

**Service Mesh Installation and Configuration   
with Istio on Kubernetes for Production**

**Version History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SI | Date | Ver | Description | Changed by | Approved/Reviewed by |
| 01 | 16-Feb-22 | 1.0 | Service Mesh Installation and Configuration  with Istio on Kubernetes for Production | Maidul Islam  Software Engineer | Md. Mahedee Hasan  Head of Software Development |

Contents

[Introduction 3](#_Toc95929244)

[Istio 3](#_Toc95929245)

[Installation Steps 4](#_Toc95929246)

[1.0 Download Istio 4](#_Toc95929247)

[1.1 Configure istioctl Command 4](#_Toc95929248)

[1.2 Installing Istio in Production 5](#_Toc95929249)

[1.3 Verify Istio 5](#_Toc95929250)

[1.2 Configure auto Proxy Injection 6](#_Toc95929251)

[Ingress Gateways 6](#_Toc95929252)

[Determining the ingress IP and ports 7](#_Toc95929253)

[Configure Kiali 8](#_Toc95929254)

[1.2 The Kiali CR 9](#_Toc95929255)

[Configure others 11](#_Toc95929256)

[References 13](#_Toc95929257)

# Introduction

is a way to control how different parts of an application share data with one another. Unlike other systems for managing this communication, a service mesh is a dedicated infrastructure layer built right into an app. This visible infrastructure layer can document how well (or not) different parts of an app interact, so it becomes easier to optimize communication and avoid downtime as an app grows.

Modern applications are often broken down in this way, as a network of services each performing a specific business function. In order to execute its function, one service might need to request data from several other services. But what if some services get overloaded with requests, like the retailer’s inventory database? This is where a service mesh comes in—it routes requests from one service to the next, optimizing how all the moving parts work together.

# Istio

Istio is an open platform for providing a uniform way to integrate microservices, manage traffic flow across microservices, enforce policies and aggregate telemetry data. Istio's control plane provides an abstraction layer over the underlying cluster management platform, such as Kubernetes, Mesos, etc.

Managing traffic is one of the core features of Istio and it is really important to good understanding of the three resources that Istio uses.

1. Gateway – External traffic comes through gateway into the cluster and bring into the service mesh, so the rules apply.
2. Virtual Services – Virtual Service can be bound to the gateway. End-user traffic comes through the gateway into the virtual service, and Istio’s rules apply.
3. Destination Rule – it configured exactly where traffic gets routed and how the network treats that traffic.

**Prometheus:**

Monitoring tool and it monitor everything in a cluster.

**Grafana:**

It is data visualization tools for matrix data.

**Jaeger-collector and tracing:**

Service for tracing microservice request.

**Zipkin**

Zipkin is alternative of Jaeger.

# Installation Steps

These steps require you to have a cluster running a supported version of Kubernetes (1.19, 1.20, 1.21, 1.22).

## 1.0 Download Istio

1. Open Git Bash or Powershell.
2. Go to below link download the installation file for your OS,

<https://github.com/istio/istio/releases/tag/1.12.2>

or download and extract the latest release automatically (Linux or macOS):

curl -L https://istio.io/downloadIstio | sh -

1. Move to the Istio package directory. For example, if the package is istio-1.12.2.

cd istio-1.12.2

The installation directory contains:

* Sample applications in samples/
* The istioctl client binary in the bin/ directory.

## 1.1 Configure istioctl Command

Following steps to configure istioctl command.

1. Go to istio folder with below command.

cd istio-1.12.2

ls

cd bin

1. Now move istioctl to local-bin directory to use the istioctl command.

sudo mv istioctl /usr/local/bin

1. Now you can use istioctl command. Check the istioctl version with below command.

istioctl version

1. Check the compatibility of Istio with kubernetes version with below command.

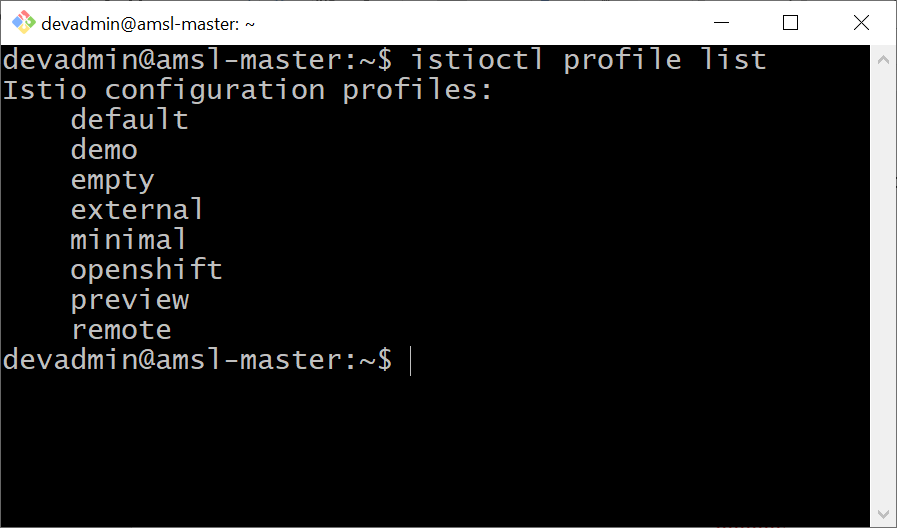
istioctl verify-install

## Installing Istio in Production

Now we are going to install istio for production use.

1. Check available profiles with below command.

istioctl profile list



1. We are going to choose default profile for production use. Now install istio with below command.

istioctl manifest apply --set profile=default

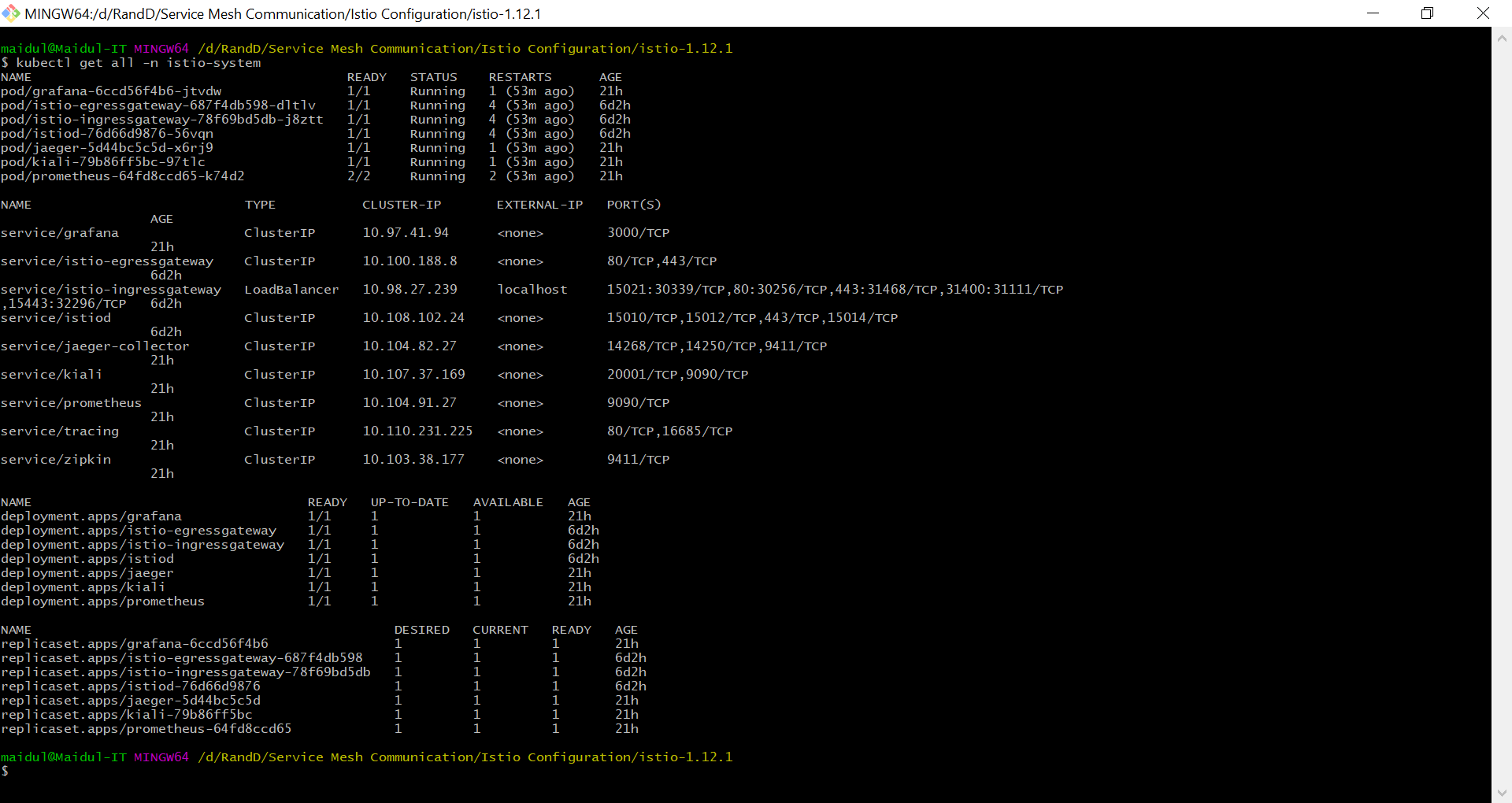
1. We can also generate manifest of istio profile with below command.

istioctl manifest generate > ./temp/istio-default-manifest.yaml

## 1.3 Verify Istio

1. Use below command to verify the running objects of istio system.

kubectl get all -n istio-system



All components have memory requests

## 1.2 Configure auto Proxy Injection

1. Add a namespace label to instruct Istio to automatically inject Envoy sidecar proxies when you deploy your application later. Use below command to configure default namespance with Istio sidecar proxy.

kubectl label namespace default istio-injection=enabled

1. Check label by using below command

kubectl describe namespace default

# Ingress Gateways

In a Kubernetes environment, the Kubernetes Ingress Resource is used to specify services that should be exposed outside the cluster. In an Istio service mesh, a better approach (which also works in both Kubernetes and other environments) is to use a different configuration model, namely Istio Gateway. A Gateway allows Istio features such as monitoring and route rules to be applied to traffic entering the cluster.

## Determining the ingress IP and ports

Execute the following command to determine if your Kubernetes cluster is running in an environment that supports external load balancers:

kubectl get svc istio-ingressgateway -n istio-system

If the EXTERNAL-IP value is set, your environment has an external load balancer that you can use for the ingress gateway. If the EXTERNAL-IP value is <none> (or perpetually <pending>), your environment does not provide an external load balancer for the ingress gateway. In this case, you can access the gateway using the service’s node port.

Follow these instructions if you have determined that your environment does not have an external load balancer, so you need to use a node port instead.

1. Set the ingress ports:

export INGRESS\_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="http2")].nodePort}')

export SECURE\_INGRESS\_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="https")].nodePort}')

Expose Prometheus command.

kubectl expose deployment prometheus --type=NodePort --name=prometheus-svc -n istio-system

Expose Grafana Command:

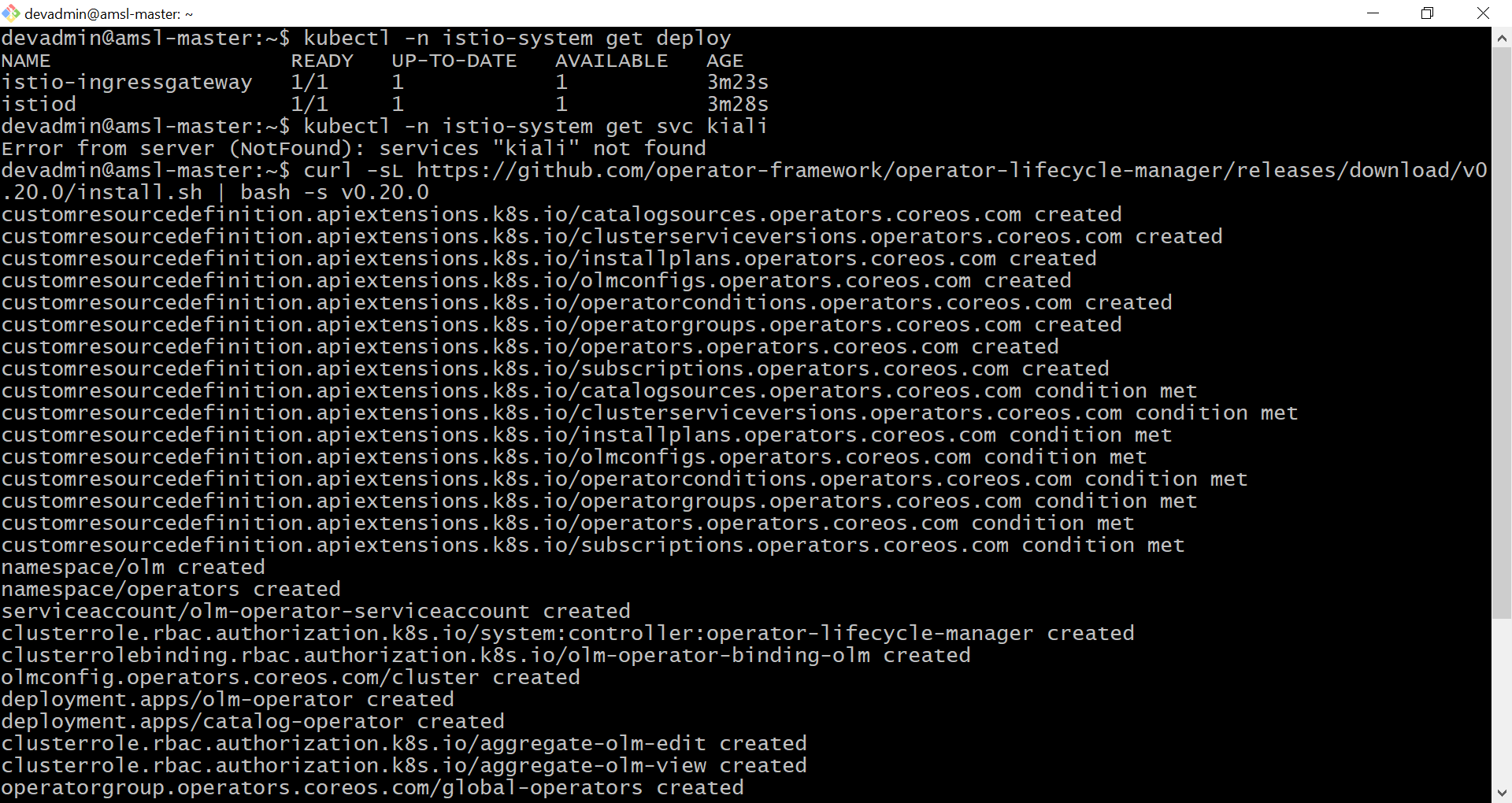
kubectl expose deployment grafana --type=NodePort --port=3000 --name=grafana-svc -n istio-system

# Configure Kiali

* 1. Kiali Operator

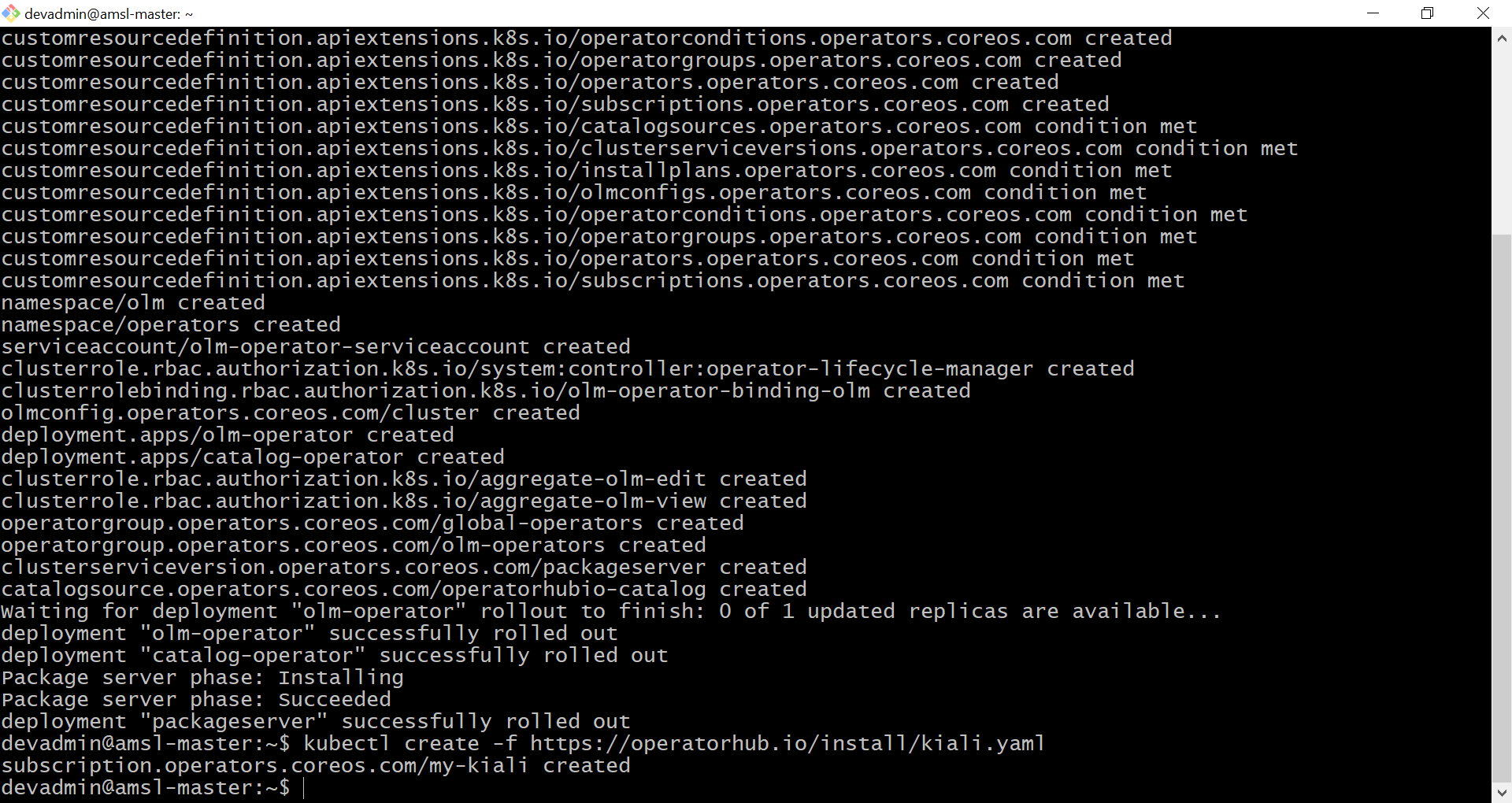
1. Install Operator Lifecycle Manager (OLM), a tool to help manage the Operators running on your cluster.

curl -sL https://github.com/operator-framework/operator-lifecycle-manager/releases/download/v0.20.0/install.sh | bash -s v0.20.0

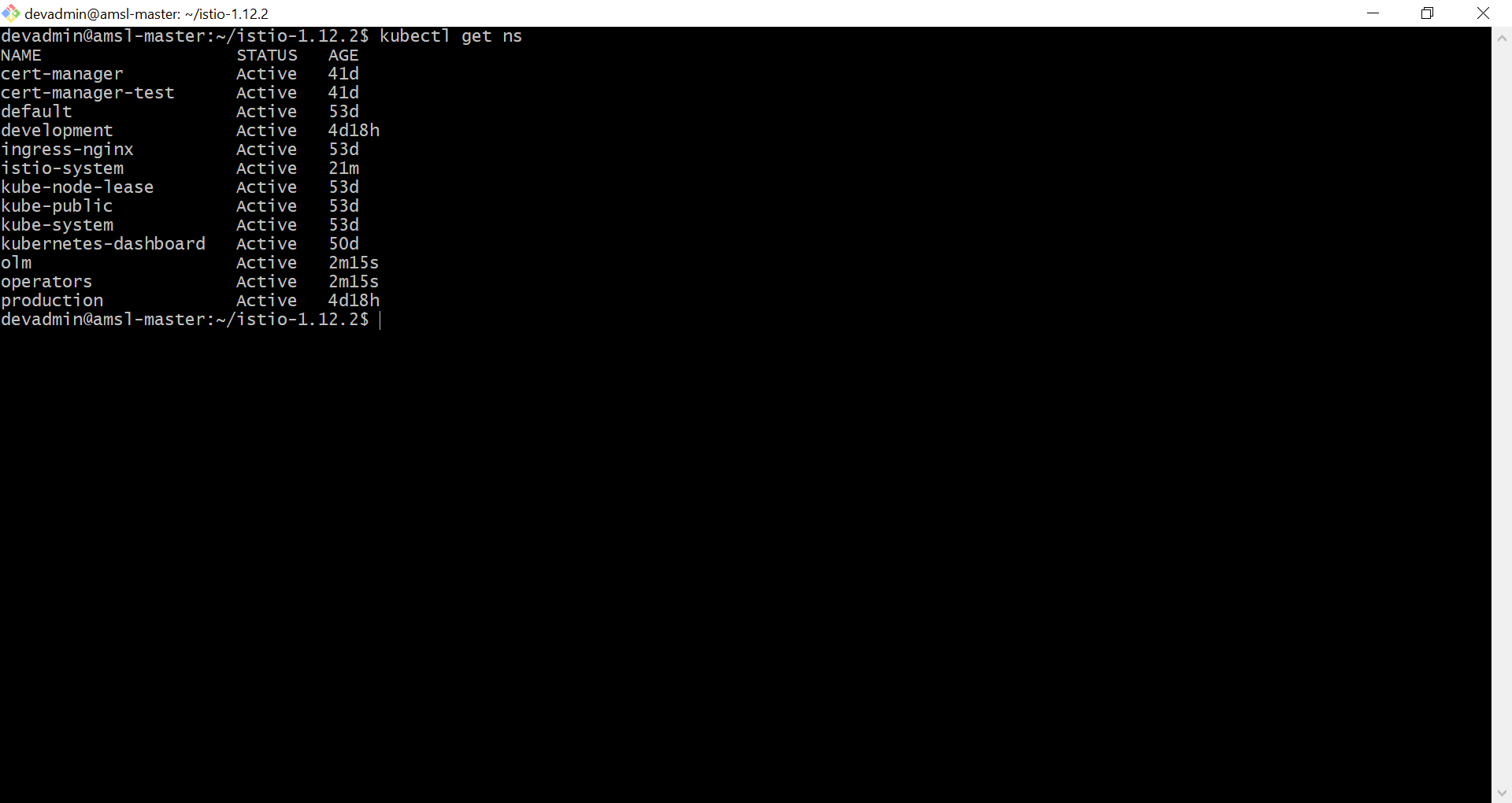


1. Install the operator by running the following command.

kubectl create -f <https://operatorhub.io/install/kiali.yaml>



This Operator will be installed in the "operators" namespace and will be usable from all namespaces in the cluster.



1. After install, watch your operator come up using next command.

kubectl get csv -n operators

## 1.2 The Kiali CR

The Kiali Operator watches the Kiali Custom Resource (Kiali CR), a YAML file that holds the deployment configuration. Creating, updating, or removing a Kiali CR will trigger the Kiali Operator to install, update, or remove Kiali.

1. Create a yaml file and name the file **my-kiali-cr.yaml** and write below code.

**apiVersion:** kiali.io/v1alpha1

**kind:** Kiali

**metadata:**

**name:** kiali

1. Execute the above file with below command.

kubectl apply -f my-kiali-cr.yaml -n istio-system

Often, but not always, Kiali is installed in the same namespace as Istio, thus the Kiali CR is also created in the Istio namespace.

1. Once created, the Kiali Operator should shortly be notified and will process the resource, performing the Kiali installation. You can check installation progress by inspecting the status attribute of the created Kiali CR:

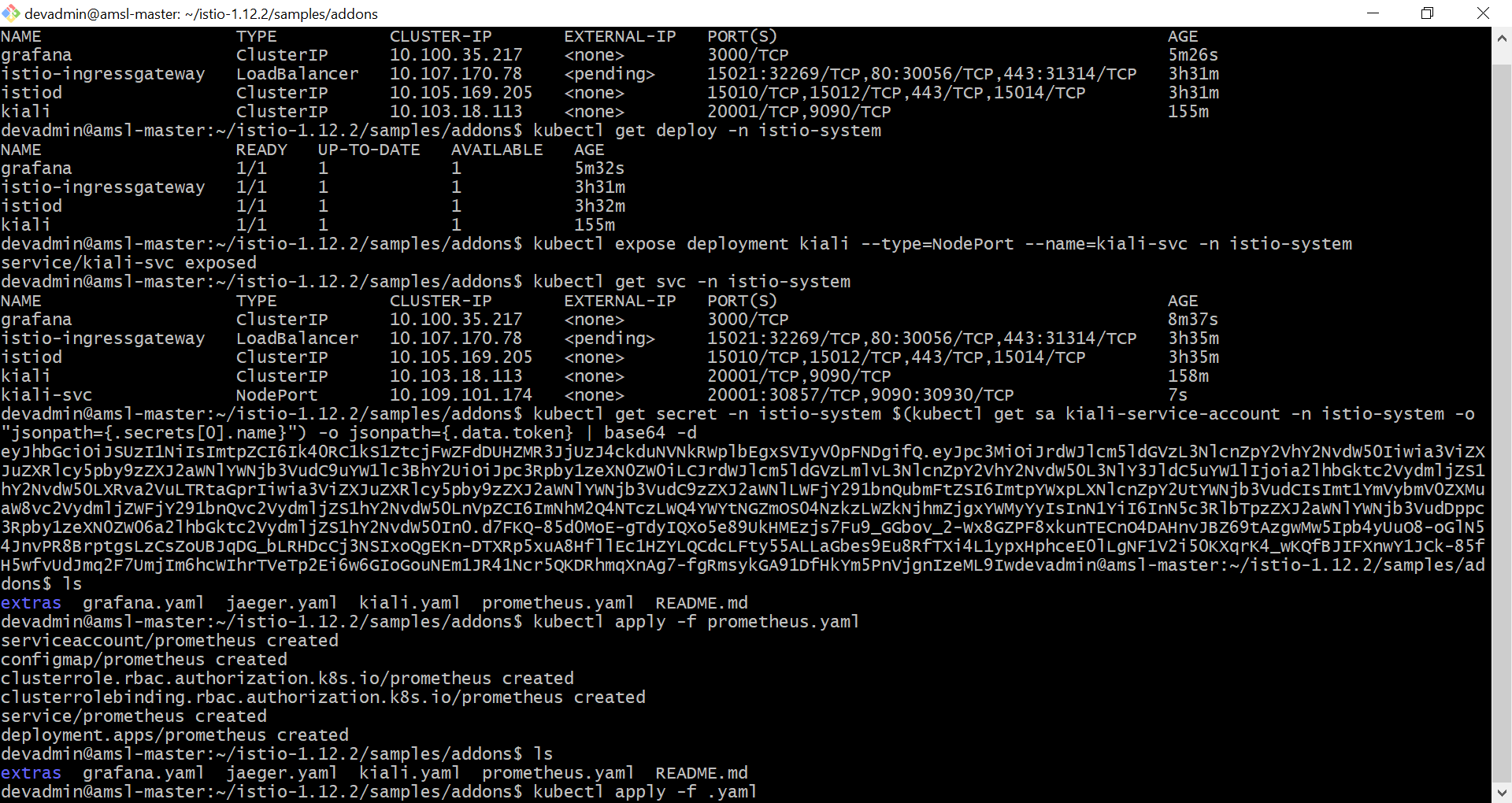
kubectl describe kiali -n istio-system

1. Now expose the kiali deployment to access from outside cluster.

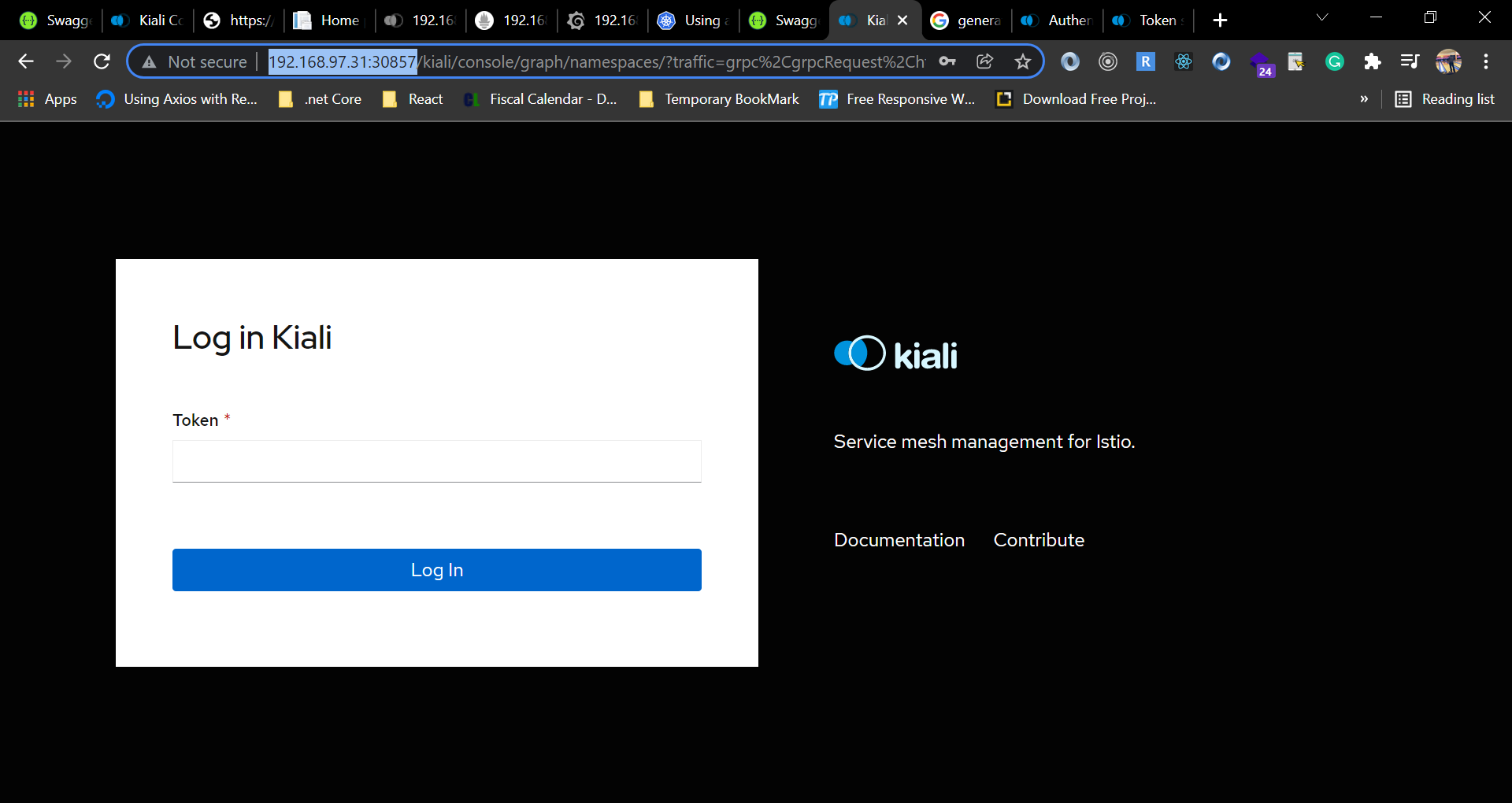
kubectl expose deployment kiali --type=NodePort --name=kiali-svc -n istio-system

1. Now check the deployment service with below command.

kubectl get svc -n istio-system

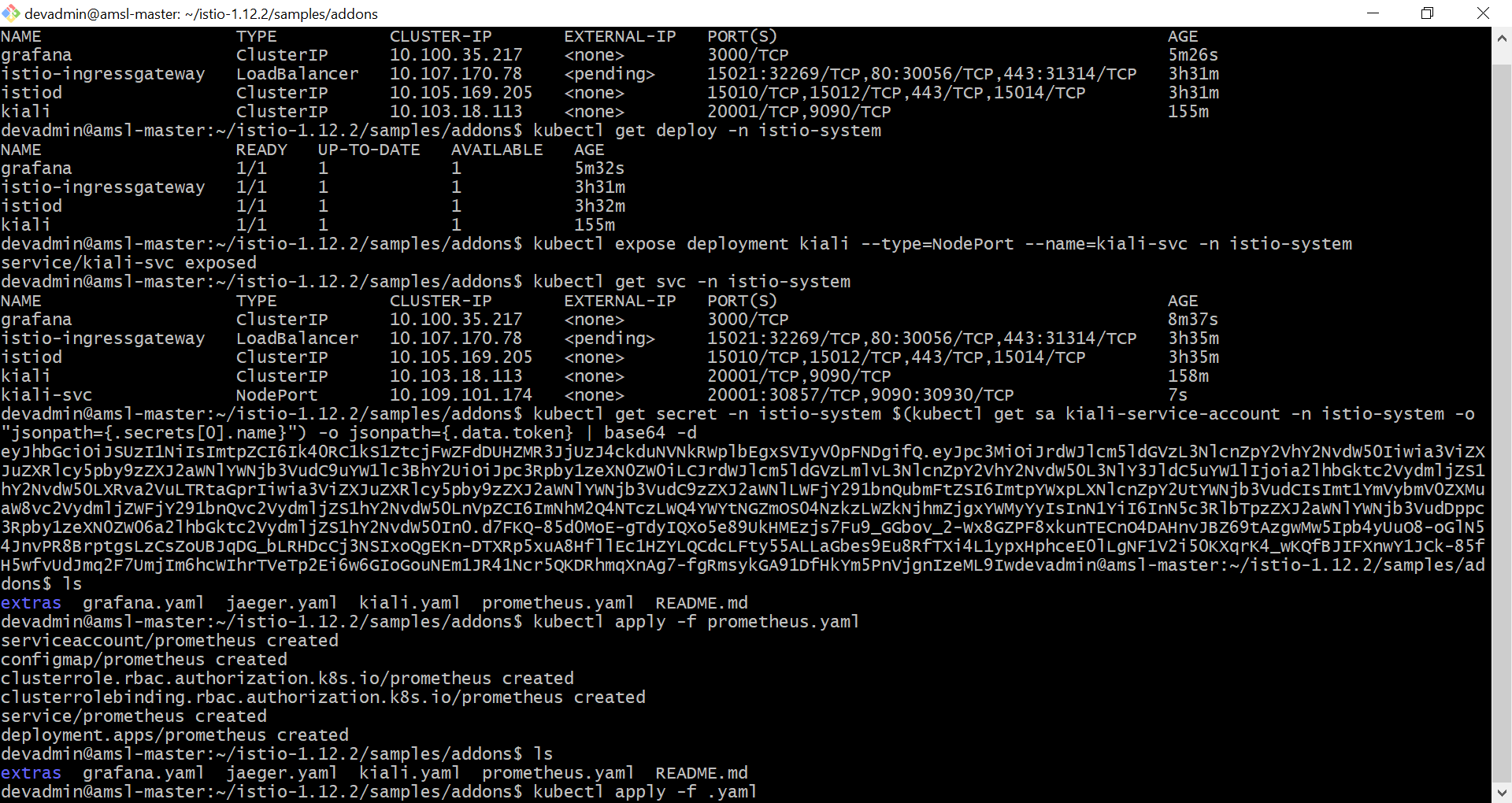


1. Now visit on <http://192.168.97.31:30857/> and you will see a screen like below,



1. Now write below command on git bash to generate a token to login.

kubectl get secret -n istio-system $(kubectl get sa kiali-service-account -n istio-system -o "jsonpath={.secrets[0].name}") -o jsonpath={.data.token} | base64 -d



1. Now copy the token and use to login on browser.

# Configure others

Now we are going to configure Prometheus, Grafana and Jaeger to collect information for kiali.

Kiali uses Grafana info to generate graph.

1. Now change the director with below command.

cd istio-1.12.2/samples/addons

1. Write below command to install.

kubectl apply -f prometheus.yaml

kubectl apply -f grafana.yaml

kubectl apply -f jaeger.yaml

1. Now expose above deployments as service.

kubectl expose deployment grafana --type=NodePort --port=3000 --name=grafana-svc -n istio-system

kubectl expose deployment prometheus --type=NodePort --name=prometheus-svc -n istio-system

kubectl expose deployment zipkin --type=NodePort --port=9411 --name=zipkin-svc -n istio-system

kubectl expose deployment jaeger --type=LoadBalancer --port=16686 --name=jaeger-svc -n istio-system

# References

<https://istio.io/latest/docs/setup/getting-started/>

<https://github.com/istio/istio/releases/tag/1.12.2>

<https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/>

<https://www.redhat.com/en/topics/microservices/what-is-a-service-mesh>

<https://www.nginx.com/blog/what-is-a-service-mesh/>

<https://kiali.io/docs/installation/quick-start/>

<https://prometheus.io/docs/prometheus/latest/getting_started/>

<https://grafana.com/>

<https://www.elastic.co/>

<https://www.digitalocean.com/community/tutorials/how-to-set-up-an-elasticsearch-fluentd-and-kibana-efk-logging-stack-on-kubernetes>