

Secure and Resilient Kafka- Based Messaging System with Istio-Driven Security



Developed by: Banda Tharun Reddy (51793A), Tesfaamanuel Dekebo
(41533A)

Università degli Studi di Milano
Cloud Computing Project

Project Objectives

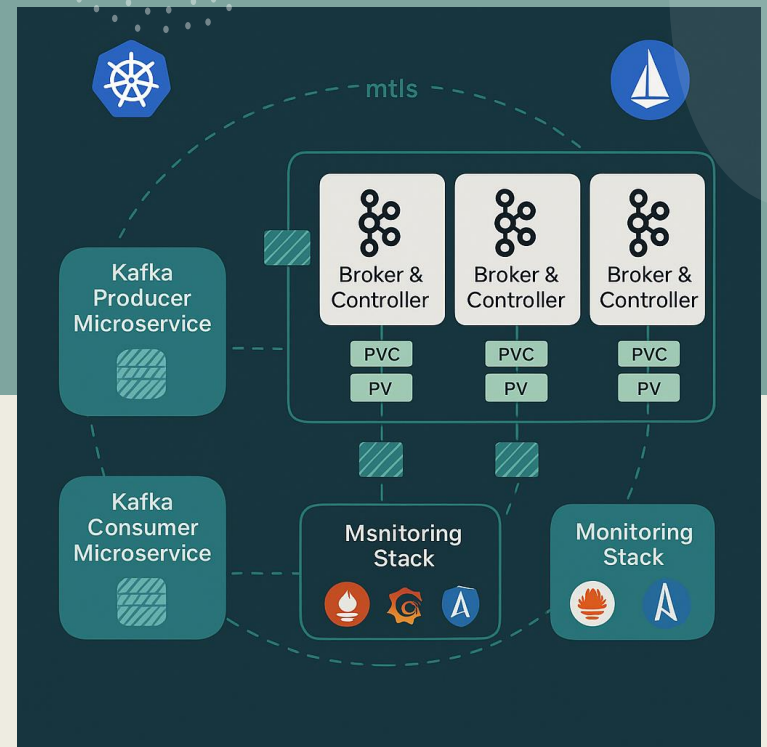
Deploy	Deploy Highly Available , fault-tolerant Kafka .
Message transmission	Simulate message flow between Producer and consumer microservices through kafka.
Enable Security	Implement security between kafka and clients with authentication and encryption.
failure recovery	Simulate failures for HA validation

Technology Stack

Component	Technology
Messaging Queue	Apache Kafka (Raft-based)
Orchestration	Kubernetes
Service Mesh	Istio
Microservices	Bitnami Kafka Docker Image
Deployment	Helm, YAML configs
Storage	Kubernetes PVCs
Observability	Prometheus, Grafana, Kiali

System Architecture

- Kafka with 3 Raft-based controller nodes
- Persistent storage via PVCs
- Producer & Consumer microservices
- Istio for secure communication (mTLS)
- Observability stack integrated



Kafka Deployment Details

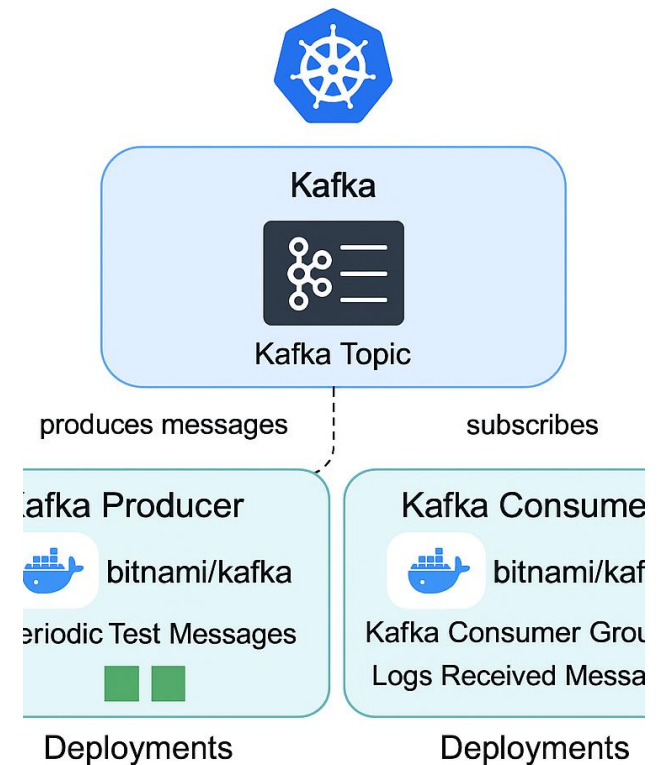
- Deployed Kafka using Bitnami Helm charts for a modular and parameterized setup.
- Provisioned a 3-node Kafka cluster using the Raft consensus protocol.
- Configured each Kafka node to act as both a controller and broker.
- Enabled Raft protocol to ensure consistent metadata management and high availability.
- Utilized Kubernetes StatefulSets for stable network identities (e.g., kafka-0, kafka-1, kafka-2).
- Attached PersistentVolumeClaims (PVCs) to each pod for durable, stateful storage.



shutterstock.com - 2429890337

Kafka Microservices

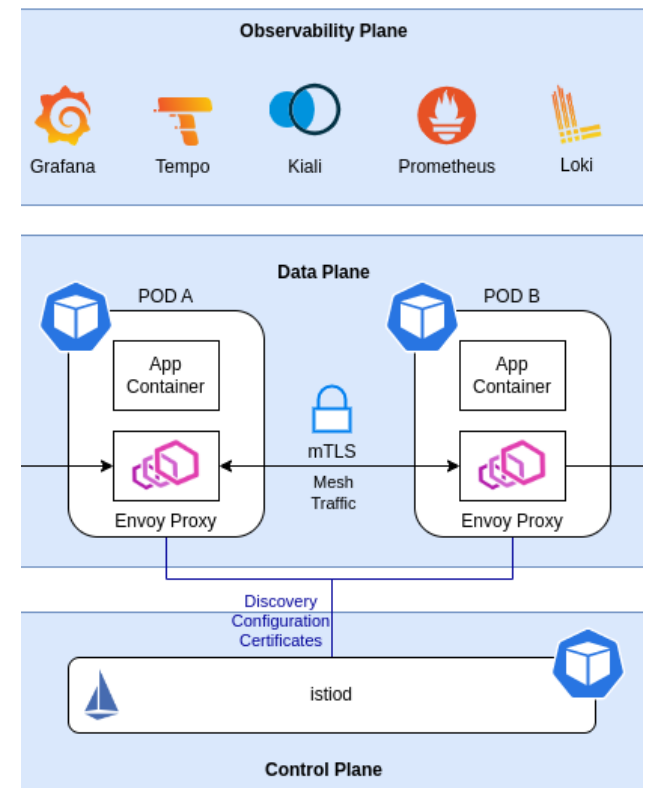
- Developed two microservices: a **Producer** that sends periodic test messages to a Kafka topic, and a **Consumer** that subscribes to the topic and logs received messages.
- Deployed both microservices on **Kubernetes as Deployments**, enabling **auto-healing** and **high availability** through replica management.
- Used the **Bitnami Kafka Docker image** for lightweight and production-ready containers.
- Employed **Kafka topics** as a decoupled communication channel between the Producer and Consumer.
- Configured the Consumer with a **Kafka Consumer Group** to enable **parallel message processing** and **load-balanced consumption** across multiple instances.



Security with Istio

- **Istio Service Mesh** integrated into the Kubernetes cluster to enable **zero-trust architecture** with automatic encryption, authentication, and traffic control for Kafka producers, consumers, and brokers.
- Enabled **mutual TLS (mTLS)** at the namespace level for **transparent pod-to-pod encryption** via Envoy sidecars, ensuring Kafka message confidentiality and integrity.
- **Authentication and Authorization** enforced using **SPIFFE-based identities** and **Istio AuthorizationPolicies**, allowing only trusted Kafka services and blocking unauthorized traffic.
- **Certificate management simplified**: no manual TLS setup needed; Istio's Citadel (or Istiod) handled **automatic certificate provisioning, rotation, and expiration**.

Istio Sidecar Architecture



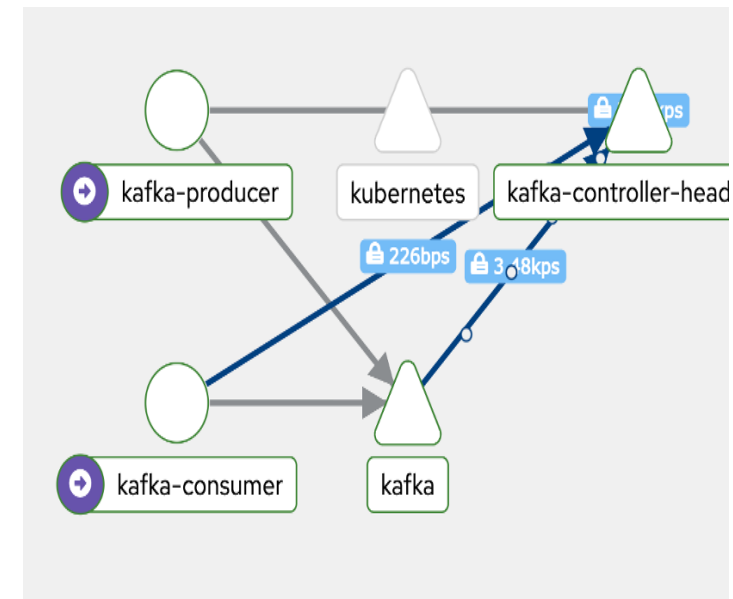
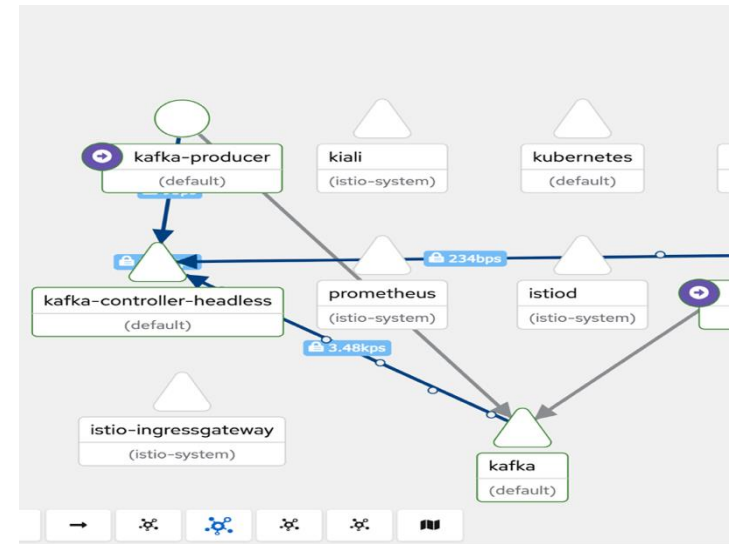
Security with Istio

- Kiali provides real-time visualization of service-to-service traffic in the Kubernetes cluster, validating secure, encrypted (mTLS) communication and monitoring message flow among Kafka components and microservices.

Key Features Demonstrated:

- Live Flow Visualization between Producer, Broker, and Consumer
- mTLS Encryption Status indicated by padlocks and colored arrows
- Traffic Metrics (bps, kps) for message volume and health
- Access & Routing Policies: Only authorized traffic is allowed
- Cluster Mesh View: Complete service topology including monitoring tools

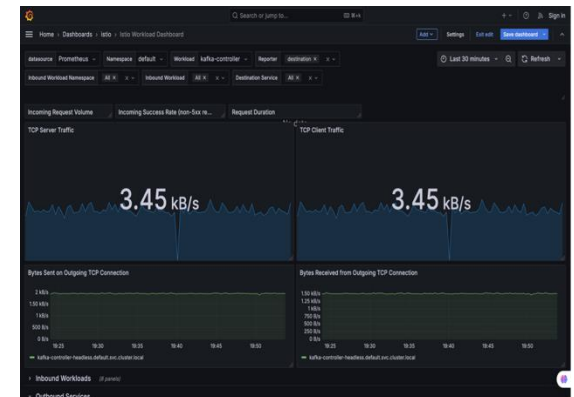
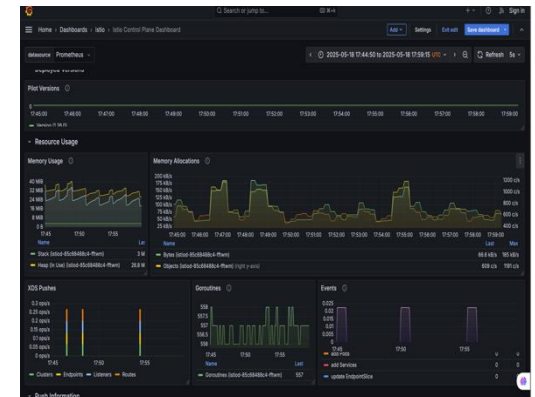
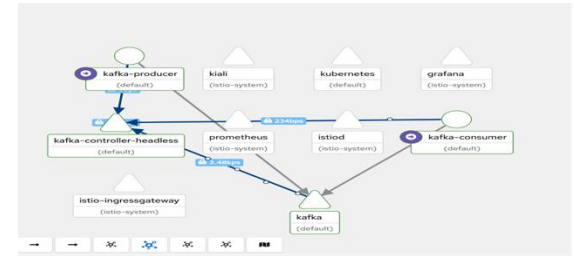
Summary: Kiali ensures verification of secure Kafka deployment, monitoring, and policy enforcement, demonstrating the effectiveness of Istio mTLS and RBAC.



Observability Tools

- A comprehensive observability stack was deployed for real-time monitoring and transparency across the Kafka-based system:
 - Prometheus: Scrapes metrics from Kafka, Kubernetes, and Istio, enabling monitoring of broker health, throughput, partition lag, resource usage, mTLS status, latency, and error rates.
 - Grafana: Provides rich dashboards to visualize performance data — Kafka topic activity, consumer lag, cluster health, resource usage, and Istio telemetry.
 - Kiali: Offers a live, graphical view of service-to-service traffic with mTLS status, request rates, routing, and RBAC validation. Essential for debugging, bottleneck detection, and validating encrypted traffic.

Together, these tools allow immediate identification of issues, policy enforcement, and ensure operational transparency for DevOps teams.



Non-Functional Demonstrations

- Fault Tolerance: Kafka broker deletion → no message loss
- Self-Healing: Consumer pod crash → auto-recovery
- Security: Unauthorized pod access blocked (403 error)

Fault Tolerance: Kafka broker deletion → no message loss

- Before test
- After test

The screenshot shows a terminal window with the following content:

```
--list
my-consumer-group
I have no name[kafka-producer-fc49cccc8-jf7f]:/$ kafka-topics.sh \
--bootstrap-server $BOOTSTRAP \
--list
my-consumer-offsets
my-topic
test
I have no name[kafka-producer-fc49cccc8-jf7f]:/$ kafka-console-producer.sh --bootstrap-server $BOOTSTRAP --topic my-topic
>d
>d
>hi
>
```

Messages received by kafka-client before the kafka pod deletion:

```
NAME READY STATUS RESTARTS AGE
kafka-client-6b574cc759-b6qgh 2/2 Running 0 15s
kafka-client-6b574cc759-zzdc2 2/2 Running 0 15s
kafka-controller-0 2/2 Running 20 (15h ago) 12d
kafka-controller-1 2/2 Running 20 (15h ago) 12d
kafka-controller-2 2/2 Running 20 (15h ago) 12d
kafka-producer-fc49cccc8-jf7f 2/2 Running 0 152m
```

kafka server running

The screenshot shows a terminal window with the following content:

```
--list
my-consumer-offsets
my-topic
test
I have no name[kafka-producer-fc49cccc8-jf7f]:/$ kafka-console-producer.sh --bootstrap-server $BOOTSTRAP --topic my-topic
>d
>d
>hi
>hello
>hi
>hello
>hi
>hello
>nsj
>jsj
>
```

sending messages after the client pod deletion

client connected to the new pod, but he haven't lost the unread messages which received during the pod deletion

here we can see the client pod deleted successfully and k8's started creating new pod

deleted the kafka client pod

Self-Healing: Consumer pod crash → auto-recovery

- Before test
- After test

Terminal 1:

```
--list
my-consumer-group
I have no name[kafka-producer-fc49cccc8-jfj7f:/]# kafka-topics.sh \
--bootstrap-server $BOOTSTRAP \
--list
--consumer_offsets
my-topic
test
I have no name[kafka-producer-fc49cccc8-jfj7f:/]# kafka-console-producer.sh \
--bootstrap-server $BOOTSTRAP --topic my-topic
>d
>d
>hi
>
```

Terminal 2:

```
NAME                READY   STATUS    RESTARTS   AGE
kafka-client-0b574cc759-b6qgh  2/2     Running   0           15s
kafka-client-0b574cc759-z2dc2  2/2     Running   0           15s
kafka-controller-0             2/2     Running   0 (15h ago)  12d
kafka-controller-1             2/2     Running   0 (15h ago)  12d
kafka-controller-2             2/2     Running   0 (15h ago)  12d
kafka-producer-fc49cccc8-jfj7f  2/2     Running   0           15m
(base) THARUN@MacBook-Pro-M2 ~ % kubectl logs -f kafka-client-0b574cc759-z2dc2
Starting Kafka consumer in consumer group: my-consumer-group
d
d
hi
```

Terminal 3:

```
NAME                READY   STATUS    RESTARTS   AGE
kafka-client-0b574cc759-b6qgh  2/2     Running   0           23m
kafka-client-0b574cc759-z2dc2  2/2     Running   0           23m
kafka-controller-0             2/2     Running   0 (15h ago)  12d
kafka-controller-1             2/2     Running   0 (15h ago)  12d
kafka-controller-2             2/2     Running   0 (15h ago)  12d
kafka-producer-fc49cccc8-jfj7f  2/2     Running   0           17m
(base) THARUN@MacBook-Pro-M2 ~ % kubectl get pods -n 100x13
NAME                READY   STATUS    RESTARTS   AGE
kafka-client-0b574cc759-b6qgh  2/2     Running   0           23m
kafka-client-0b574cc759-z2dc2  2/2     Running   0           23m
kafka-controller-0             2/2     Running   0 (15h ago)  12d
kafka-controller-1             2/2     Running   0 (15h ago)  12d
kafka-controller-2             2/2     Running   0 (15h ago)  12d
kafka-producer-fc49cccc8-jfj7f  2/2     Running   0           17m
```

Terminal 4:

```
NAME                READY   STATUS    RESTARTS   AGE
kafka-client-0b574cc759-b6qgh  2/2     Running   0           26m
kafka-client-0b574cc759-z2dc2  2/2     Running   0           26m
kafka-controller-0             2/2     Running   0 (15h ago)  12d
kafka-controller-1             2/2     Running   0 (15h ago)  12d
kafka-controller-2             2/2     Running   0 (15h ago)  12d
kafka-producer-fc49cccc8-jfj7f  2/2     Running   0           30m
(base) THARUN@MacBook-Pro-M2 ~ % kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
kafka-client-0b574cc759-b6qgh  2/2     Running   0           26m
kafka-client-0b574cc759-z2dc2  2/2     Running   0           26m
kafka-controller-0             2/2     Running   0 (15h ago)  12d
kafka-controller-1             2/2     Running   0 (15h ago)  12d
kafka-controller-2             2/2     Running   0 (15h ago)  12d
kafka-producer-fc49cccc8-jfj7f  2/2     Running   0           30m
```

Annotations:

- client messages sent before client pod deletion
- producer messages received before client pod deletion
- client pods are running state

Terminal 1:

```
my-topic
I have no name[kafka-producer-fc49cccc8-jfj7f:/]# kafka-console-producer.sh \
--bootstrap-server $BOOTSTRAP --topic my-topic
>d
>hi
>hello
>hi
>hi
>
Processed a total of 5 messages
(base) THARUN@MacBook-Pro-M2 ~ % kubectl logs -f kafka-client-0b574cc759-ckt9j
Starting Kafka consumer in consumer group: my-consumer-group
d
d
hi
hi
hi
nsj
nsj
jsh
>
```

Terminal 2:

```
NAME                READY   STATUS    RESTARTS   AGE
kafka-client-0b574cc759-b6qgh  2/2     Running   0           5s
kafka-client-0b574cc759-z2dc2  2/2     Running   0           37s
kafka-controller-0             2/2     Running   0 (15h ago)  12d
kafka-controller-1             2/2     Running   0 (15h ago)  12d
kafka-controller-2             2/2     Running   0 (15h ago)  12d
kafka-producer-fc49cccc8-jfj7f  2/2     Running   0           17m
(base) THARUN@MacBook-Pro-M2 ~ % kubectl get pods -n 100x13
NAME                READY   STATUS    RESTARTS   AGE
kafka-client-0b574cc759-b6qgh  2/2     Running   0           23m
kafka-client-0b574cc759-z2dc2  2/2     Running   0           23m
kafka-controller-0             2/2     Running   0 (15h ago)  12d
kafka-controller-1             2/2     Running   0 (15h ago)  12d
kafka-controller-2             2/2     Running   0 (15h ago)  12d
kafka-producer-fc49cccc8-jfj7f  2/2     Running   0           17m
```

Terminal 3:

```
NAME                READY   STATUS    RESTARTS   AGE
kafka-client-0b574cc759-b6qgh  2/2     Running   0           26m
kafka-client-0b574cc759-z2dc2  2/2     Running   0           26m
kafka-controller-0             2/2     Running   0 (15h ago)  12d
kafka-controller-1             2/2     Running   0 (15h ago)  12d
kafka-controller-2             2/2     Running   0 (15h ago)  12d
kafka-producer-fc49cccc8-jfj7f  2/2     Running   0           30m
(base) THARUN@MacBook-Pro-M2 ~ % kubectl delete pod kafka-client-0b574cc759-z2dc2
pod "kafka-client-0b574cc759-z2dc2" deleted
(base) THARUN@MacBook-Pro-M2 ~ % kubectl delete pod kafka-client-0b574cc759-z2dc2
pod "kafka-client-0b574cc759-z2dc2" deleted
(base) THARUN@MacBook-Pro-M2 ~ % kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
kafka-client-0b574cc759-b6qgh  2/2     Running   0           26m
kafka-client-0b574cc759-z2dc2  2/2     Running   0           26m
kafka-controller-0             2/2     Running   0 (15h ago)  12d
kafka-controller-1             2/2     Running   0 (15h ago)  12d
kafka-controller-2             2/2     Running   0 (15h ago)  12d
kafka-producer-fc49cccc8-jfj7f  2/2     Running   0           30m
```

Annotations:

- sending messages after the client pod deletion
- client connected to the new pod, but he haven't lost the unread messages which received during the pod deletion
- here you can see the client pod deleted successfully and k8s started creating new pod
- deleted the kafka client pod

• Security: Unauthorized pod access blocked (403 error)

```
Tharun -- zsh -- 113x17
-- kafka-client-6b574cc759-ckt9j 2/2 Running 0 27m
my kafka-controller-0 2/2 Running 20 (16h ago) 12d
te kafka-controller-1 2/2 Running 0 43m
I kafka-controller-2 2/2 Running 20 (16h ago) 12d
ce kafka-producer-fc49cccc8-jfj7f 2/2 Running 0 3h48m
> (base) THARUN@MacBook-Pro-M2 ~ % kubectl create namespace attacker
namespace/attacker created
> (base) THARUN@MacBook-Pro-M2 ~ % kubectl run attacker \
--restart='Never' \
--image=docker.io/bitnami/kafka:latest \
--namespace=attacker \
--command -- sleep infinity
pod/attacker created
> (base) THARUN@MacBook-Pro-M2 ~ % kubectl get pods -n attacker
NAME READY STATUS RESTARTS AGE
attacker 1/1 Running 0 117s
(base) THARUN@MacBook-Pro-M2 ~ %

Tharun -- kubectl exec -it attacker -n attacker -- bash -- 119x21
istio-system kiali-6d774d8bb8-76gxt 1/1 Running 10 (16h ago) 12d
istio-system prometheus-689cc795d4-bgp56 2/2 Running 14 (16h ago) 9d
kube-system coredns-668d6bf9bc-pm6bw 1/1 Running 11 (16h ago) 13d
kube-system etcd-minikube 1/1 Running 11 (16h ago) 13d
kube-system kube-apiserver-minikube 1/1 Running 11 (16h ago) 13d
kube-system kube-controller-manager-minikube 1/1 Running 12 (16h ago) 13d
kube-system kube-proxy 1/1 Running 11 (16h ago) 13d
kube-system kube-scheduler-minikube 1/1 Running 11 (16h ago) 13d
kube-system storage-provisioner 0/1 Error 0 11d
new kafka-producer1 0/1 Error 0 11d
(base) THARUN@MacBook-Pro-M2 ~ % kubectl exec -it attacker -- bash
Error from server (NotFound): pods "attacker" not found
(base) THARUN@MacBook-Pro-M2 ~ % kubectl exec -it attacker -n attacker -- bash
I have no name!@attacker:/$ kafka-topics.sh \
--bootstrap-server kafka.default.svc.cluster.local:9092 \
--list
Error while executing topic command : Timed out waiting for a node assignment. Call: listTopics
[2025-05-28 16:38:41,728] ERROR org.apache.kafka.common.errors.TimeoutException: Timed out waiting for a node assignmen
t. Call: listTopics
(org.apache.kafka.tools.TopicCommand)
I have no name!@attacker:/$
```

sucessfully deployed attacker pod in the new attacker namespace where istio is not configured for

Tried to login into attacker pod and attempted to connect to the kafka broker, here we can see the connection failed, where istio blocked the access

Conclusion

- • Kafka + Kubernetes + Istio achieves:
 - - Security (mTLS, RBAC)
 - - Resilience (auto-healing, fault tolerance)
 - - Observability (metrics, dashboards, tracing)
- • Production-grade messaging system
- • Repository: github.com/BandaTharun/Secure-and-Resilient-Kafka-Based-Messaging-System-with-Istio-Driven-Security-and-HA



Thank You