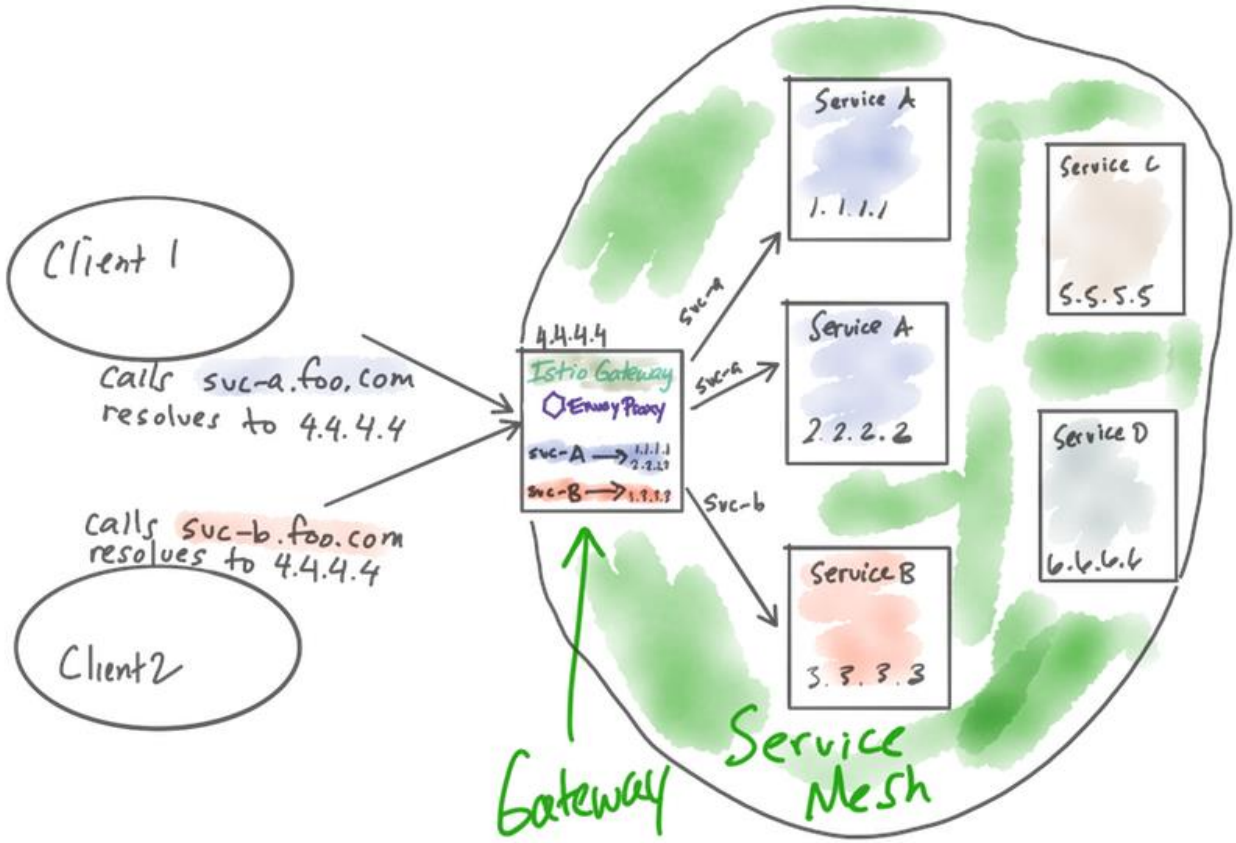
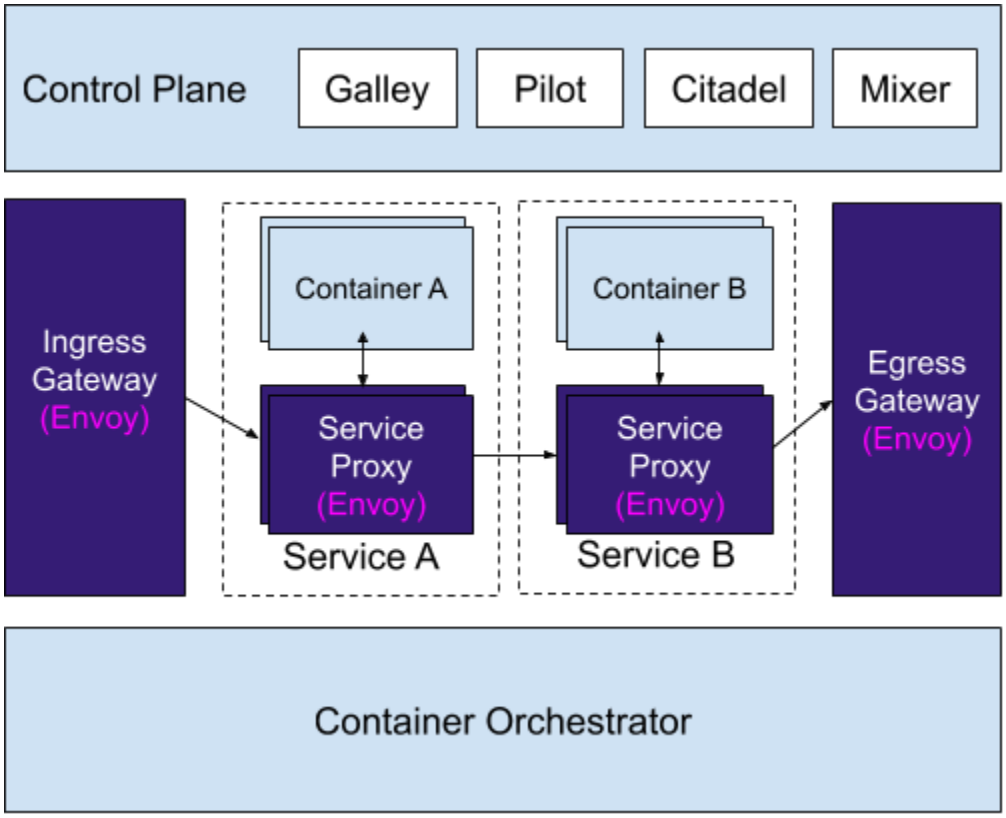
Expose Microservices through Istio Ingress Gateway:

The components deployed on the service mesh by default are not exposed outside the cluster. External access to individual services so far has been provided by creating an external load balancer on each service.

Ingress gateway: An ingress gateway service is deployed as a LoadBalancer service. For making Bookinfo accessible from outside, we have to create an Istio Gateway for the service and also define an Istio **VirtualService** for Bookinfo with the routes we need.

1. Inspecting the Istio Ingress gateway:





The ingress gateway gets exposed as a normal kubernetes service load balancer:

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

Because the Istio Ingress Gateway has an Envoy Proxy you can inspect it using the admin routes. First find the name of the istio-ingressgateway:

```
kubectl get pods -n istio-system
```

Copy and paste your ingress gateway's pod name. Execute:

```
kubectl -n istio-system exec -it <istio-ingressgateway-...> bash
```

You can view the statistics, listeners, routes, clusters and server info for the Envoy proxy by forwarding the local port:

```
curl localhost:15000/help
curl localhost:15000/stats
```

```
curl localhost:15000/listeners
curl localhost:15000/clusters
curl localhost:15000/server_info
```

See the admin docs for more details.

Also it can be helpful to look at the log files of the Istio ingress controller to see what request is being routed. We should also be able to view the curl calls we just made from inside the ingressgateway.

Before we check the logs, let us get out of the container back on the host:

```
exit
```

Now let us find the ingress pod and output the log:

```
kubectl logs istio-ingressgateway-... -n istio-system
```

2. Configure Istio Ingress Gateway for Bookinfo:

2.1 - Configure the Bookinfo route with the Istio Ingress gateway:

We can create a virtualservice & gateway for bookinfo app in the ingress gateway by running the following:

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

2.2 - View the Gateway and VirtualServices

Check the created `Istio Gateway` and `Istio VirtualService` to see the changes deployed:

```
kubectl get gateway
kubectl get gateway -o yaml

kubectl get virtualservices
kubectl get virtualservices -o yaml
```

2.3 - Find the external port of the Istio Ingress Gateway by running:

```
kubectl get service istio-ingressgateway -n istio-system -o wide
```

To just get the first port of istio-ingressgateway service, we can run this:

```
kubectl get service istio-ingressgateway -n istio-system --
template='{{(index .spec.ports 0).nodePort}}'
```

The default HTTP port is usually 31380.

2.4 - Browse to Bookinfo:

Browse to the website of the Bookinfo:

To view the product page, you will have to append `/productpage` to the url.

2.5 - Reload Page

Now, reload the page multiple times and notice how it round robins between v1, v2 and v3 of the reviews service.

3 Inspect the Istio proxy of the productpage pod

To better understand the istio proxy, let's inspect the details. Let us exec into the productpage pod to find the proxy details. To do so we need to first find the full pod name and then exec into the istio-proxy container:

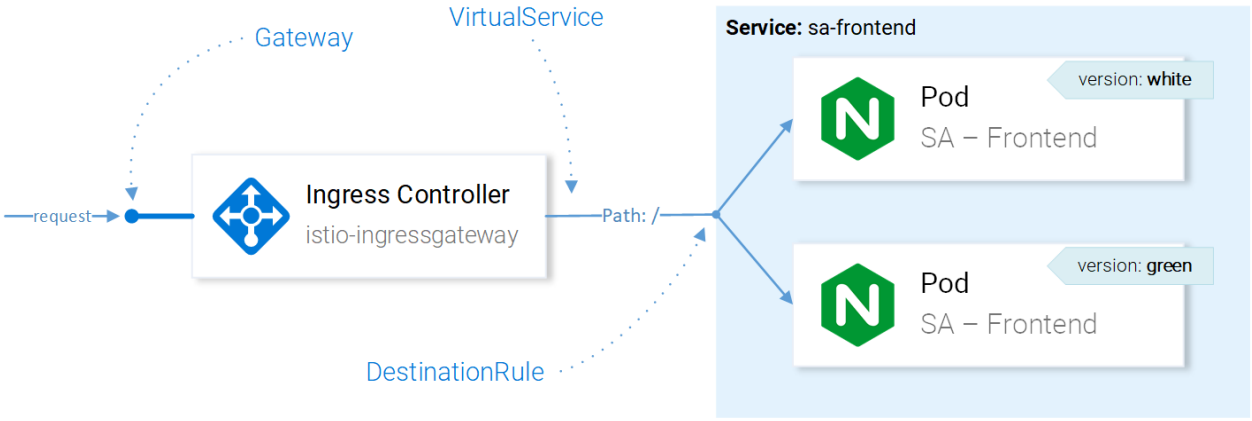
```
kubectl get pods
kubectl exec -it productpage-v1-... -c istio-proxy  sh
```

Once in the container look at some of the envoy proxy details by inspecting it's config file:

```
ps aux
ls -l /etc/istio/proxy
cat /etc/istio/proxy/envoy-rev0.json
```

For more details on envoy proxy please check out their admin docs for more details.

4 Apply default destination rules



DestinationRule defines policies that apply to traffic intended for a service after routing has occurred. Using destination rule we can manipulate the traffic with policy (Exampe: Circuit breakers) before sending traffic to the final destination.

Before we start playing with Istio's traffic management capabilities we need to define the available versions of the deployed services. They are called subsets, in destination rules.

Run the following command to create default destination rules for the Bookinfo services:

```
kubectl apply -f destination-rule-all-mtls.yaml
```

In a few seconds we should be able to verify the destination rules created by using the command below:

```
kubectl get destinationrules
```

```
kubectl get destinationrules -o yaml
```