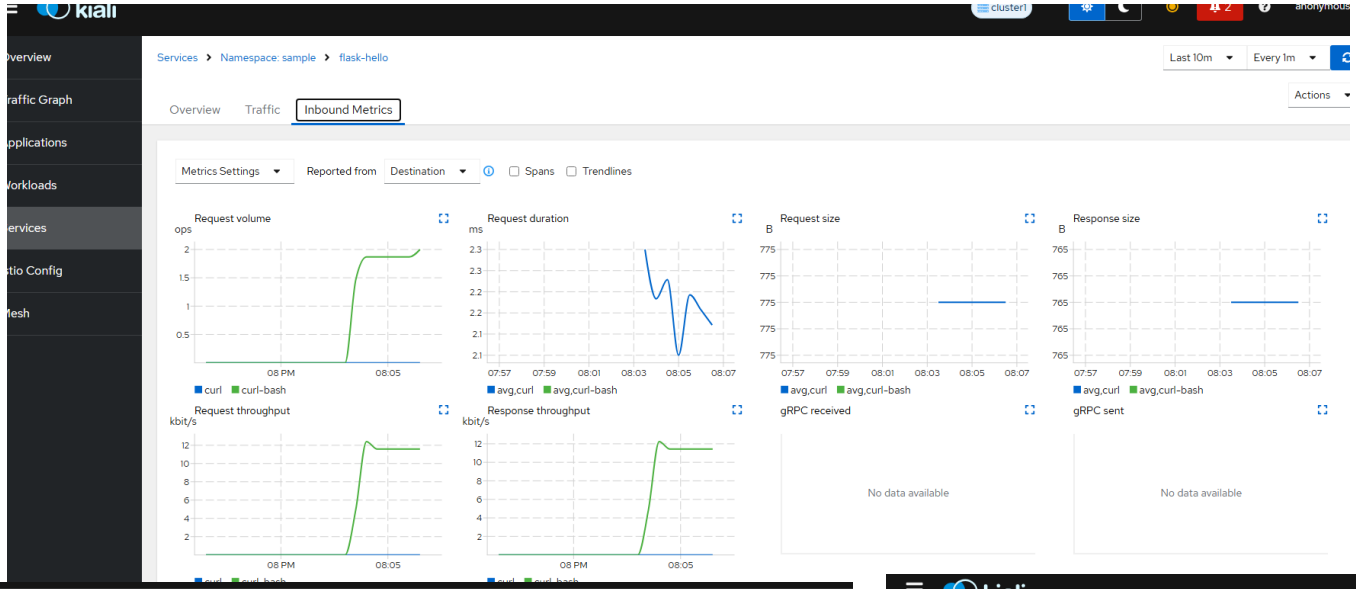
- Command: istioctl dashboard kiali

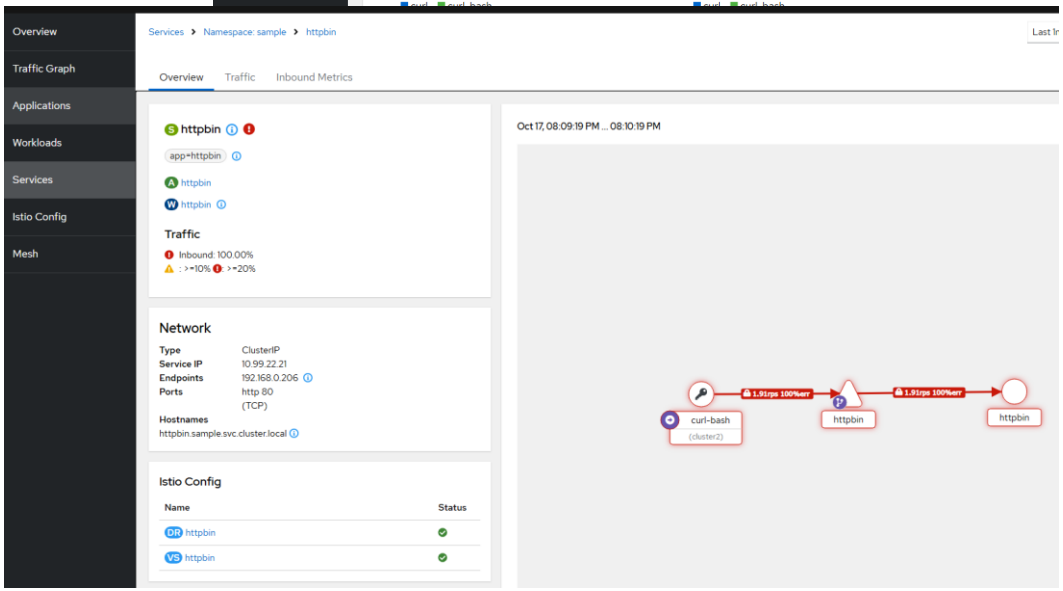
```
onfadmin@5g1-comp2:~/Downloads/istio-1.24.3/samples/addons$ istioctl dashboard kiali
http://localhost:20001/kiali
Failed to open browser; open http://localhost:20001/kiali in your browser.
```

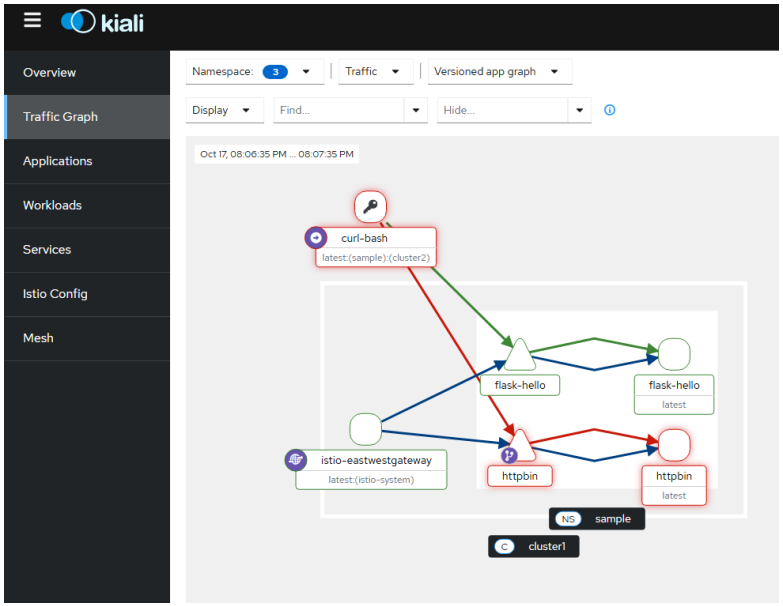
Browser -> localhost:20001/kiali



Kiali Visualization
– Traffic, service, workload metrics







Observability - Grafana

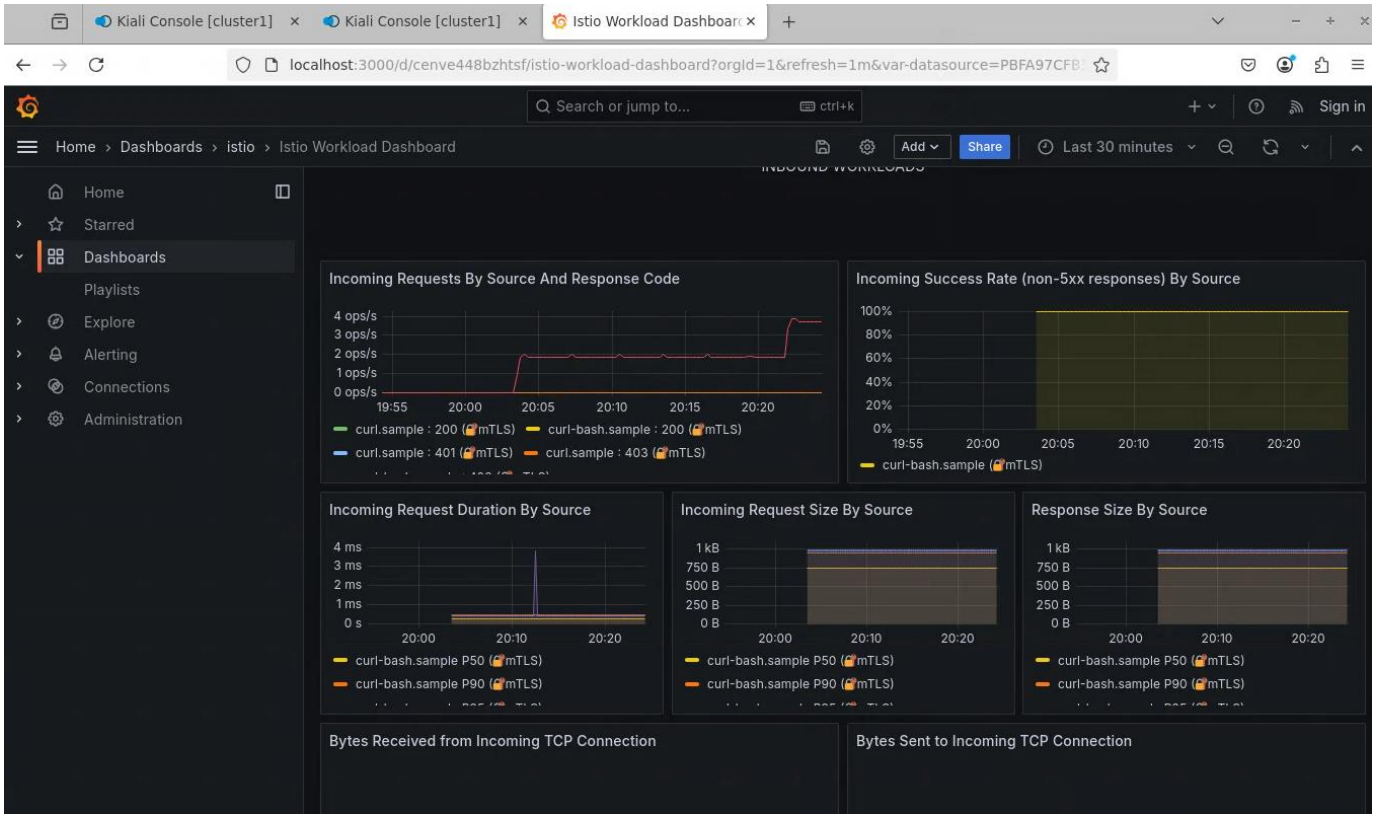
Enable Command (if not enabled)

- istioctl dashboard grafana

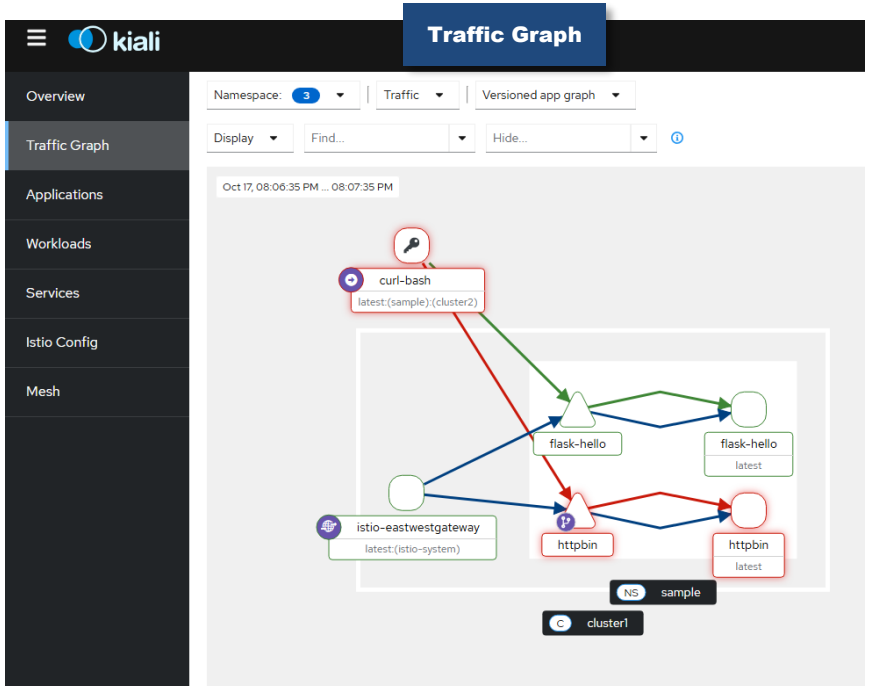
```
onfadmin@5g1-comp2:~$ istioctl dashboard grafana
http://localhost:3000
Failed to open browser; open http://localhost:3000 in your browser.
```

Browser -> localhost:3000

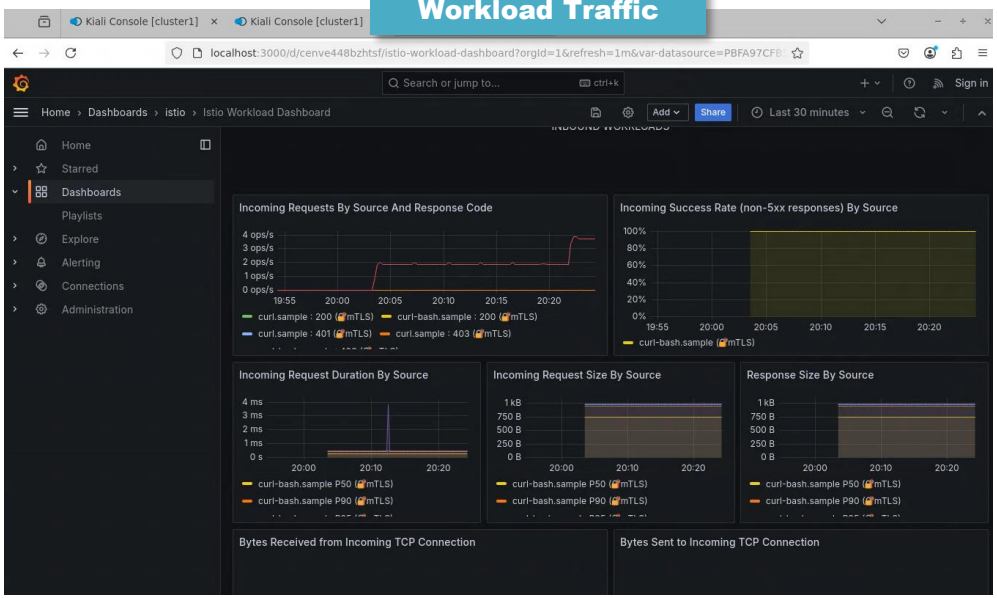
- httpbin workload statistics



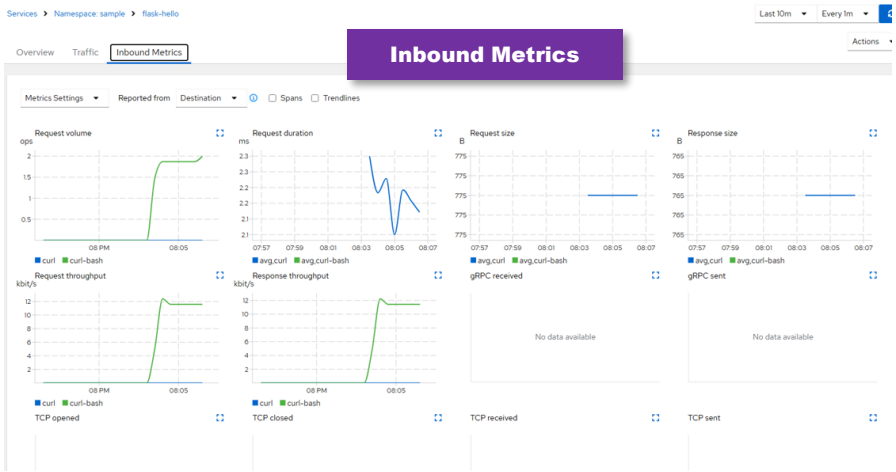
Traffic Graph



Workload Traffic



Inbound Metrics



Cluster Status

```
onfadmin@5g1-comp2:~$ kubectl get po
NAMESPACE      NAME                                       READY   STATUS    RESTARTS      AGE
default        client-575688f76-9dmtv                     1/1     Running   0             84d
ingress-nginx  ingress-nginx-controller-dcf9c89b4-7xgfg   1/1     Running   0             32h
istio-system   grafana-6b45c49476-72phw                   1/1     Running   0             134d
istio-system   istio-eastwestgateway-84d87d448d-m6v47     1/1     Running   0             138d
istio-system   istio-ingressgateway-748855649b-5fs4f      1/1     Running   0             138d
istio-system   istiod-896cc8f4b-xh9mq                     1/1     Running   0             138d
istio-system   kiali-79b6d98d5d-nckwp                     1/1     Running   0             134d
istio-system   prometheus-6dd9fd5446-ctwlm                2/2     Running   0             134d
jwt-test       httpbin-5d76c6469b-zbj54                   2/2     Running   0             133d
kube-system    cilium-operator-ddb9b866-xvcqk             1/1     Running   1 (138d ago)  139d
kube-system    cilium-t6zrx                               1/1     Running   1 (138d ago)  139d
kube-system    coredns-7c65d6cfc9-5zjlg                   1/1     Running   0             139d
kube-system    coredns-7c65d6cfc9-trqjd                   1/1     Running   0             139d
kube-system    etcd-5g1-comp2                             1/1     Running   7 (138d ago)  139d
kube-system    kube-apiserver-5g1-comp2                   1/1     Running   2 (138d ago)  139d
kube-system    kube-controller-manager-5g1-comp2          1/1     Running   2 (138d ago)  139d
kube-system    kube-proxy-s652v                           1/1     Running   1 (138d ago)  139d
kube-system    kube-scheduler-5g1-comp2                   1/1     Running   2 (138d ago)  139d
metallb-system controller-5456bd6d98-xkzfm                1/1     Running   0             138d
metallb-system speaker-5x8k6                              1/1     Running   0             138d
sample         curl-5b549b49b8-5jfmf                      2/2     Running   0             138d
sample         flask-hello-64576d5bf-x9g5w                2/2     Running   0             133d
sample         helloworld-v1-6d65866976-g52xg             2/2     Running   0             138d
sample         httpbin-569988df8-jfcth                    2/2     Running   0             134d
```

References

[1] https://gitlab.nist.gov/gitlab/kyehwanl/zta-testbed
[2] https://www.keycloak.org/getting-started/getting-started-docker
[3] https://istio.io/latest/docs/tasks/observability/metrics/using-istio-dashboard/

EXTRA Slides

FYI. To Obtain Access Token

- Get access token with “get-access-token.sh”

```
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ . ./Keycloak/get-access-token.sh
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ echo $ACCESS_TOKEN | cut -d. -f2 | base64 -d | jq
base64: invalid input
{
  "exp": 1760928257,
  "iat": 1760927957,
  "jti": "onrtro:aaaaac3b-e46b-fe35-500b-d414329e3366",
  "iss": "http://10.5.0.2:8080/realms/myrealm",
  "aud": "account",
  "sub": "e9c17aa2-9519-46cc-8528-b9127aad9b98",
  "typ": "Bearer",
  "azp": "myclient",
  "sid": "67ee109b-12c9-8ac6-a453-47ed7aced4b8",
  "acr": "1",
  "allowed-origins": [
    "https://www.keycloak.org"
  ],
  "realm_access": {
    "roles": [
      "default-roles-myrealm",
      "offline_access",
      "uma_authorization"
    ]
  },
  "resource_access": {
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "scope": "email profile",
  "email_verified": false,
  "name": "foo bar",
  "preferred_username": "myuser",
  "given_name": "foo",
  "family_name": "bar",
  "email": "myuser@nocom.foo"
}
```

```
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ date -d @1760928257
Sun 19 Oct 2025 10:44:17 PM EDT
onfadmin@5g1-comp3:~/Downloads/zta-testbed$ date
Sun 19 Oct 2025 10:39:58 PM EDT
```

Dynamically Install image within .gitlab-ci.yml

```yaml
image: docker:stable

services:
  - docker:dind

variables:
  REGISTRY_URL: registry.mycompany.com
  IMAGE_NAME: $REGISTRY_URL/myteam/myapp
  DOCKER_DRIVER: overlay2
  DOCKER_TLS_CERTDIR: ""

stages:
  - build

build-image:
  stage: build
  script:
    - echo "$REGISTRY_PASSWORD" | docker login "$REGISTRY_URL" -u "$REGISTRY_USERNAME" --password-

    # 여기서 Dockerfile을 동적으로 생성
    - |
      cat > Dockerfile << 'EOF'
      FROM python:3.12-slim
      WORKDIR /app
      COPY . /app
      RUN pip install --no-cache-dir -r requirements.txt
      CMD ["python", "main.py"]
      EOF

    - docker build -t "$IMAGE_NAME:$CI_COMMIT_SHORT_SHA" .
    - docker push "$IMAGE_NAME:$CI_COMMIT_SHORT_SHA"
  only:
    - main
```