|  |  |
| --- | --- |
| **DOCUMENT RULES:** | |
| **Task Number / Name:** | **Envoy&Issio** |
| **Task name & column name should be written:** | **Bold (CTRL+B)** |
| **Commands should be written in the after # sign:** | *Italic (CTRL+I) #hostname* |
| **Output photo should be cropped or compressed:**  **Photo could be more than one:**  **If you need extra lines, add the line next after it:** | ***Description photo should be with title bar (CTRL + I + B)*** |
| **All other text should be written:** | Standard |
| **Font name and text size:** | Calibri and 9 |
| **Group name:** | Dev\_ops\_ |
| **Student name and surname:** | Murad Abbaszade |
| **E-mail:** | [muradabbaszade6@gmail.com](mailto:muradabbaszade6@gmail.com) |
| **WhatsApp number:** | **+994703664205** |

|  |  |
| --- | --- |
| 1.Envoy Basics |  |
| Verify that func-e is installed  func-e --version  Run the *help* command to familiarize yourself with the CLI.  func-e help  Review the help for the run subcommand.  func-e run --help |  |
| Review the file's contents:  cat minimal-config.yaml  envoy will listen on port 10000, and is configured with no filters. |  |
| Run envoy in the background:  func-e run --config-path minimal-config.yaml |  |
| Try to send a request to localhost:10000 :  curl -v localhost:10000  envoy does not yet know how to route the request.  Review the file's contents:  cat direct-response.yaml |  |
| Review the file's contents:  cat direct-response.yaml |  |
| Run envoy in the background:  func-e run --config-path direct-response.yaml & |  |
| Send a request to localhost:10000 :  curl -v localhost:10000  We now get an HTTP 200 response, with the "Hello!" response body. |  |
| Use docker to run an instance of [httpbin](https://httpbin.org/" \t "_blank) on port 8100:  docker run -d -p 8100:80 kennethreitz/httpbin |  |
| Run a second instance on port 8200:  docker run -d -p 8200:80 kennethreitz/httpbin |  |
| Verify that the two docker containers are up and running:  docker ps |  |
| Next, inspect the contents of the envoy configuration file named clusters.yaml :  cat clusters.yaml |  |
| Run envoy in the background:  func-e run --config-path clusters.yaml & |  |
| Send a token request to envoy:  curl localhost:10000/headers |  |
| And another:  curl localhost:10000/html |  |
| Review the contents of the envoy configuration file named routing.yaml :  cat routing.yaml |  |
| Run envoy in the background:  func-e run --config-path routing.yaml & |  |
| Check that the two docker containers are still running:  docker ps |  |
| Send a request to envoy, targeting the first backend:  curl localhost:10000/one |  |
| And another, targeting the second backend:  curl localhost:10000/two |  |
| 2.Istio 0 to 60 |  |
| Verify that kubectl is configured:  k get ns |  |
| export ISTIO\_VERSION=1.15.0  Download the Istio distribution:  curl -L https://istio.io/downloadIstio | TARGET\_ARCH=x86\_64 sh - |  |
| ls -lF |  |
| cp istio-1.15.0/bin/istioctl /usr/local/bin  Verify that the CLI is functioning:  istioctl version |  |
| istioctl x precheck |  |
| Install Istio onto your Kubernetes cluster:  istioctl install -f install-manifest.yaml -y |  |
| istioctl verify-install |  |
| istioctl version |  |
| List the namespaces in your Kubernetes cluster:  k get ns |  |
| k get deploy -n istio-system |  |
| k get svc -n istio-system |  |
| Label the default namespace  kubectl label namespace default istio-injection=enabled |  |
| Verify that the label has been applied:  kubectl get ns -Listio-injection |  |
| cat web-frontend.yaml |  |
| cat customers.yaml |  |
| Apply the two files to your Kubernetes cluster.  k apply -f customers.yaml  k apply -f web-frontend.yaml |  |
| k get pod |  |
| k get mutatingwebhookconfigurations |  |
| cat sleep.yaml |  |
| k apply -f sleep.yaml |  |
| k get pod |  |
| Capture the name of the sleep pod to an environment variable  SLEEP\_POD=$(kubectl get pod -l app=sleep -ojsonpath='{.items[0].metadata.name}')  Use the kubectl exec command to call the customers service.  k exec $SLEEP\_POD -it -- curl customers |  |
| Call the web-frontend service  kubectl exec $SLEEP\_POD -it -- curl web-frontend | head |  |
| View the corresponding Istio ingress gateway pod in the istio-system namespace.  k get pod -n istio-system |  |
| A corresponding LoadBalancer type service was also created:  k get svc -n istio-system |  |
| Review the following Gateway specification.  cat gateway.yaml |  |
| Apply the gateway resource to your cluster.  k apply -f gateway.yaml |  |
| Expose the ingress gateway, using the arbitrary host port number 1234:  k port-forward -n istio-system --address 0.0.0.0 service/istio-ingressgateway 1234:80 |  |
| Review the following VirtualService specification.  cat web-frontend-virtualservice.yaml |  |
| Apply the virtual service resource to your cluster.  k apply -f web-frontend-virtualservice.yaml |  |
| List virtual services in the default namespace.  k get virtualservice |  |
| Navigate to the addons directory  cd ~/istio-1.15.0/samples/addons |  |
| Deploy each addon:  k apply -f extras/zipkin.yaml  k apply -f prometheus.yaml  k apply -f grafana.yaml  k apply -f kiali.yaml |  |
| Verify that the istio-system namespace is now running additional workloads for each of the addons.  k get pod -n istio-system |  |
| Install a simple load generating tool named [siege](https://github.com/JoeDog/siege).  apt-get install siege |  |
| With siege now installed, familiarize yourself with the command and its options.  siege --help |  |
| Run the following command to generate a mild load against the application.  INGRESS\_CLUSTER\_IP=$(kubectl get svc -n istio-system istio-ingressgateway -ojsonpath='{.spec.clusterIP}')  siege --delay=3 --concurrent=3 --time=20M http://$INGRESS\_CLUSTER\_IP/ |  |
| Launch the Zipkin dashboard:  k port-forward -n istio-system --address 0.0.0.0 service/zipkin 9411:9411 |  |
| Capture the customers pod name to a variable.  CUSTOMERS\_POD=$(kubectl get pod -l app=customers -ojsonpath='{.items[0].metadata.name}')  Run the following command:  k exec $CUSTOMERS\_POD -it -- curl localhost:15090/stats/prometheus | grep istio\_requests |  |
| Start the prometheus dashboard  k port-forward -n istio-system --address 0.0.0.0 service/prometheus 9090:9090 |  |
| Note the selector on the customers service:  k get svc customers -o wide |  |
| k get virtualservice |  |