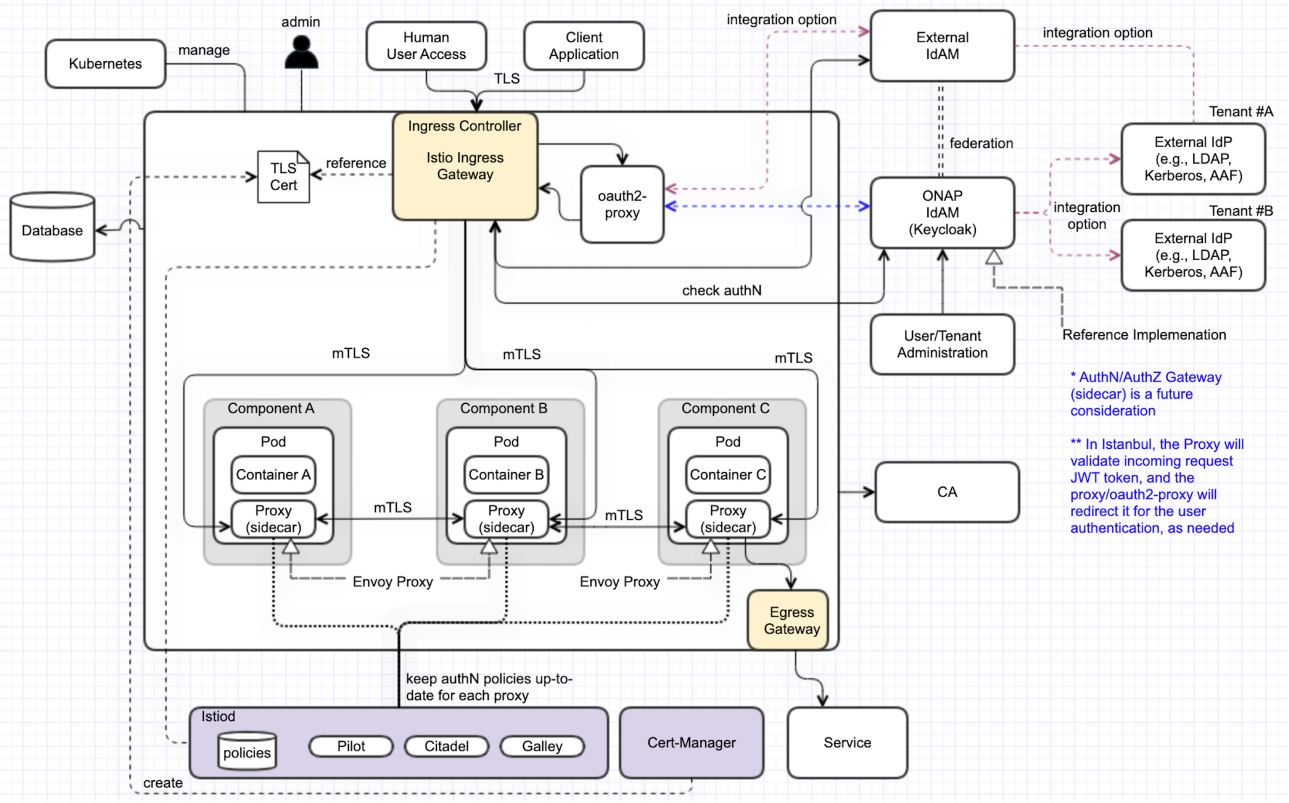
ONAP on ServiceMesh setup guide

Background

After the SM PoC (Guilin) we are focussing in deploying SM in Kohn using Istio as SM framework in ONAP.
Target picture is described in SECCOM page:
ONAP Next Generation Security & Logging Architecture



Cluster Preparation

During the setup of the K8S Cluster the Istio resources need to be dinstalled.
As basis in Kohn we use the following platform versions:

- helm_release: v3.8.2
- kubernetes_release: v1.23.8
- istio_release: 1.17.0
- Cert-Manager: 1.5.4
- Strimzi-Operator: 0.30.0

More information

Istio Best Practices:
https://docs.solo.io/gloo-mesh-enterprise/latest/setup/prod/namespaces/

Install Istio

Source: https://istio.io/latest/docs/setup/install/helm/

Istio basics

1. Configure the Helm repository:

```
$ helm repo add istio https://istio-release.storage.googleapis.com/charts
$ helm repo update
```

2. Create a namespace for "mesh-level" configurations

```
$ kubectl create namespace istio-config
```

3. Create a namespace istio-system for Istio components:

```
$ kubectl create namespace istio-system
```

4. Install the **Istio Base** chart which contains cluster-wide resources used by the Istio control plane:

```
$ helm upgrade -i istio-base istio/base -n istio-system --version 1.17.0
```

Install the **Istio Discovery** chart which deploys the istiod service:
(enable the variable to enforce the (sidecar) proxy startup before the container start)

Create a values-override.yaml file to override settings (required for oauth2-proxy):

```
global:
  proxy:
    # Controls if sidecar is injected at the front of the container list and blocks the start of the other
containers until the proxy is ready
    holdApplicationUntilProxyStarts: true
  #logging:
  #  level: "default:debug"
meshConfig:
  rootNamespace: istio-config
  extensionProviders:
  - name: oauth2-proxy
    envoyExtAuthzHttp:
      service: oauth2-proxy.default.svc.cluster.local
      port: 80
      timeout: 1.5s
      includeHeadersInCheck: ["authorization", "cookie"]
      headersToUpstreamOnAllow: ["x-forwarded-access-token", "authorization", "path", "x-auth-request-user", "x-
auth-request-email", "x-auth-request-access-token"]
      headersToDownstreamOnDeny: ["content-type", "set-cookie"]
pilot:
  env:
    PILOT_ENABLE_MYSQL_FILTER: true
    PILOT_HTTP10: true
```

Install Istio Discovery using the override file

```
$ helm upgrade -i istiod istio/istiod -n istio-system --version 1.17.0 --wait -f ./values-override.yaml
```

Add an EnvoyFilter for HTTP header case

When handling HTTP/1.1, Envoy will normalize the header keys to be all lowercase.
While this is compliant with the HTTP/1.1 spec, in practice this can result in issues when migrating existing systems that might rely on specific header casing.
In our case a problem was detected in the SDC client implementation, thich relies on uppercase header values.
To solve this problem in general

- we add a EnvoyFilter to keep the uppercase header in the istio-config namespace to apply for all namespaces.
- but set the context to SIDECAR_INBOUND and SIDECAR_OUTBOUND to avoid problems in the connection between Istio-Gateway and Services

1. Create a EnvoyFilter file (e.g. envoyfilter-case.yaml)

```
---
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: header-casing-inbound
  namespace: istio-config
  #annotations:
  #  argocd.argoproj.io/hook: PostSync
spec:
  configPatches:
  - applyTo: CLUSTER
    match:
      context: SIDECAR_INBOUND
    patch:
      operation: MERGE
      value:
        typed_extension_protocol_options:
          envoy.extensions.upstreams.http.v3.HttpProtocolOptions:
            '@type': type.googleapis.com/envoy.extensions.upstreams.http.v3.HttpProtocolOptions
            use_downstream_protocol_config:
              http_protocol_options:
                header_key_format:
                  stateful_formatter:
                    name: preserve_case
                    typed_config:
                      '@type': type.googleapis.com/envoy.extensions.http.header_formatters.preserve_case.
v3.PreserveCaseFormatterConfig
  - applyTo: NETWORK_FILTER
    match:
      listener:
        filterChain:
          filter:
            name: envoy.filters.network.http_connection_manager
    patch:
      operation: MERGE
      value:
        typed_config:
          '@type': type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.
HttpConnectionManager
          http_protocol_options:
            header_key_format:
              stateful_formatter:
                name: preserve_case
                typed_config:
                  '@type': type.googleapis.com/envoy.extensions.http.header_formatters.preserve_case.v3.
PreserveCaseFormatterConfig
---
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: header-casing-outbound
  namespace: istio-config
  #annotations:
  #  argocd.argoproj.io/hook: PostSync
spec:
  configPatches:
  - applyTo: CLUSTER
    match:
      context: SIDECAR_OUTBOUND
    patch:
      operation: MERGE
      value:
        typed_extension_protocol_options:
          envoy.extensions.upstreams.http.v3.HttpProtocolOptions:
            '@type': type.googleapis.com/envoy.extensions.upstreams.http.v3.HttpProtocolOptions
            use_downstream_protocol_config:
              http_protocol_options:
                header_key_format:
                  stateful_formatter:
```

```
                    name: preserve_case
                    typed_config:
                      '@type': type.googleapis.com/envoy.extensions.http.header_formatters.preserve_case.
v3.PreserveCaseFormatterConfig
  - applyTo: NETWORK_FILTER
    match:
      listener:
        filterChain:
          filter:
            name: envoy.filters.network.http_connection_manager
    patch:
      operation: MERGE
      value:
        typed_config:
          '@type': type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.
HttpConnectionManager
          http_protocol_options:
            header_key_format:
              stateful_formatter:
                name: preserve_case
                typed_config:
                  '@type': type.googleapis.com/envoy.extensions.http.header_formatters.preserve_case.v3.
PreserveCaseFormatterConfig
```

2. Apply the change to Istio

```
$ kubectl apply -f envoyfilter-case.yaml
```

Istio Ingress Gateway

1. Create a namespace istio-ingress for the Istio Ingress gateway and enable istio-injection:

```
$ kubectl create namespace istio-ingress
$ kubectl label namespace istio-ingress istio-injection=enabled
```

2. Install the **Istio Gateway** chart:

```
$ helm upgrade -i istio-ingress istio/gateway -n istio-ingress --version 1.15.1 --wait
```

(Addon required for

Install Jaeger/Kiali

Kiali Installation

see: https://kiali.io/docs/installation/installation-guide/example-install/

1. Create kiali-operator Namespace

```
$ kubectl create namespace kiali-operator
$ kubectl label namespace kiali-operator istio-injection=enabled
```

2. Install Kiali Operator

```
$ helm repo add kiali https://kiali.org/helm-charts
$ helm repo update kiali
$ helm install \
    --namespace kiali-operator \
    kiali/kiali-operator
```

3. Create Kiali CR file (e.g. kiali.yaml)

kiali.yaml

```
apiVersion: kiali.io/v1alpha1
kind: Kiali
metadata:
  name: kiali
  namespace: istio-system
  annotations:
    ansible.operator-sdk/verbosity: "1"
spec:
  auth:
    strategy: anonymous
  istio_component_namespaces:
    prometheus: monitoring
  external_services:
    grafana:
      in_cluster_url: http://prometheus-stack-grafana.monitoring
    prometheus:
      url: http://prometheus-stack-kube-prom-prometheus.monitoring:9090
    tracing:
      in_cluster_url: http://istio-query.observability:16686
  deployment:
    accessible_namespaces: ["**"]
    view_only_mode: false
  server:
    web_root: "/kiali"
```

4. Install Kiali

```
$ kubectl apply -f kiali.yaml
```

5. Create Ingress gateway entry for the Kiali web interface

kiali-Ingress.yaml

```
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
  name: kiali-gateway
spec:
  selector:
    istio: ingress
  servers:
    - hosts:
        - kiali.simpledemo.onap.org
      port:
        name: http
        number: 80
        protocol: HTTP
---
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: kiali-service
spec:
  hosts:
    - kiali.simpledemo.onap.org
  gateways:
    - kiali-gateway
  http:
    - route:
      - destination:
          port:
            number: 20001
          host: kiali
```

6. Add the Ingress entry for Kiali

```
$ kubectl -n istio-system apply -f kiali-ingress.yaml
```

Configure ONAP charts

Global settings

Global values used for ServiceMesh and Ingress setup can be found in

- https://git.onap.org/oom/tree/kubernetes/onap/values.yaml
- https://git.onap.org/oom/tree/kubernetes/onap/resources/overrides/onap-all-ingress-istio.yaml

The following variable settings are used for enabling ServiceMesh as well as Istio Ingress:

```
#ingress virtualhost based configuration
global:
  ingress:
    enabled: true
    virtualhost:
      baseurl: "simpledemo.onap.org"
    # All http requests via ingress will be redirected
    config:
      ssl: "redirect"
    # you can set an own Secret containing a certificate
    #  tls:
    #    secret: 'my-ingress-cert'
    # optional: Namespace of the Istio IngressGateway
    namespace: istio-ingress
...
  serviceMesh:
    enabled: true
    tls: true
    # be aware that linkerd is not well tested
    engine: "istio" # valid value: istio or linkerd
  aafEnabled: false
  cmpv2Enabled: false
  tlsEnabled: false
  msbEnabled: false
```

ServiceMesh settings:

- **enabled: true** enables ServiceMesh functionality in the ONAP Namespace (Istio: enables Sidecar deployment)
- **tls: true** enables mTLS encryption in Sidecar communication
- **engine: istio** sets the SM engine (currently only Istio is supported)
- **aafEnabled: false** disables AAF usage for TLS interfaces
- **tlsEnabled: false** disables creation of TLS in component services
- **cmpv2Enabled: false** disable cmpv2 feature
- **msbEnabled: false** MSB is not used in Istio setup (Open, if all components are MSB independend)

Ingress settings:

- **enabled: true** enables Ingress using: Nginx (when SM disabled), Istio IngressGateway (when SM enabled)
- **virtualhost.baseurl: "simpledemo.onap.org"** sets globally the URL for all Interfaces set by the components, resulting in e.g. "aai-api.simpledemo.onap.org"
- **config.ssl: redirect** sets in the Ingress globally the redirection of all Interfaces from http (port 80) to https (port 443)
- **config.tls.secret: "..."** (optional) overrides the default selfsigned SSL certificate with a certificate stored in the specified secret
- **namespace: istio-ingress** (optional) overrides the namespace of the ingress gateway which is used for the created SSL certificate

Install ONAP

1. Clone OOM repository from ONAP

   ```
   $ git clone -b <BRANCH> http://gerrit.onap.org/r/oom --recurse-submodules
   ```

2. Create an ServiceMesh override file (here ~/onap-overides.yaml) with the following example content (including a workarround for DMAAP AAF issue)
   The Ingress configuration can be found in ~/oom/kubernetes/onap/resources/overrides/onap-all-ingress-istio.yaml

```
---
global:
  serviceMesh:
    enabled: true
    tls: true
  aafEnabled: false
  tlsEnabled: false
  msbEnabled: false

aaf:
  # workarround for DMAAP SM issue
  enabled: true
  global:
    aafEnabled: true
dmaap:
  # workarround for DMAAP SM issue
  global:
    aafEnabled: true
```

3. Install Helm Plugins

```
$ helm plugin install --version v0.10.2 https://github.com/chartmuseum/helm-push.git
$ helm plugin install /opt/oom/kubernetes/helm/plugins/deploy
$ helm plugin install /opt/oom/kubernetes/helm/plugins/undeploy
```

4. Install ChartMuseum as Helm Registry, start it and add local repository

```
$ curl -LO https://s3.amazonaws.com/chartmuseum/release/latest/bin/linux/amd64/chartmuseum
$ chmod +x ./chartmuseum
$ mv ./chartmuseum /usr/local/bin
$ chartmuseum --storage local --storage-local-rootdir ~/helm3-storage -port 8879 &
$ helm repo add local http://127.0.0.1:8879
```

5. Compile ONAP helm charts (here with 4 parallel threads)

```
$ cd ~/oom
$ make all -j4
```

6. Create ONAP namespace and label it for Istio sidecar injection:

```
$ kubectl create namespace onap
$ kubectl label namespace onap istio-injection=enabled --overwrite=true
```

7. Deploy ONAP:

```
$ helm deploy onap local/onap --namespace onap --version 11.0.0 --values ~/oom/kubernetes/onap/resources
/overrides/onap-all-ingress-istio.yaml --values ~/oom/kubernetes/onap/resources/overrides/environment.
yaml --values ~/onap-overrides.yaml --timeout '900s'
```

8. Re-deploy or upgrade a single components (here platform)

```
$ helm upgrade -i onap-platform local/platform --namespace onap --version 11.0.0 --values ~/oom
/kubernetes/onap/values.yaml --values ~/oom/kubernetes/onap/resources/overrides/onap-all-ingress-istio.
yaml --values ~/oom/kubernetes/onap/resources/overrides/environment.yaml --values ~/onap-overrides.yaml
--timeout '900s'
```

Access ONAP APIs/UIs

In the ServiceMesh deployment the Istio IngressGateway is the only accesspoint for ONAP component interfaces.
Usually the Ingress is accessed via a LoadBalancer IP (<ingress-IP>, which is used as central address.
All APIs/UIs are provided via separate URLs which are routed to the component service.
To use these URLs they need to be resolvable via DNS or via /etc/hosts, here is the example:

```
<ingress-IP> kiali.simpledemo.onap.org
<ingress-IP> aaf-cm-api.simpledemo.onap.org
<ingress-IP> aaf-fs-api.simpledemo.onap.org
<ingress-IP> aaf-locate-api.simpledemo.onap.org
<ingress-IP> aaf-oauth-api.simpledemo.onap.org
<ingress-IP> aaf-service-api.simpledemo.onap.org
<ingress-IP> aaf-ui.simpledemo.onap.org
<ingress-IP> aai-api.simpledemo.onap.org
<ingress-IP> aai-babel-api.simpledemo.onap.org
<ingress-IP> aai-sparkybe-api.simpledemo.onap.org
<ingress-IP> appc-dgbuilder.simpledemo.onap.org
<ingress-IP> appc-api.simpledemo.onap.org
<ingress-IP> cds-blueprintsprocessor-api.simpledemo.onap.org
<ingress-IP> cds-ui.simpledemo.onap.org
<ingress-IP> cli-api.simpledemo.onap.org
<ingress-IP> cli2-api.simpledemo.onap.org
<ingress-IP> consul-api.simpledemo.onap.org
<ingress-IP> cps-core-api.simpledemo.onap.org
<ingress-IP> cps-ncmp-dmi-plugin-api.simpledemo.onap.org
<ingress-IP> cps-temporal-api.simpledemo.onap.org
<ingress-IP> dcaemod-distributor-api.simpledemo.onap.org
<ingress-IP> dcaemod-genprocessor-api.simpledemo.onap.org
<ingress-IP> dcaemod-nifi-ui.simpledemo.onap.org
<ingress-IP> dcaemod-nifi-api.simpledemo.onap.org
<ingress-IP> dcaemod-onboarding-api.simpledemo.onap.org
<ingress-IP> dmaap-bc-api.simpledemo.onap.org
<ingress-IP> dmaap-dr-node-api.simpledemo.onap.org
<ingress-IP> dmaap-dr-prov-api.simpledemo.onap.org
<ingress-IP> dmaap-mr-api.simpledemo.onap.org
<ingress-IP> keycloak-ui.simpledemo.onap.org
<ingress-IP> log-es-api.simpledemo.onap.org
<ingress-IP> log-kibana-ui.simpledemo.onap.org
<ingress-IP> log-ls-api.simpledemo.onap.org
<ingress-IP> log-ls-http-api.simpledemo.onap.org
<ingress-IP> msb-consul-api.simpledemo.onap.org
<ingress-IP> msb-discovery-api.simpledemo.onap.org
<ingress-IP> msb-eag-ui.simpledemo.onap.org
<ingress-IP> msb-iag-ui.simpledemo.onap.org
<ingress-IP> nbi-api.simpledemo.onap.org
<ingress-IP> oof-has-api.simpledemo.onap.org
<ingress-IP> oof-osdf-api.simpledemo.onap.org
<ingress-IP> policy-ui.simpledemo.onap.org
<ingress-IP> robot-api.simpledemo.onap.org
<ingress-IP> sdc-be-api.simpledemo.onap.org
```