Manage ONAP Microservices with Istio Service Mesh-Mutual TLS Authentication Enabled

Introduction

In Casablanca release, MSB project is integrating Istio Service Mesh with ONAP to manage ONAP microservices. Istio Service Mesh is a dedicated infrastructure layer to connect, manage and secure microservices, which brings the below benefits:

- Stability and Reliability: Reliable communication with retries and circuit breaker
- Security: Secured communication with TLS
- Performance: Latency aware load balancing with warm cache
- Observability: Metrics measurement and distributed tracing without instrumenting application
- Manageability: Routing rule and rate limiting enforcement
- Testability: Fault injection to test resilience of the services

Installation

Download installation scripts from ONAP Gerrit:

```
git clone https://gerrit.onap.org/r/msb/service-mesh
```

Kubernetes Master

We need Kubernetes1.9 or newer to enable automatic sidecar injection, so we don't have to modify every individual ONAP kubernetes yaml deployment files to add the sidecar container, which would be inconvenient.

Istio leverages the webhook feature of Kubernetes to automatically inject an Envoy sidecar to each Pod. Kubernetes API server will call the Istio sidecar injection webhook when it receives a request to create a Pod resource, the webhook adds an Envoy sidecar container to the Pod, then the modified Pod resource is stored into etcd.

Webhook and other needed features have already been configured in the install scripts to enable Istio sidecar injection.

Create the Kubernetes master by running this script:

```
cd service-mesh/install/
./1_install_k8s_master.sh
```

This script will create a Kubernetes master node with Kubeadm and install calico network plugin. Some other needed tools such as Docker, Kubectl and Helm will be installed as well.

From the output of the script, you should see a command on how to join a node to the created Kubernets cluster. Note that this is an example, the token and cert-hash of your installation will be different, please copy & paste the command to somewhere, we will need it later.

```
You can now join any number of machines by running the following on each node
as root:

  kubeadm join 10.12.5.104:6443 --token 1x62yf.60ys5p2iw13tx2t8 --discovery-token-ca-cert-hash sha256:
f06628c7cee002b262e69f3f9efadf47bdec125e19606ebff743a3e514a8383b
```

Kubernetes worker Node

Log in the worker node machine, run this script to create a kubernetes worker node:

```
./2_install_k8s_minion.sh
```

You can now join this machines by running "kubeadmin join" command as root:

```
sudo kubeadm join 10.12.5.104:6443 --token 1x62yf.60ys5p2iw13tx2t8 --discovery-token-ca-cert-hash sha256:
f06628c7cee002b262e69f3f9efadf47bdec125e19606ebff743a3e514a8383b
```

Please note that this is just an example, please refer to the output of the "kubeamin init" when creating the k8s master for the exact command to use in your k8s cluster.

If you would like to get kubectl talk to your k8s master, you need to copy the administrator kubeconfig file from your master to your workstation like this:

```
scp root@<master ip>:/etc/kubernetes/admin.conf .
kubectl --kubeconfig ./admin.conf get nodes
```

or you can manually copy the content of this file to ~/.kube/conf if scp can't be used due to security reason.

Istio Control Plane with Mutual TLS

Install Istio by running this script:

```
./ 3_install_istio_with_auth.sh
```

This script installs the followings Istio components:

- Install Istioctl command line tool in the /usr/bin directory
- Install Istio control plane components, including Pilot, Citadel, Mixer
- Install addons including servicegraph, Prometheus, Grafana, jaeger

Confirm Istio was installed:

```
kubectl get svc -n istio-system
NAME                       TYPE           CLUSTER-IP       EXTERNAL-IP   PORT
(S)                                                                      AGE
grafana                    NodePort       10.109.190.71    <none>        3000:30300
/TCP                                                                      20m
istio-citadel              ClusterIP      10.106.185.181   <none>        8060/TCP,9093
/TCP                                                                   20m
istio-egressgateway        ClusterIP      10.102.224.133   <none>        80/TCP,443
/TCP                                                                      20m
istio-ingressgateway       LoadBalancer   10.100.168.32    <pending>     80:31380/TCP,443:31390/TCP,31400:31400
/TCP                            20m
istio-pilot                ClusterIP      10.101.64.153    <none>        15003/TCP,15005/TCP,15007/TCP,15010
/TCP,15011/TCP,8080/TCP,9093/TCP   20m
istio-policy               ClusterIP      10.104.11.162    <none>        9091/TCP,15004/TCP,9093
/TCP                                          20m
istio-sidecar-injector     ClusterIP      10.100.229.40    <none>        443
/TCP                                                                        20m
istio-statsd-prom-bridge   ClusterIP      10.107.27.91     <none>        9102/TCP,9125
/UDP                                                                   20m
istio-telemetry            ClusterIP      10.101.153.114   <none>        9091/TCP,15004/TCP,9093/TCP,42422
/TCP                                  20m
prometheus                 ClusterIP      10.103.0.205     <none>        9090
/TCP                                                                   20m
servicegraph               NodePort       10.106.49.168    <none>        8088:30088
/TCP                                                                      20m
tracing                    LoadBalancer   10.100.158.236   <pending>     80:30188
/TCP                                                                         20m
zipkin                     NodePort       10.96.164.255    <none>        9411:30411
/TCP                                                                      20m
```

Sidecar Injection

In the transition phase, the Istio sidecar injector policy is configured as "disabled" when installing Istio. So the sidecar injector will not inject the sidecar into pods by default. Add the `sidecar.istio.io/inject annotation` with value `true` to the pod template spec to enable injection.

Example:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ include "common.fullname" . }}
  namespace: {{ include "common.namespace" . }}
  labels:
    app: {{ include "common.name" . }}
    chart: {{ .Chart.Name }}-{{ .Chart.Version | replace "+" "_" }}
    release: {{ .Release.Name }}
    heritage: {{ .Release.Service }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: multicloud-vio
  template:
    metadata:
      labels:
        app: {{ include "common.name" . }}
        release: {{ .Release.Name }}
      name: {{ include "common.name" . }}
      annotations:
        sidecar.istio.io/inject: "{{.Values.istioSidecar}}"
```

Note: when all ONAP projects are ready for Istio integration, the Istio sidecar injector policy could be configured as "enabled", then the annotation in the pod will not be necessary any more.

Enable Istio sidecar injection webhook.

```
kubectl create namespace onap
kubectl label namespace onap istio-injection=enabled
```

Confirm that auto sidecar injection has been enabled on onap namespace.

```
kubectl get namespace -L istio-injection
NAME           STATUS    AGE       ISTIO-INJECTION
default        Active    20m
istio-system   Active    10m
kube-public    Active    20m
kube-system    Active    20m
onap           Active    8s        enabled
```

Start a local helm repository server and add it to helm repository list:

```
helm serve &
helm repo add local http://127.0.0.1:8879
```

Download OOM Gerrit repository and build the helm charts.

```
git clone -b beijing http://gerrit.onap.org/r/oom
cd oom/kubernetes
make all
```

Confirm that ONAP charts have been successfully created.

```
helm search onap
NAME                    CHART VERSION   APP VERSION     DESCRIPTION
local/onap              2.0.0           beijing         Open Network Automation Platform (ONAP)
local/aaf               2.0.0                           ONAP Application Authorization Framework
local/aai               2.0.0                           ONAP Active and Available Inventory
local/clamp             2.0.0                           ONAP Clamp
local/cli               2.0.0                           ONAP Command Line Interface
local/consul            2.0.0                           ONAP Consul Agent
local/dcaegen2          2.0.0                           ONAP DCAE Gen2
local/dmaap             2.0.0                           ONAP DMaaP components
local/esr               2.0.0                           ONAP External System Register
local/log               2.0.0                           ONAP Logging ElasticStack
local/msb               2.0.0                           ONAP MicroServices Bus
local/multicloud        2.0.0                           ONAP multicloud broker
local/nbi               2.0.0                           ONAP Northbound Interface
local/oof               2.0.0                           ONAP Optimization Framework
local/policy            2.0.0                           ONAP Policy Administration Point
local/portal            2.0.0                           ONAP Web Portal
local/postgres          2.0.0                           ONAP Postgres Server
local/robot             2.0.0                           A helm Chart for kubernetes-ONAP Robot
local/sdnc-prom         2.0.0                           ONAP SDNC Policy Driven Ownership Management
local/sniro-emulator    2.0.0                           ONAP Mock Sniro Emulator
local/so                2.0.0                           ONAP Service Orchestrator
local/uui               2.0.0                           ONAP uui
local/vfc               2.0.0                           ONAP Virtual Function Controller (VF-C)
local/vid               2.0.0                           ONAP Virtual Infrastructure Deployment
local/vnfsdk            2.0.0                           ONAP VNF SDK
```

Install local/onap chart. Local/onap chart will do some initialization setup which is needed for onap components, such as creating service accounts.

```
cd oom/kubernetes
helm install local/onap -n common --namespace onap -f onap/resources/environments/disable-allcharts.yaml
```

In Casablanca, MSB project is working with VF-C and MultiCloud as pilot projects, we would like to roll out it to the other ONAP projects after verifying the integration and Istio features.

```
helm install local/msb -n msb --namespace onap
 helm install local/multicloud -n multicloud --namespace onap  --set liveness.enabled=false,multicloud-ocata.
liveness.enabled=false,multicloud-vio.liveness.enabled=false,multicloud-windriver.liveness.enabled=false
helm install local/multicloud -n multicloud --namespace onap
```

Note:

- The current version of Istio mutual TLS authentication can't work with kubernetes liveness probe, Istio is working on a long-term fix to solve this problem. A simple workaround for the time being is to disable liveness probe by passing a 'liveness.enabled=false' value to helm install command.
- You can also install other ONAP projects with helm install if they are needed. But Istio sidecar will not be injected to their Pods by default.

Confirm that ONAP microservices have been started

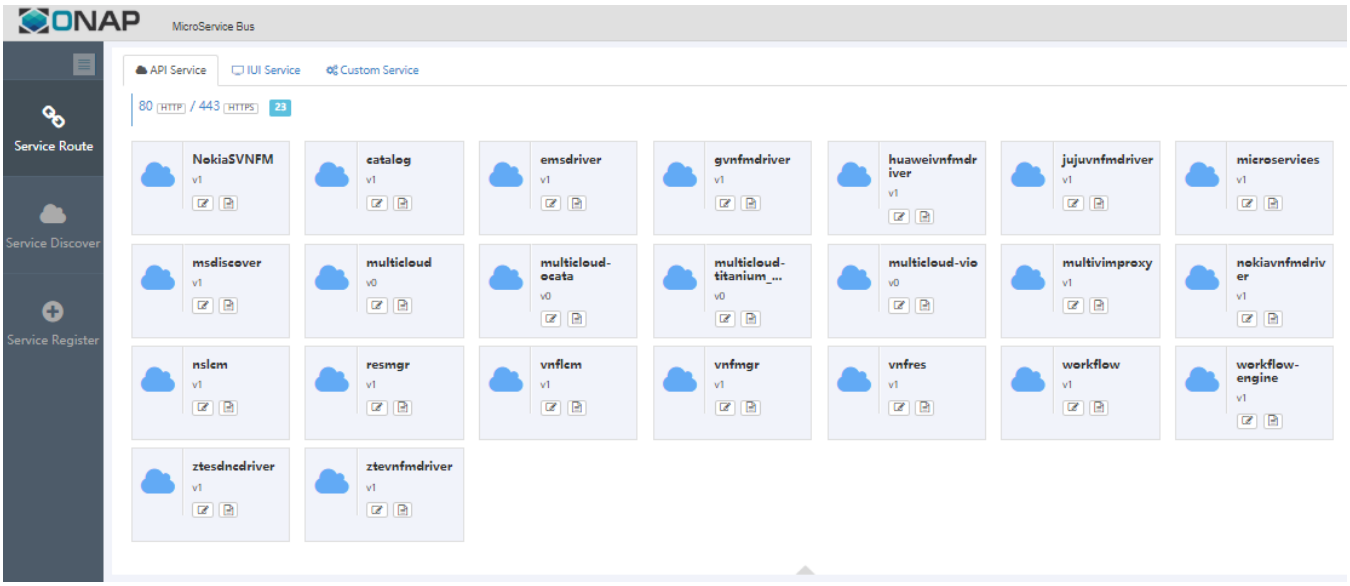
```
kubectl get all -n onap
NAME READY STATUS RESTARTS AGE
pod/msb-kube2msb-77ccb675dd-rhfn7 1/1 Running 0 3h
pod/msb-msb-consul-646987f5cf-qms5v 2/2 Running 0 3h
pod/msb-msb-discovery-7647f6476f-cl6xw 3/3 Running 0 3h
pod/msb-msb-eag-d678c65d6-fmfn6 3/3 Running 0 3h
pod/msb-msb-iag-647d5f998c-dc766 3/3 Running 0 3h
pod/multicloud-multicloud-5679bd9876-tzxzw 2/2 Running 0 1h
pod/multicloud-multicloud-ocata-774579596-f7smf 3/3 Running 0 1h
pod/multicloud-multicloud-vio-8c7dbc8d5-lfcw6 3/3 Running 0 1h
pod/multicloud-multicloud-windriver-85b595675d-5vx45 3/3 Running 0 1h
pod/vfc-vfc-catalog-79764dfd8f-rkx6f 2/2 Running 1 2d
pod/vfc-vfc-ems-driver-75bc68b946-6r6r6 1/1 Running 1 2d
pod/vfc-vfc-generic-vnfm-driver-69bf778bfd-pscjn 2/2 Running 0 2d
pod/vfc-vfc-huawei-vnfm-driver-8574569f4c-8jwc4 2/2 Running 1 2d
pod/vfc-vfc-juju-vnfm-driver-6dfd876bb8-bh7dq 2/2 Running 0 2d
pod/vfc-vfc-multivim-proxy-58c7bd47dc-7qdtd 1/1 Running 0 2d
pod/vfc-vfc-nokia-v2vnfm-driver-7b77c469bd-krfrw 1/1 Running 0 2d
pod/vfc-vfc-nokia-vnfm-driver-98fbdb5b5-p9zqw 2/2 Running 0 2d
pod/vfc-vfc-nslcm-74956bb876-v9kbt 2/2 Running 0 2d
pod/vfc-vfc-resmgr-57dc4c98b5-dzp7f 2/2 Running 0 2d
pod/vfc-vfc-vnflcm-6f9dc7df44-hncf4 2/2 Running 1 2d
pod/vfc-vfc-vnfmgr-5585c688c6-7qrnp 2/2 Running 0 2d
pod/vfc-vfc-vnfres-54bc985599-9zkqn 2/2 Running 0 2d
pod/vfc-vfc-workflow-6db56f95b9-np8tg 1/1 Running 1 2d
pod/vfc-vfc-workflow-engine-7fb49fd974-kcb8q 1/1 Running 1 2d
pod/vfc-vfc-zte-sdnc-driver-585d449797-87nhp 1/1 Running 0 2d
pod/vfc-vfc-zte-vnfm-driver-59d4756fbc-rpn9v 2/2 Running 0 2d

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/msb-consul NodePort 10.96.255.198 <none> 8500:30285/TCP 3h
service/msb-discovery NodePort 10.105.163.81 <none> 10081:30281/TCP 3h
service/msb-eag NodePort 10.100.221.66 <none> 80:30282/TCP,443:30284/TCP 3h
service/msb-iag NodePort 10.96.179.117 <none> 80:30280/TCP,443:30283/TCP 3h
service/multicloud NodePort 10.102.72.237 <none> 9001:30291/TCP 1h
service/multicloud-ocata NodePort 10.99.131.129 <none> 9006:30293/TCP 1h
service/multicloud-vio NodePort 10.111.175.58 <none> 9004:30292/TCP 1h
service/multicloud-windriver NodePort 10.110.92.61 <none> 9005:30294/TCP 1h
service/vfc-catalog ClusterIP 10.99.98.115 <none> 8806/TCP 2d
service/vfc-ems-driver ClusterIP 10.96.189.14 <none> 8206/TCP 2d
service/vfc-generic-vnfm-driver ClusterIP 10.109.48.184 <none> 8484/TCP 2d
service/vfc-huawei-vnfm-driver ClusterIP 10.104.208.38 <none> 8482/TCP,8483/TCP 2d
service/vfc-juju-vnfm-driver ClusterIP 10.96.182.14 <none> 8483/TCP 2d
service/vfc-multivim-proxy ClusterIP 10.107.106.216 <none> 8481/TCP 2d
service/vfc-nokia-v2vnfm-driver ClusterIP 10.107.12.32 <none> 8089/TCP 2d
service/vfc-nokia-vnfm-driver ClusterIP 10.102.179.150 <none> 8486/TCP 2d
service/vfc-nslcm ClusterIP 10.106.43.164 <none> 8403/TCP 2d
service/vfc-resmgr ClusterIP 10.98.174.184 <none> 8480/TCP 2d
service/vfc-vnflcm ClusterIP 10.108.132.123 <none> 8801/TCP 2d
service/vfc-vnfmgr ClusterIP 10.108.59.102 <none> 8803/TCP 2d
service/vfc-vnfres ClusterIP 10.111.85.161 <none> 8802/TCP 2d
service/vfc-workflow ClusterIP 10.97.184.206 <none> 10550/TCP 2d
service/vfc-workflow-engine ClusterIP 10.109.175.61 <none> 8080/TCP 2
service/vfc-zte-sdnc-driver ClusterIP 10.103.94.142 <none> 8411/TCP 2d
service/vfc-zte-vnfm-driver ClusterIP 10.108.146.237 <none> 8410/TCP 2d
```

Create an Istio Gateway so we can access the MSB portal out of the Mesh

```
cd service-mesh/install/
kubectl apply -f msb-istio-gateway.yaml -n onap
```

Now you can open the MSB portal http://${INGRESS_IP}:${INGRESS_PORT}/msb in the browser to see all the registered services.

Note: INGRESS_IP and INGRESS_PORT can be found by executing this command 'kubectl get svc istio-ingressgateway -n istio-system'



Explore Istio Features

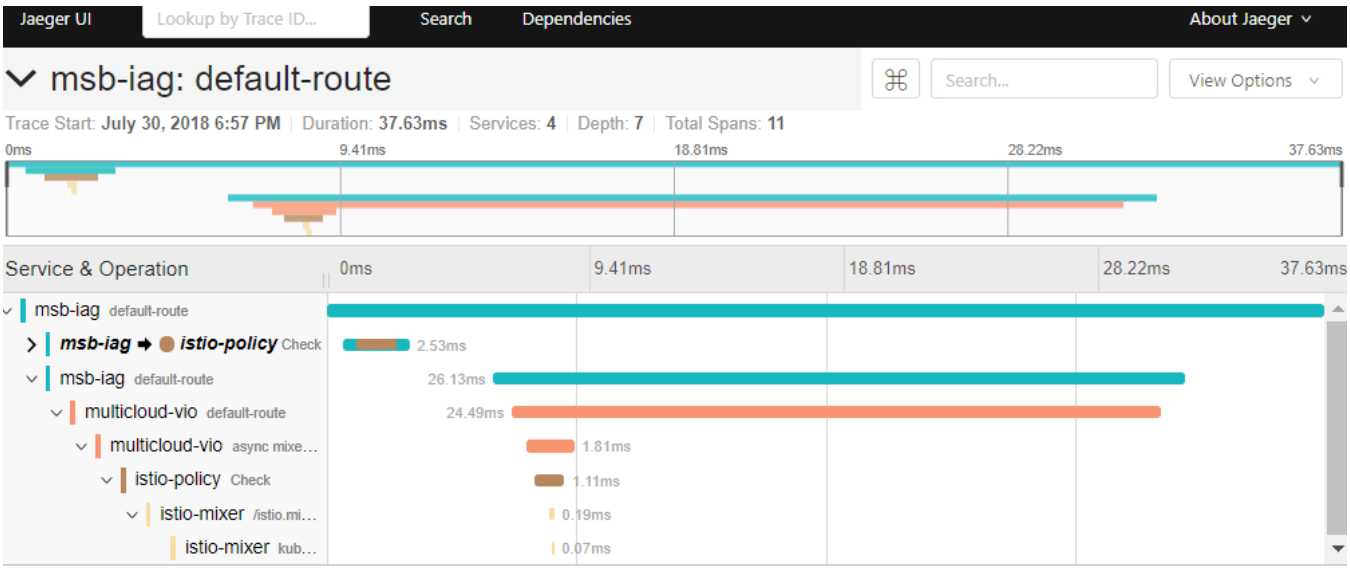
Distributed Tracing

First, let's generate some traffics in the application, access the following URLs with curl command or open them in the browser

http://${INGRESS_IP}:${INGRESS_PORT}/api/multicloud-vio/v0/swagger.json

http://${INGRESS_IP}:${INGRESS_PORT}/api/multicloud-ocata/v0/swagger.json

Then open your browser at http://tracing_node_ip:tracing_node_port/, you should see something similar to the following:



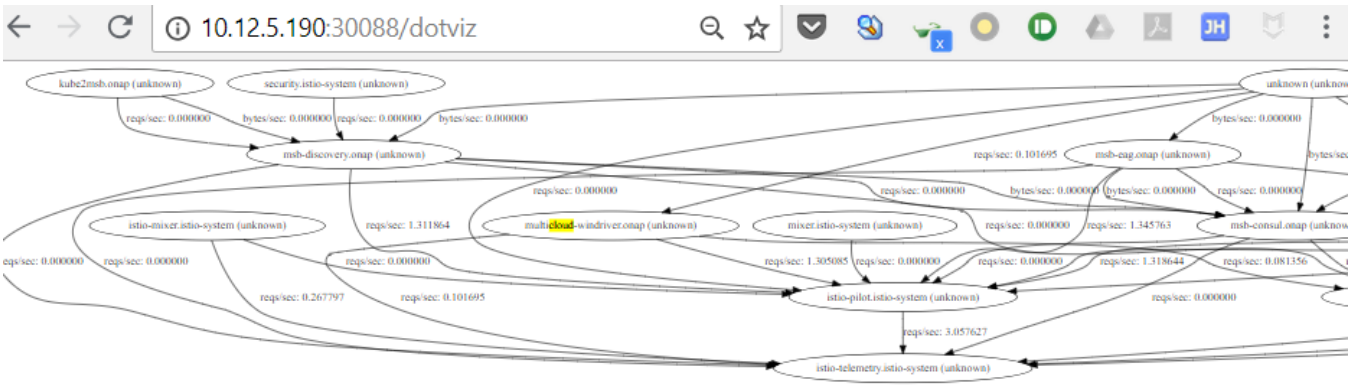
Note

- Tracing_node_port can be found by 'kubctl get svc -n istio-system'.
- ONAP microservices need to propagate the appropriate HTTP headers so that when the proxies send span information, the spans can be correlated correctly into a single trace.

Service Graph

Istio provides a Servicegraph service which generates and visualizes graph representations of the services in the mesh.

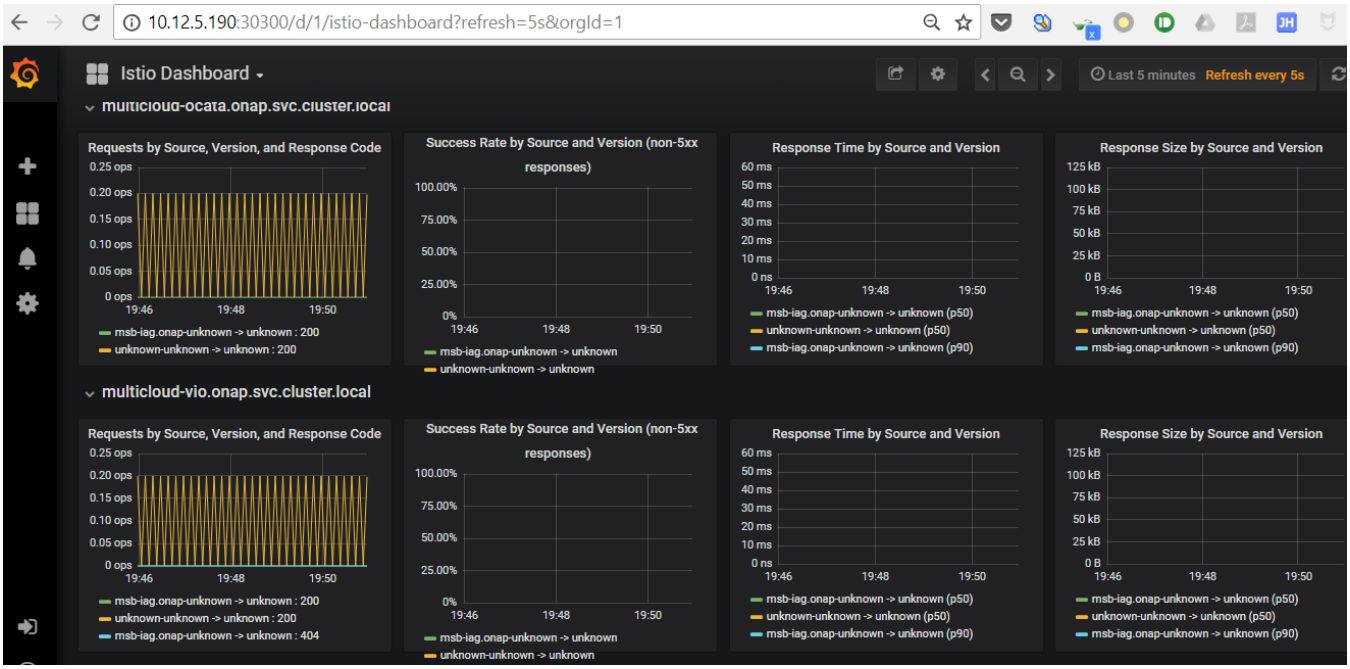
Open your browser at http://node_ip:30088/dotviz or http://node_ip:30088/force/forcegraph.html, you should see the service graph:



Metrics Visualization

Istio automatically gathers telemetry for services in a mesh. A Prometheus adapter is plugged into Mixer to serve the generated metric data.
A Grafana addon is pre-configured with a Prometheus data source and has an Istio dashboard installed for the metric visualization.

Open your browser at http://node_ip:30300, you should see the Grafana Istio dashboard:



Authorization(RBAC)

Istio authorization is disabled by default, running the following command to enable it for onap namespace:

```
cd /service-mesh/install
kubectl apply -f enable-istio-rbac.yaml
```

Point your browser at the msb portal or multicloud swagger file:

http://${INGRESS_IP}:${INGRESS_PORT}/msb

http://${INGRESS_IP}:${INGRESS_PORT}/api/multicloud-vio/v0/swagger.json

Now you should see "RBAC: access denied". This is because Istio authorization is “deny by default”, which means that you need to explicitly define access control policy to grant access to any service. Note: There may be some delays due to caching and other propagation overhead.

Running the following command to allow Istio Ingress gateway read access to onap Namespace:

```
cd /service-mesh/install
kubectl apply -f apply -f istio-ingress-rbac.yaml
```

Now if you point your browser at the msb portal (http://${INGRESS_IP}:${INGRESS_PORT}/msb). You should see the msb portal page with registered services.

Note: There may be some delays due to caching and other propagation overhead.

If you try to access http://${INGRESS_IP}:${INGRESS_PORT}/api/multicloud-vio/v0/swagger.json, you should still see "RBAC: access denied". What's happening? This request actually goes through browser->Istio-ingress->MSB->multicloud, even now Istio-ingress can access MSB, MSB is not allowed to access multicloud. So we need to create another RBAC rule to grant the access permission of multicloud to MSB.

Running the following command to grant access of multicloud to MSB:

```
cd /service-mesh/install
kubectl apply -f msb-rbac.yaml
```

Try to access http://${INGRESS_IP}:${INGRESS_PORT}/api/multicloud-vio/v0/swagger.json again, you should be able to see the swagger file return from multicloud microservice.

Note: There may be some delays due to caching and other propagation overhead.

Service Mesh Migration

Without Istio Authentication and Authorization

ONAP can be easily integrated with Istio service mesh if Istio Auth is disabled. In that case, ONAP can leverage the traffic management, telemetry and policies capabilities of Istio to connect, control and observe ONAP microservies, but without Mutual TLS authentication and authorization.

Though ONAP services can talk to each other within the mesh, to maximize the benefits brought by Istio, we still need to make little compatible changes to the existing services:

Service Port Name

The port names must be of the form `protocol-suffix` with http, http2, grpc, mongo, or redis as the `protocol` in order to take advantage of Istio's routing features.

For example, `name: http2-foo` or `name: http` are valid port names, but `name: http2foo` is not. If the port name does not begin with a recognized prefix or if the port is unnamed, traffic on the port will be treated as plain TCP traffic (unless the port explicitly uses Protocol: UDP to signify a UDP port).

```
kubectl describe svc aai -n onap
Name:                     aai
Namespace:                onap
Labels:                   app=aai
                          chart=aai-2.0.0
                          heritage=Tiller
                          release=aai1
Annotations:              <none>
Selector:                 app=aai
Type:                     NodePort
IP:                       10.96.29.203
Port:                     http-aai  8080/TCP
---omitted for brevity
```

Propagate Http Header for Distributed Tracing

Istio uses HTTP headers to record the request tracing information across multiple spans. Although Istio proxies are able to automatically send all the spans to Mixer, they need some hints to tie together the individual spans to get the entire trace.

To do this, ONAP microservies needs to collect and propagate the following headers from the incoming request to any outgoing requests:

- `x-request-id`
- `x-b3-traceid`
- `x-b3-spanid`
- `x-b3-parentspanid`
- `x-b3-sampled`

- x-b3-flags
- x-ot-span-context

With Istio Authentication and Authorization

In addition to the port name format and http header propagation, the followings need to be done to leverage Istio auth.

Liveness probe

Mutual TLS can't work with 8Shttp/tcp liveness probe. If mutual TLS is enabled, http and tcp health checks from the kubelet will not work since they do not have Istio-issued certs. The workaround is using liveness command instead or disabling http and tcp liveness probe for the time being.

Allow both Mutual TLS and Plain Traffic

During the migration, we can use “PERMISSIVE”
mode of Istio Auth policy to allow both TLS and plain traffic. After migration is done, the mode can be switched to "STRICT" mode so only TLS traffics are permitted to access services.

```
cat <<EOF | kubectl apply -n onap -f -
apiVersion: "authentication.istio.io/v1alpha1"
kind: "Policy"
metadata:
  name: "default"
  namespace: onap
spec:
  peers:
  - mtls:
      mode: PERMISSIVE
EOF
```

In that case, the RBAC should be set to allow all users, including the unauthenticated users, to access the services.

```
 cat <<EOF | kubectl apply -n onap -f -
apiVersion: "rbac.istio.io/v1alpha1"
kind: ServiceRole
metadata:
  name: onap-default
  namespace: onap
spec:
  rules:
  - services: ["*"]
    methods: ["*"]
---
apiVersion: "rbac.istio.io/v1alpha1"
kind: ServiceRoleBinding
metadata:
  name: bind-service-default
  namespace: onap
spec:
  subjects:
  - user: "*"
  roleRef:
    kind: ServiceRole
    name: "onap-default"
EOF
```

By this approach, ONAP can be smoothly migrated to Istio with auth enabled. After every ONAP microservice adopts Istio auth, then we can set the authentication to "STRICT" mode and enforce strict access control per the needs of each service.

What's the next? we will provide a user-friendly Istio UI to manage Istio rules and policies. Comment here to leave your thoughts or join our weekly project meeting if you're interested.