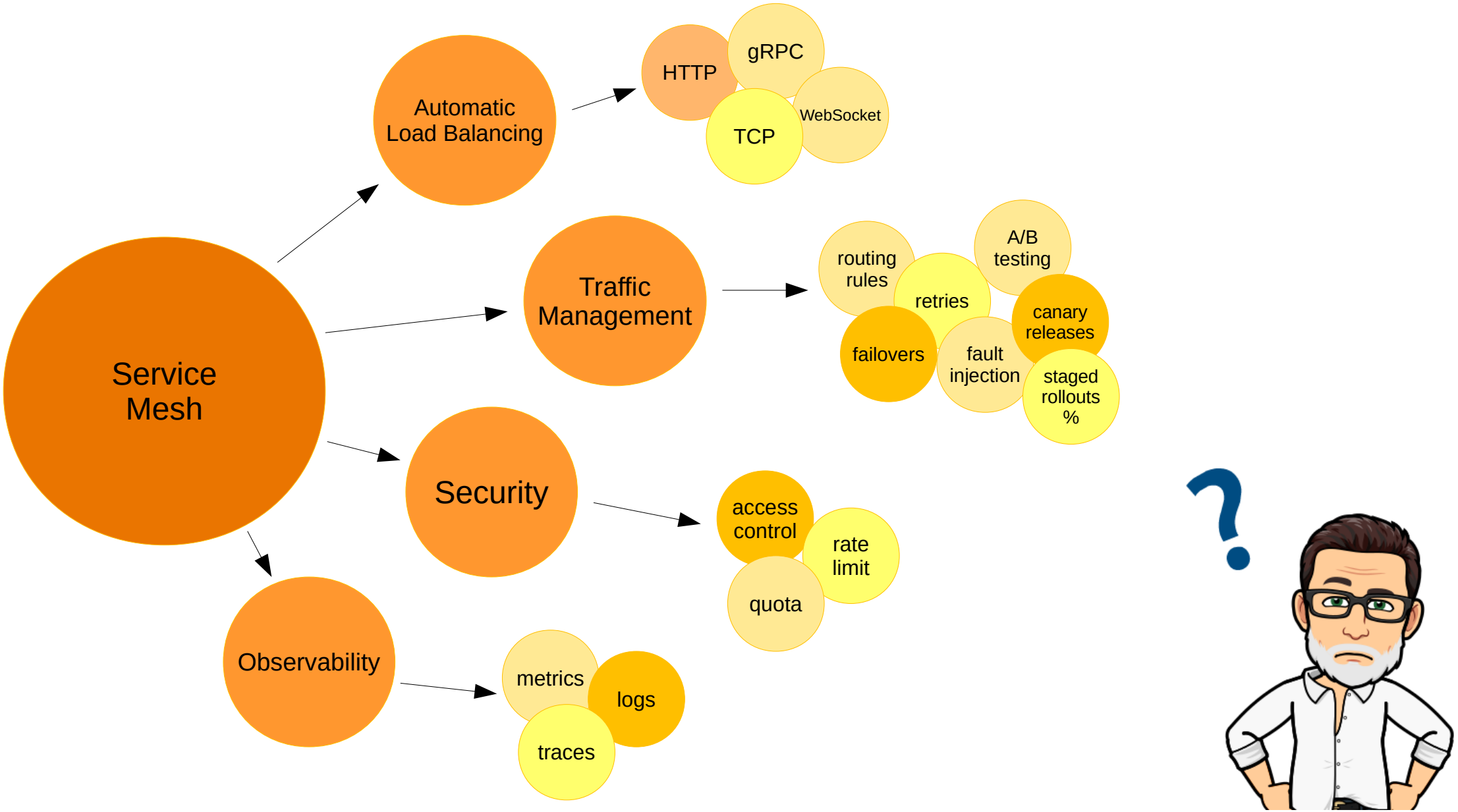
LOGICALIS

Service Mesh “Workshop”

12/07/19

Presented by Fabio De Santi

Contents

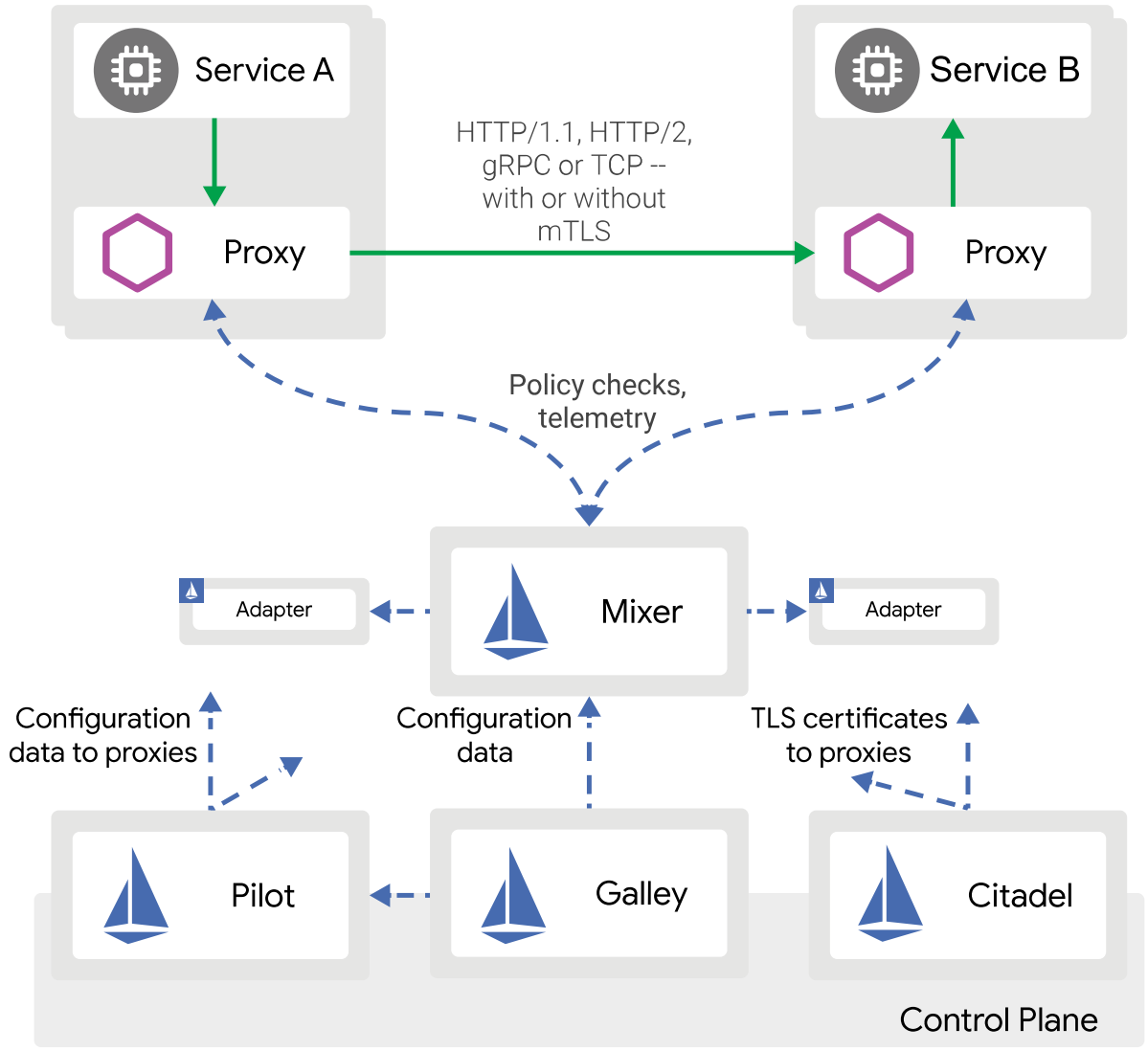


Service Mess? Micro-what??

LOGICALIS

“Do one thing and do it well.”

Unix Philosophy



Envoy

Load balance, HTTP/2 & gRPC proxy, circuit breakers, health checks, staged rollouts, fault injection, metrics

Mixer

Access control, usage policies, telemetry data collection

Pilot

Service discovery, traffic management, cpoordination

Citadel

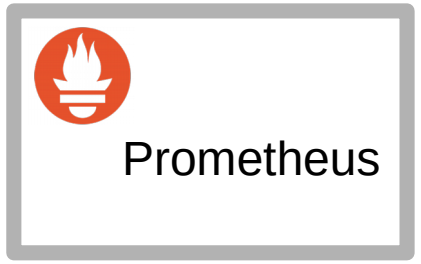
Service-to-service & pod-to-pod authentication and credential management

Galley

Manage configuration distribution

Istio - "core" components

LOGICALIS



Istio – tooling for free

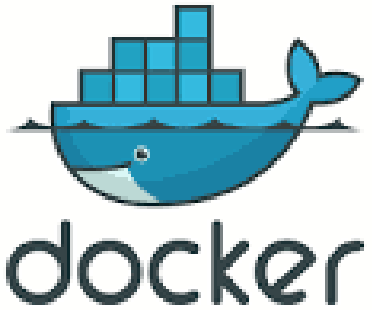
LOGICALIS

Demo Use Case

Demo scenario

LOGICALIS

minikube



let engine = ( ||  || );

Requirements

LOGICALIS

for minikube:
open dashboard & tunnel!

Start a kubernetes ‘cluster’

```
$ minikube start --memory=8192 --cpus=4 --kubernetes-version=v1.13.1 \
    --extra-config=controller-manager.cluster-signing-cert-file="/var/lib/minikube/certs/ca.crt" \
    --extra-config=controller-manager.cluster-signing-key-file="/var/lib/minikube/certs/ca.key" \
    --vm-driver=virtualbox
```

Install Istio

```
$ curl -L https://git.io/getLatestIstio | ISTIO_VERSION=1.1.7 sh -
$ cd istio-1.1.7
$ export PATH=$PWD/bin:$PATH
```

Install all Istio CRD *(custom resource definitions)*

```
$ for i in install/kubernetes/helm/istio-init/files/crd*yaml; do kubectl apply -f $i; done
```

Apply demo profile variant: permissive mutual TLS

```
$ kubectl apply -f install/kubernetes/istio-demo-auth.yaml
```

Verify

```
$ kubectl get svc -n istio-system
```

Setup Helm

```
$ curl -LO https://git.io/get_helm.sh
$ chmod 700 get_helm.sh
$ ./get_helm.sh

$ helm init --history-max 200

$ helm repo update
```

Create demo namespaces

```
$ kubectl apply -f demo-namespaces.yml
```

Enable automatic *sidecar* injection for microservices

```
$ kubectl label namespace development istio-injection=enabled
```

`demo-namespaces.yml`

```
apiVersion: v1
kind: Namespace
metadata:
  name: dev-commons
  labels:
    name: dev-commons
---
apiVersion: v1
kind: Namespace
metadata:
  name: development
  labels:
    name: development
```


Some more prep...

Add incubator repository for Helm

```
$ helm repo add incubator http://storage.googleapis.com/kubernetes-charts-incubator
$ helm repo update
```

Install Kafka and zookeeper clusters

```
$ helm install --namespace dev-commons --name kafka --set zookeeper.storage=1Gi \
     --set persistence.size=1Gi incubator/kafka
```

https://github.com/helm/charts/tree/master/incubator/kafka

Install

```
$ helm install --namespace dev-commons --set persistentVolume.size=2Gi --set hardAntiAffinity=false \
         --name redis stable/redis-ha
```

https://github.com/helm/charts/tree/master/stable/redis

LOGICALIS

Install

```
$ helm install --namespace dev-commons --set persistentVolume.size=2Gi --name rabbitmq stable/rabbitmq-ha
```

https://github.com/helm/charts/tree/master/stable/rabbitmq-ha

Inspect the Ingress controller

```
$ kubectl get svc istio-ingressgateway -n istio-system -o yaml
```

Port forward for Ingress

```
$ kubectl port-forward $(kubectl get pod -n istio-system -l app=istio-ingressgateway \
    -o jsonpath='{.items[0].metadata.name}') -n istio-system 15000
```

Grafana

```
$ kubectl port-forward $(kubectl get pod -n istio-system -l app=grafana \
    -o jsonpath='{.items[0].metadata.name}') -n istio-system 3000
```

Kiali

```
$ kubectl port-forward $(kubectl get pod -n istio-system -l app=kiali \
    -o jsonpath='{.items[0].metadata.name}') -n istio-system 20001
```

Jaeger

```
$ kubectl port-forward $(kubectl get pod -n istio-system -l app=jaeger \
    -o jsonpath='{.items[0].metadata.name}') -n istio-system 16686
```


Kubernetes port forwarding

Build a new service

```
$ cd request-api-project
$ mvn package
```

Push to Docker Hub (or any other repo)

```
$ docker images | grep sync
$ docker tag <hash> fsanti68/sync-service:latest
$ docker push fsanti68/sync-service:1.0
$ docker push fsanti68/sync-service:latest
```

Deploy the service to Kubernetes

```
$ cd deploy
$ kubectl apply -f sync-api-deploy.yml
```

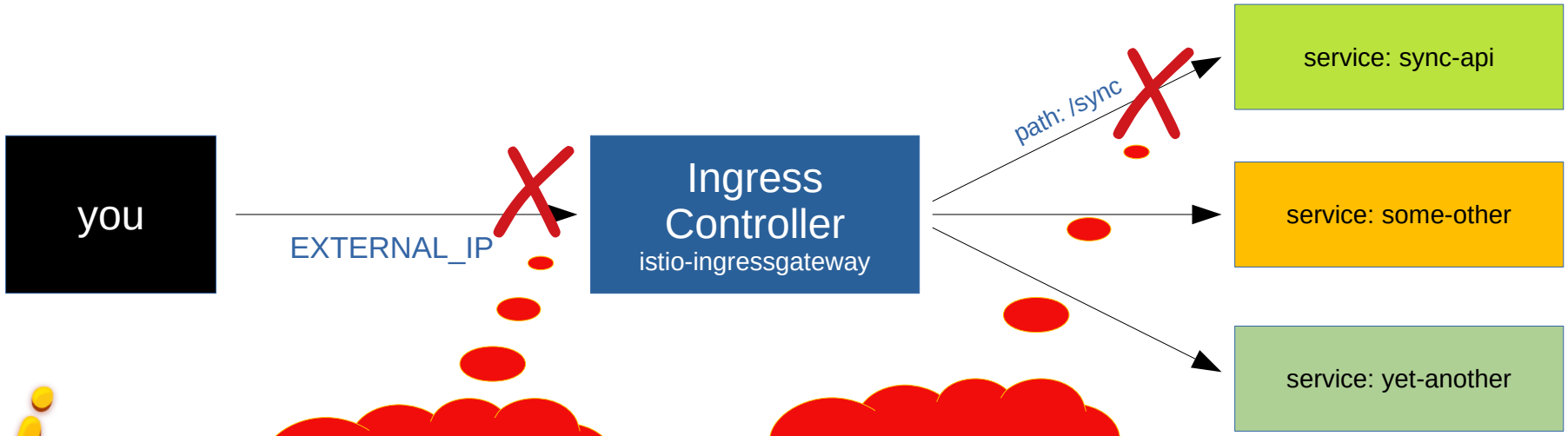
Obtain the external IP of Istio Ingress GW

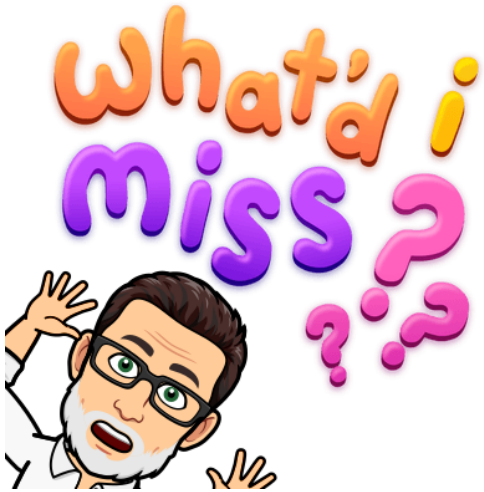
```
$ EXTERNAL_IP=$(kubectl get svc -n istio-system -l app=istio-ingressgateway \
     -o jsonpath='{.items[0].status.loadBalancer.ingress[0].ip}')
```

Deploy services

Test our new service

```
$ curl http://$EXTERNAL_IP/sync/echo/Supercalifragilisticexpialidocious
```





LOGICALIS

Create a http gateway

(allow external requests)

```
$ kubectl apply -f http-gateway.yml
```

http-gateway.yml

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: http-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
```

Create a virtual service

(route requests to service)

```
$ kubectl apply -f development-virtualservices.yml
```

development-virtualservices.yml

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: development-services
spec:
  hosts:
  - "*"
  gateways:
  - http-gateway
  http:
  - match:
    - uri:
        prefix: /sync
    route:
    - destination:
        host: sync-api
        port:
          number: 8080
```

Test (*again*) our new service

```
$ curl http://$EXTERNAL_IP/sync/echo/Supercalifragilisticexpialidocious
```

Get configuration from k8s:

- **env**

any-deploy.yml

```
…
env:
    - name: WHATAMI
      value: "Hard-coded config entry"
…
```

OR

- **ConfigMap**

clever-deploy.yml

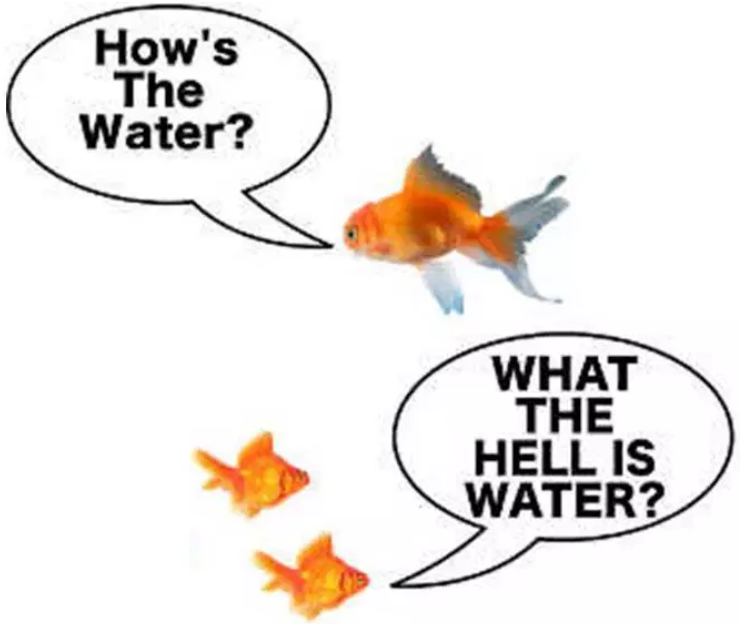
```
…
env:
    - name: WHATAMI
      valueFrom:
        configMapKeyRef:
          name: whatami
          key: WHATAMI
…
```

+

my-configmap.yml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: env-configmap
data:
  WHATAMI: Nice, elegant and maintanable config entry
```

Sharable!!!!

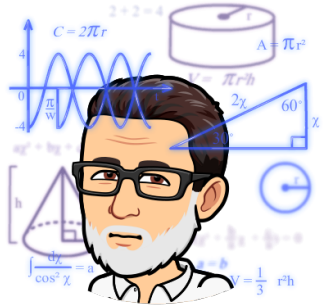


LOGICALIS

```
$ kubectl port-forward $(kubectl get pod -n istio-system -l app=grafana \
    -o jsonpath='{.items[0].metadata.name}') -n istio-system 3000

OU

$ kubectl port-forward -n istio-system svc/grafana 3000
```



```
$ kubectl apply -f grafana-vs.yml
$ kubectl apply -f prometheus-vs.yml
$ kubectl apply -f kiali-vs.yml
$ kubectl apply -f tracing-vs.yml
$ EXTERNAL_IP=$(kubectl get svc -n istio-system -l app=istio-ingressgateway \
     -o jsonpath='{.items[0].status.loadBalancer.ingress[0].ip}')
```

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: grafana-gateway
  namespace: istio-system
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 15031
      name: http-grafana
      protocol: HTTP
    hosts:
    - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: grafana-vs
  namespace: istio-system
spec:
  hosts:
  - "*"
  gateways:
  - grafana-gateway
  http:
  - match:
    - port: 15031
    route:
    - destination:
        host: grafana
        port:
          number: 3000
```

http://k8scluster:15030/

http://k8scluster:15031/

http://k8scluster:15032/

http://k8scluster:15029/kiali/

Monitoring – Metrics, Logs, Tracing – Observe!

LOGICALIS

Routing

- Canary
- Fault injection

Security

- Rate limit
- Quota

AND THAT'S ALL I HAVE

TO SAY ABOUT THAT.

LOGICALIS

LOGICALIS

Thank you

Contact details
Fabio De Santi
Fabio.Santi@la.logicalis.com