**Exercise 3.1: Deploy Kong Service Mesh in standalone mode**

1. Download the latest version of Kong Service Mesh. The below will download and extract the latest package.

student@controlplane~: $ curl -L https://docs.konghq.com/mesh/installer.sh | sh -

| % Total % Received % Xferd Average Speed Time Time Time Current  Dload Upload Total Spent Left Speed  100 3278 100 3278 0 0 16308 0 --:--:-- --:--:-- --:--:-- 16308  INFO Welcome to the Kong Mesh automated download!  INFO Fetching latest Kong Mesh version..  INFO Kong Mesh version: 1.3.2  INFO Kong Mesh architecture: amd64  INFO Operating system: centos  INFO Downloading Kong Mesh from: https://download.konghq.com/mesh-alpine/kong-mesh-1.3.2-centos-amd64.tar.gz  <output omitted> |
| --- |

1. Change to the bin folder and run kumactl install to generate the necessary YAML to install the control plane. We will pipe the output to kubectl to apply it to our cluster.

student@controlplane ~: $ cd kong-mesh-1.3.3/bin

student@controlplane bin: $ ./kumactl install control-plane --license-path=./license.json | kubectl apply -f -

| namespace/kong-mesh-system created  serviceaccount/kong-mesh-control-plane created  secret/kong-mesh-tls-cert created  secret/kong-mesh-license created  configmap/kong-mesh-control-plane-config created  customresourcedefinition.apiextensions.k8s.io/opapolicies.kuma.io created  customresourcedefinition.apiextensions.k8s.io/circuitbreakers.kuma.io created  customresourcedefinition.apiextensions.k8s.io/retries.kuma.io created  customresourcedefinition.apiextensions.k8s.io/serviceinsights.kuma.io created  customresourcedefinition.apiextensions.k8s.io/timeouts.kuma.io created  <output omitted> |
| --- |

1. Verify the installation via command line. Assuming the setup was successful we should be able to reach the Admin API.
2. Setup port forwarding for the Admin API.

student@controlplane bin: $ kubectl port-forward svc/kong-mesh-control-plane 5681:5681 -n kong-mesh-system &

1. Confirm the installation by ensuring the port is accessible.

student@controlplane bin: $ curl 127.0.0.1:5681

| {  "hostname": "kong-mesh-cp.internal",  "tagline": "Kong Mesh",  "version": "1.3.2",  "basedOnKuma": "1.2.2"  } |
| --- |

**Exercise 3.2: Run the Marketplace Application**

The Kuma Demo Application is a clothing marketplace where you can browse listed items along with the reviews left by users. It consists of four components: Vue frontend UI, Node backend API, PostgreSQL, and Redis. More details can be found [here](https://github.com/kumahq/kuma-demo).

1. Apply the Marketplace manifest to the cluster to create the necessary resources.

student@controlplane bin: $ kubectl apply -f https://bit.ly/demokuma

| namespace/kuma-demo created  deployment.apps/postgres-master created  service/postgres created  deployment.apps/redis-master created  service/redis created  service/backend created  deployment.apps/kuma-demo-backend-v0 created  deployment.apps/kuma-demo-backend-v1 created  deployment.apps/kuma-demo-backend-v2 created  service/frontend created  deployment.apps/kuma-demo-app created |
| --- |

1. Setup port forwarding and view the Marketplace Application in a browser by navigating to <http://127.0.0.1:8080>

student@controlplane bin: $ kubectl port-forward svc/frontend -n kuma-demo 8080:8080 &

| <no output> |
| --- |

1. View the sidecar proxy injected in the demo application namespace.

On Kubernetes the data-planes are automatically injected by Kuma as long as the K8s Namespace or Pod are annotated with kuma.io/sidecar-injection = enabled. The Marketplace application sets this by default.

student@controlplane bin: $ kubectl describe ns kuma-demo

| Name: kuma-demo  Labels: <none>  Annotations: **kuma.io/sidecar-injection: enabled**  Status: Active  No resource quota.  No LimitRange resource. |
| --- |

You can view the sidecar proxy injected in the demo application namespace by navigating to: <http://127.0.0.1:5681/gui/#/mesh/all/dataplanes>

**Exercise 3.3: Configure mTLS**

1. Enable Mutual TLS with a builtin CA backend using the provided YAML in your assets folder.

student@controlplane bin: $ kubectl apply -f mTLS.yaml

| mesh.kuma.io/default configured |
| --- |

**Exercise 3.4: Create a Traffic Permission**

The default TrafficPermission policy that Service Mesh creates allows all communication between *all* services in the new Mesh. Let’s take a look at the current config.

1. student@controlplane bin: $ kubectl get trafficpermissions.kuma.io allow-all-default -o yaml

| <output omitted>  spec:  destinations:  - match:  kuma.io/service: '\*'  sources:  - match:  kuma.io/service: '\*' |
| --- |

1. Let’s delete the Traffic Permission policy and see how the Marketplace application responds.

student@controlplane bin: $ kubectl delete trafficpermissions.kuma.io allow-all-default

| trafficpermission.kuma.io "allow-all-default" deleted |
| --- |

1. Reload the Marketplace Application by navigating to <http://localhost:8080/#>.

**There are Product API issues:**

Error: Request failed with status code 503

1. We will now explicitly allow traffic between the Marketplace Application services.

**Note:** The service names used can be obtained from the Service Mesh GUI [http://localhost:5681/gui/#/mesh/all/dataplanes]

* frontend: the entry-point service that serves the web application.
* backend: the underlying backend component that powers the frontend service.
* postgres: the database that stores the marketplace items.
* redis: the backend storage for items reviews.

student@controlplane bin: $ kubectl apply -f trafficPermission.yaml

| trafficpermission.kuma.io/marketplace created |
| --- |

1. Refresh the Marketplace Application page to confirm it now loads.