Lab 7 Report

Course: Cloud_Computing
Student: ZhengYang
GitHub Repo: yangzheng.github.io — Repository for personal website project using GitHub Pages
Date: 2025.11.8

1. Objectives

Deploy and configure Istio service mesh on a Kubernetes cluster.
Install Kubernetes Gateway API CRDs and deploy the Bookinfo sample application.
Expose the application to external traffic via Istio Gateway and access the service.
Explore Istio telemetry tools including Kiali, Prometheus, Grafana, and Jaeger dashboards.
Document the full process with screenshots and detailed observations.

2. Environment & Prerequisites

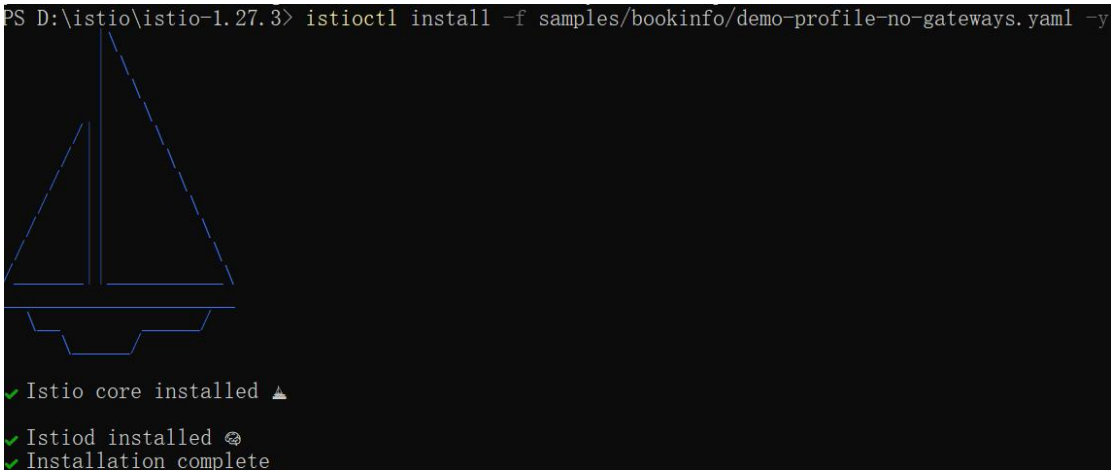
Operating System: windows11
Minikube Version: minikube version: v1.37.0
Kubectl Version: Client Version: v1.34.0; Kustomize Version: v5.7.1; Server Version: v1.34.0
Istio Version: v1.28.0
Driver: docker
Hardware: ⩾ 2 CPUs, ⩾ 4 GB RAM, ⩾ 30 GB disk space
Prerequisites: Running Kubernetes cluster (Minikube), curl utility installed, internet connectivity for downloading Istio and related images

3. Lab Steps & Observations

3.1 Download and Install Istio

Commands: curl -L https://istio.io/downloadIstio | sh -; cd istio-1.28.0; export PATH=$PWD/bin:$PATH; istioctl install -f samples/bookinfo/demo-profile-no-gateways.yaml -y; kubectl label namespace default istio-injection=enabled

Screenshots:

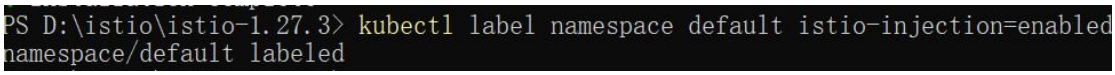


Observations: Istio installation completed successfully with core components and Istiod deployed. The default namespace is labeled to enable automatic sidecar injection, ensuring future deployments include Istio Envoy proxies.

3.2 Install Kubernetes Gateway API CRDs

Commands: kubectl get crd gateways.gateway.networking.k8s.io &> /dev/null || { kubectl kustomize "github.com/kubernetes-sigs/gateway-api/config/crd?ref=v1.4.0" | kubectl apply -f -; }

Screenshots:

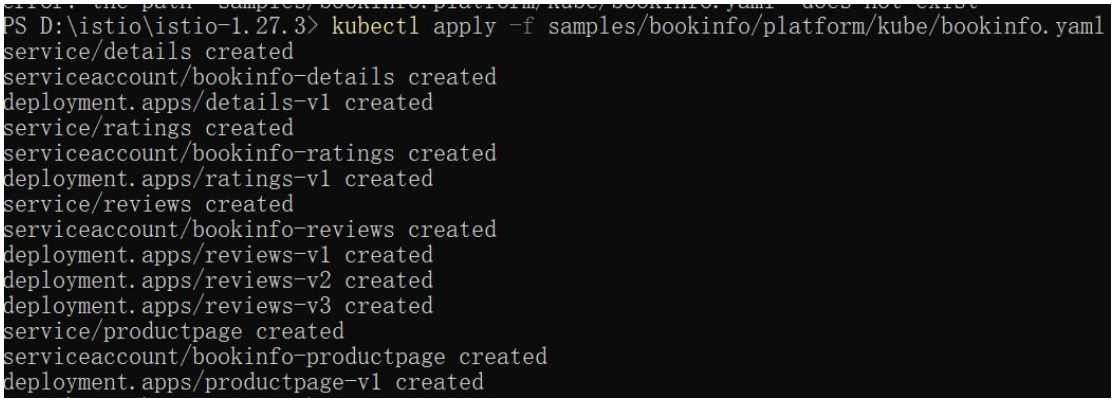


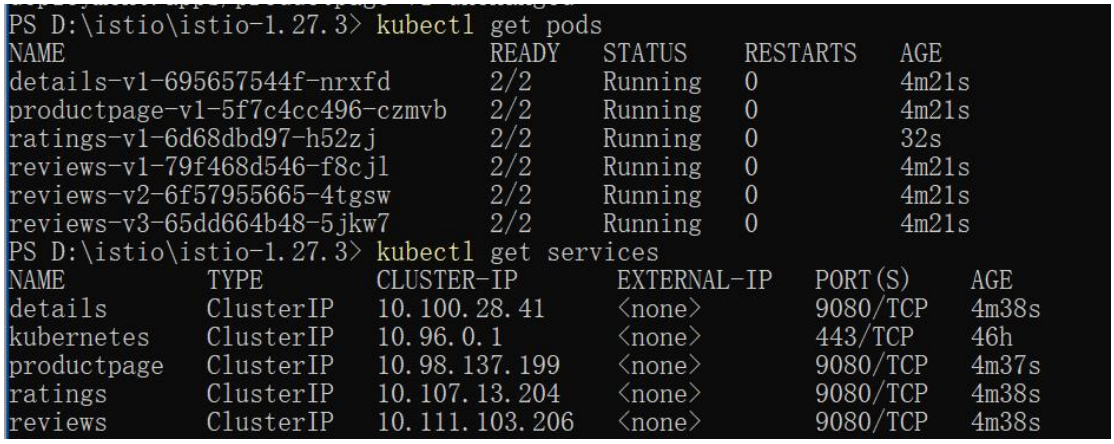
Observations: Kubernetes Gateway API CRDs were installed successfully as they were not present by default in the cluster. This step enables the use of Gateway API for traffic management in Istio.

3.3 Deploy Bookinfo Sample Application

Commands: kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml; kubectl get services; kubectl get pods; kubectl exec "$(kubectl get pod -l app=ratings -o jsonpath='{.items[0].metadata.name}')" -c ratings -- curl -sS productpage:9080/productpage | grep -o "<title>.*</title>"

Screenshots:



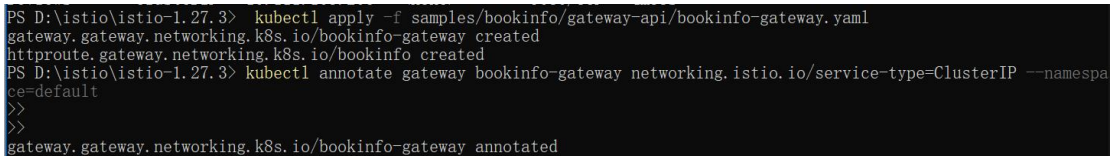


Observations: All Bookinfo services (details, ratings, reviews, productpage) and corresponding deployments were created successfully. Each pod shows 2/2 ready status, indicating the Istio sidecar proxy was injected automatically. The curl command verified the application is running within the cluster with the expected title.

3.4 Expose Application via Istio Gateway

Commands: kubectl apply -f samples/bookinfo/gateway-api/bookinfo-gateway.yaml; kubectl annotate gateway bookinfo-gateway networking.istio.io/service-type=ClusterIP --namespace=default; kubectl get gateway; kubectl port-forward svc/bookinfo-gateway-istio 8080:80

Screenshots:



Observations: The Bookinfo Gateway and HTTP route were created successfully, with the gateway status showing "PROGRAMMED: True". Annotating the gateway set the service type to ClusterIP, eliminating the need for an external load balancer. Port forwarding to the gateway service established a local connection to the cluster.

3.5 Access the Bookinfo Application

Commands: [No additional commands beyond port-forward; access via browser at http://localhost:8080/productpage]

Screenshots:

```
C:\Users\zy>kubectl port-forward svc/bookinfo-gateway-istio 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

Observations: The application was accessible via the local port 8080, displaying the full Bookinfo interface including book summary, details, and reviews. Refreshing the page rotated reviews across different versions of the reviews service, demonstrating Istio's traffic distribution capabilities.

3.6 Explore Istio Telemetry Dashboards

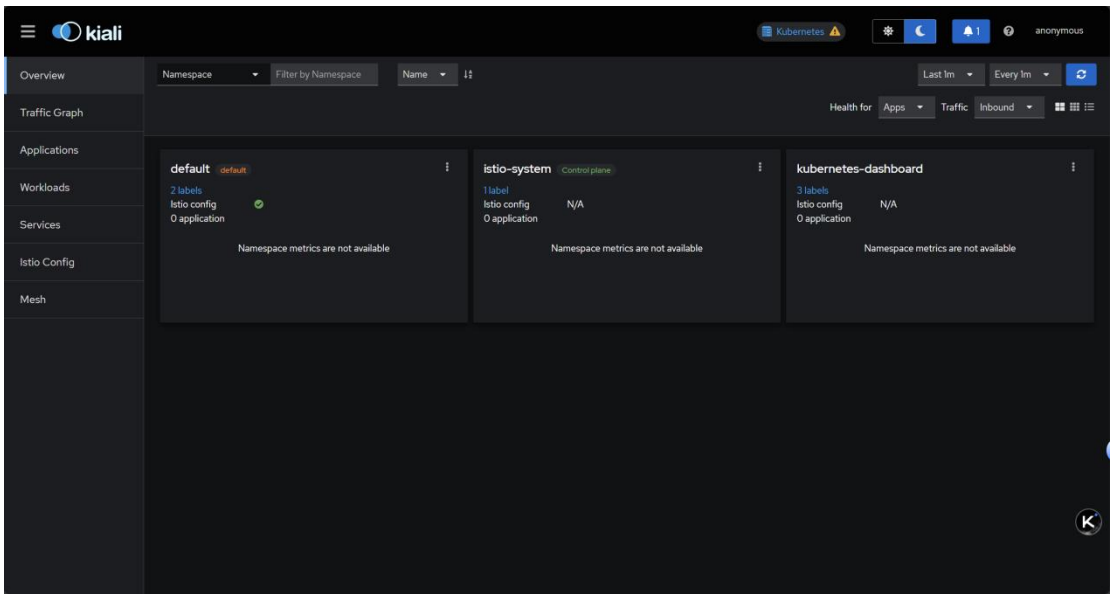
Commands: kubectl apply -f samples/addons/kiali.yaml; kubectl rollout status deployment/kiali -n istio-system; istioctl dashboard kiali; for i in $(seq 1 100); do curl -s -o /dev/null "http://localhost:8080/productpage"; done

Screenshots:

```
D:\istio\istio-1.27.3>kubectl apply -f samples/addons/kiali.yaml
serviceaccount/kiali created
configmap/kiali created
clusterrole.rbac.authorization.k8s.io/kiali created
clusterrolebinding.rbac.authorization.k8s.io/kiali created
service/kiali created
deployment.apps/kiali created

D:\istio\istio-1.27.3>kubectl rollout status deployment/kiali -n istio-system
Waiting for deployment "kiali" rollout to finish: 0 of 1 updated replicas are available.
deployment "kiali" successfully rolled out

D:\istio\istio-1.27.3>istioctl dashboard kiali
http://localhost:20001/kiali
```



Observations: Kiali and associated telemetry tools (Prometheus, Grafana, Jaeger) were deployed successfully. The Kiali dashboard visualized the service mesh topology, showing relationships

between Bookinfo services. Sending 100 requests generated trace data, which was visible in the dashboard, with 100% success rate for HTTP traffic.

4. Summary

What went well: Smooth Istio installation and configuration, successful automatic sidecar injection, seamless exposure of the Bookinfo application via Istio Gateway, and clear visualization of the service mesh through Kiali.

Challenges: Ensuring sufficient RAM for running both the Kubernetes cluster and Istio components, troubleshooting port forwarding conflicts, and waiting for trace data to appear in Kiali

Key Takeaways: Understanding the role of Istio as a service mesh for traffic management and observability, the importance of sidecar proxies for intercepting and managing service-to-service communication, how Gateway API enables controlled external access to mesh services, and the value of telemetry tools for monitoring mesh health and traffic flow.