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

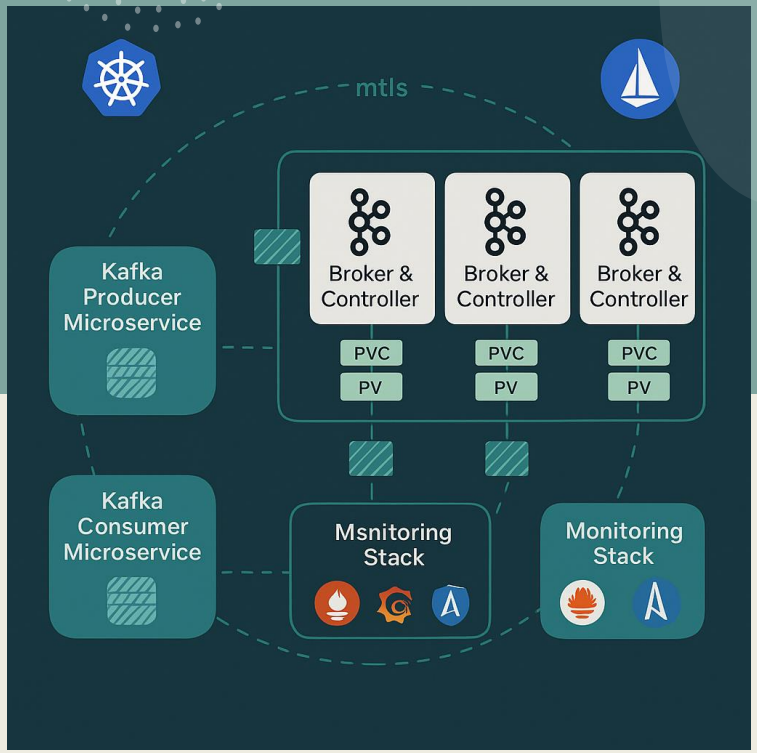
Deploy	Deploy Highly Available , fault-tolerant Kafka .
Message transmission	Simulate message flow between Producer and consumer microservices through kafka.
Enable Security	Implement security between kafka and clients with authentication and encryption.
failure recovery	Simulate failures for HA validation

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

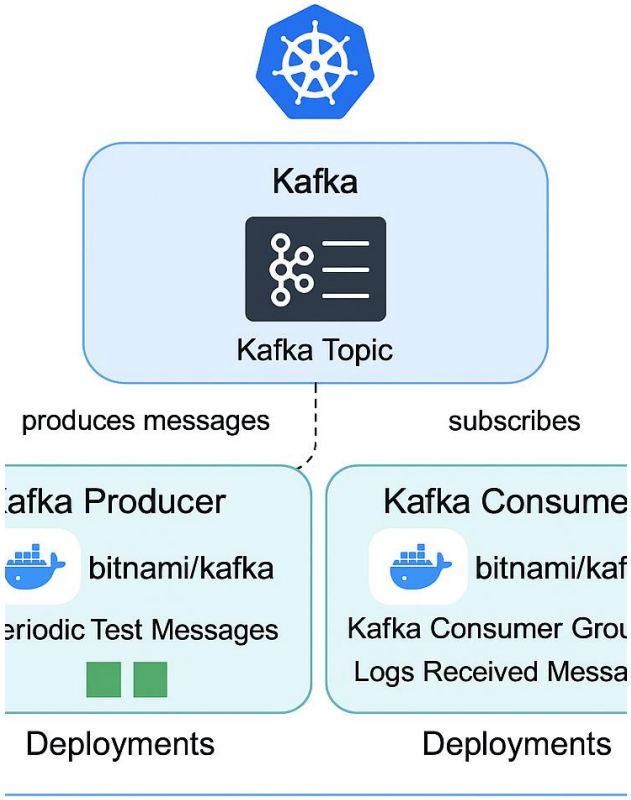


Kafka Deployment Details

- 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.

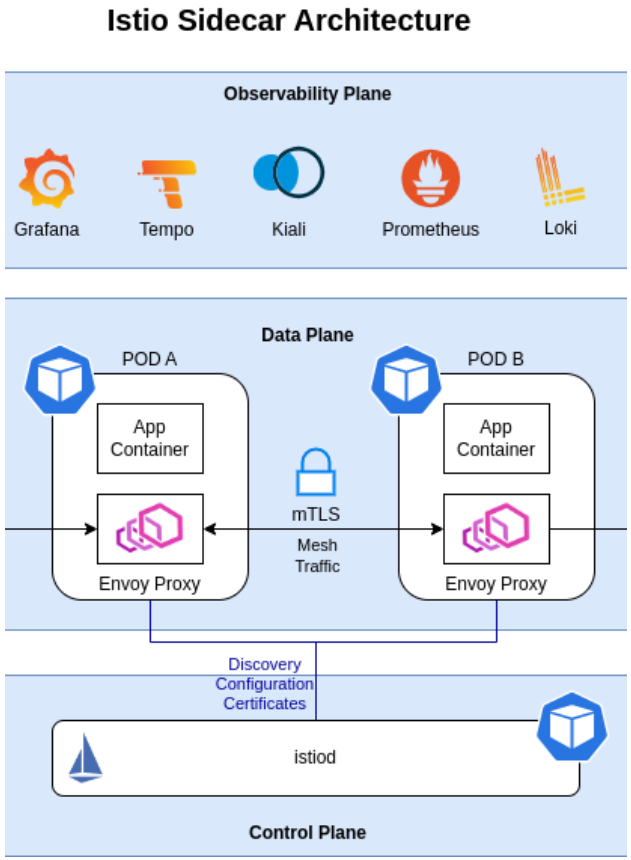
Kafka Microservices

- 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**.

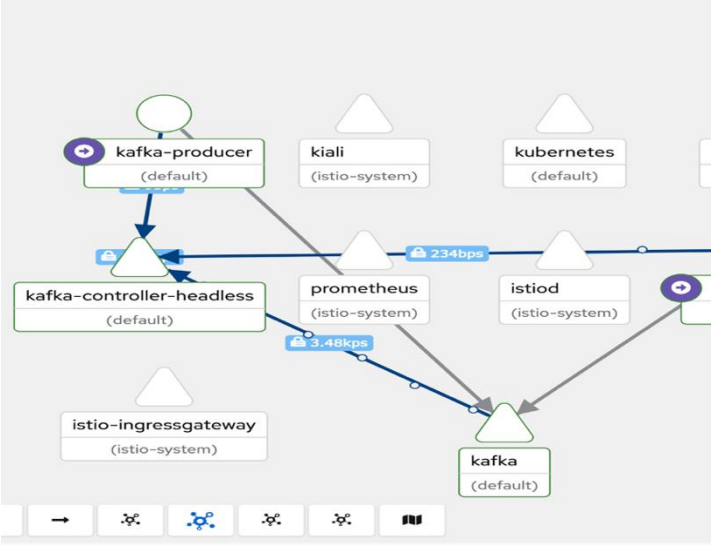


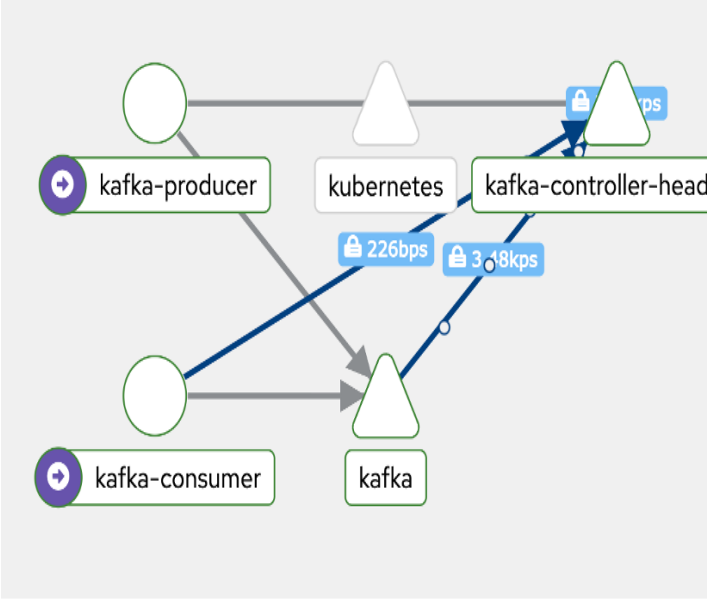
Security with Istio

- 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.

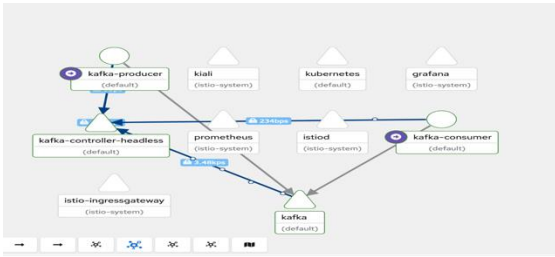
  Key Features Demonstrated:
  • Live Flow Visualization between Producer, Broker, and Consumer
  • mTLS Encryption Status indicated by padlocks and colored arrows
  • Traffic Metrics (bps, kps) for message volume and health
  • Access & Routing Policies: Only authorized traffic is allowed
  • Cluster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

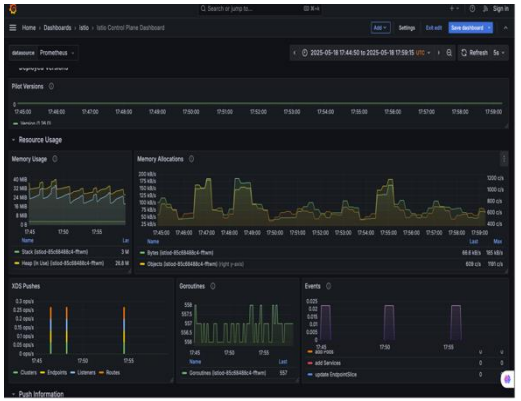
  Summary: Kiali ensures verification of secure Kafka deployment, monitoring, and policy enforcement, demonstrating the effectiveness of Istio mTLS and RBAC.

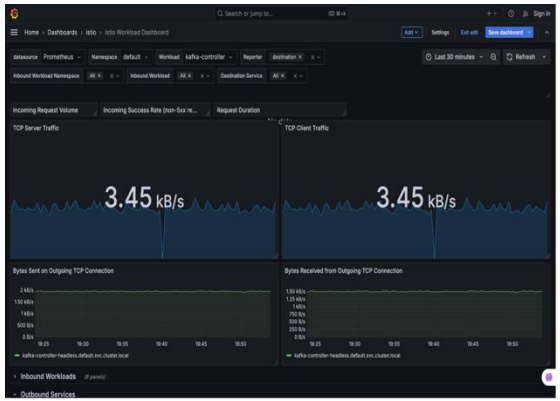




Observability Tools







- 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.

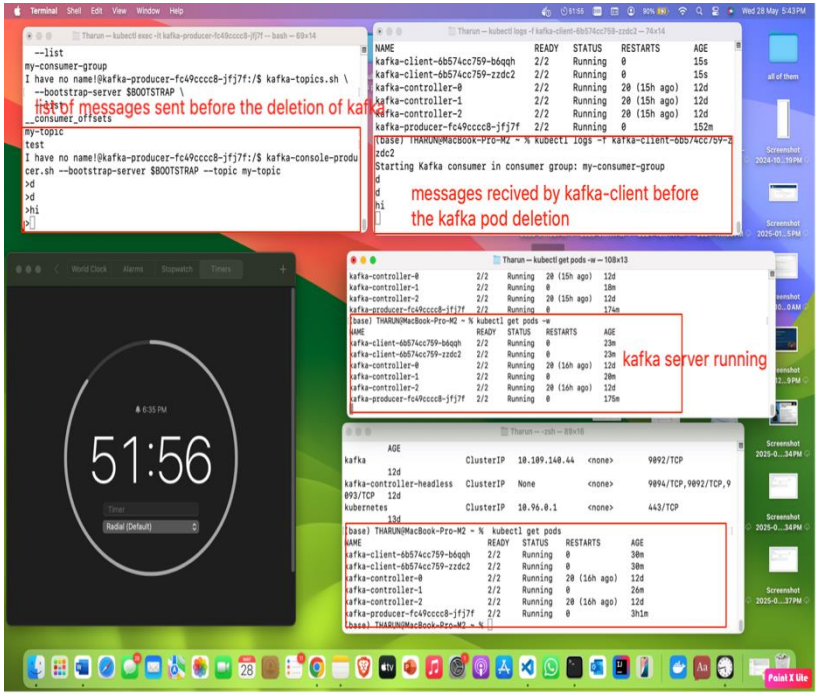
  Together, these tools allow immediate identification of issues, policy enforcement, and ensure operational transparency for DevOps teams.

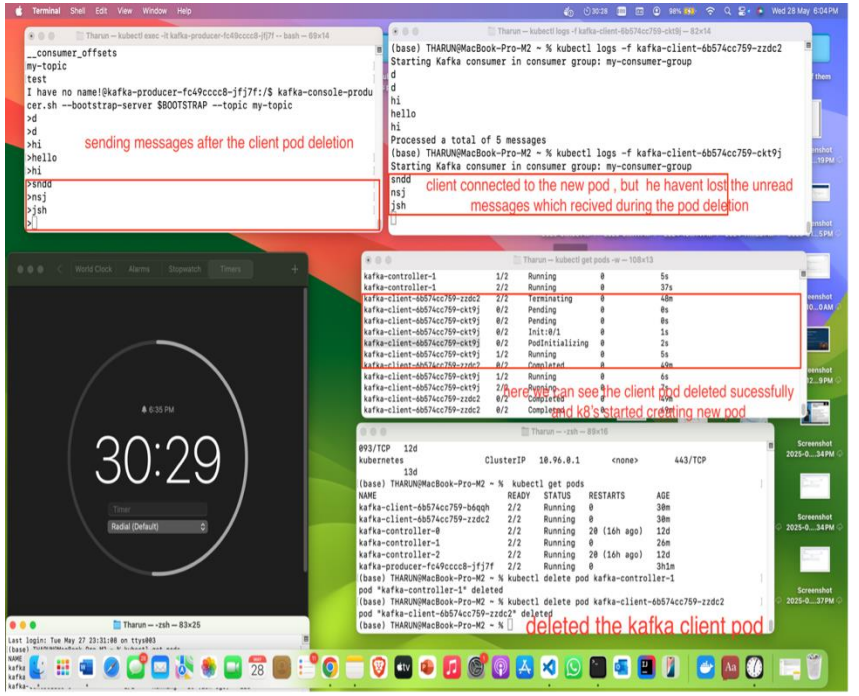
Non-Functional Demonstrations

- 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

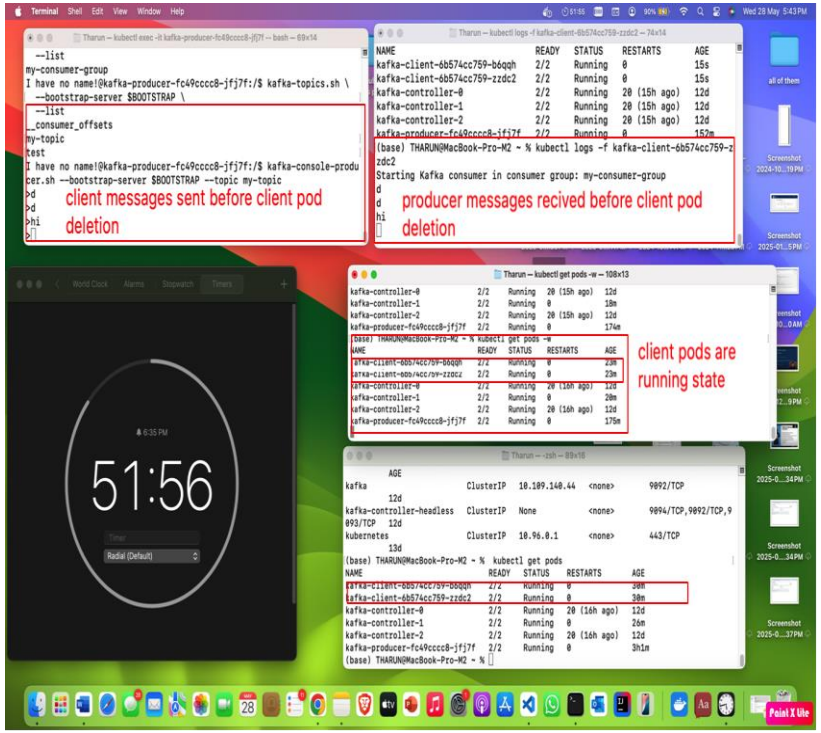
- Before test
- After test

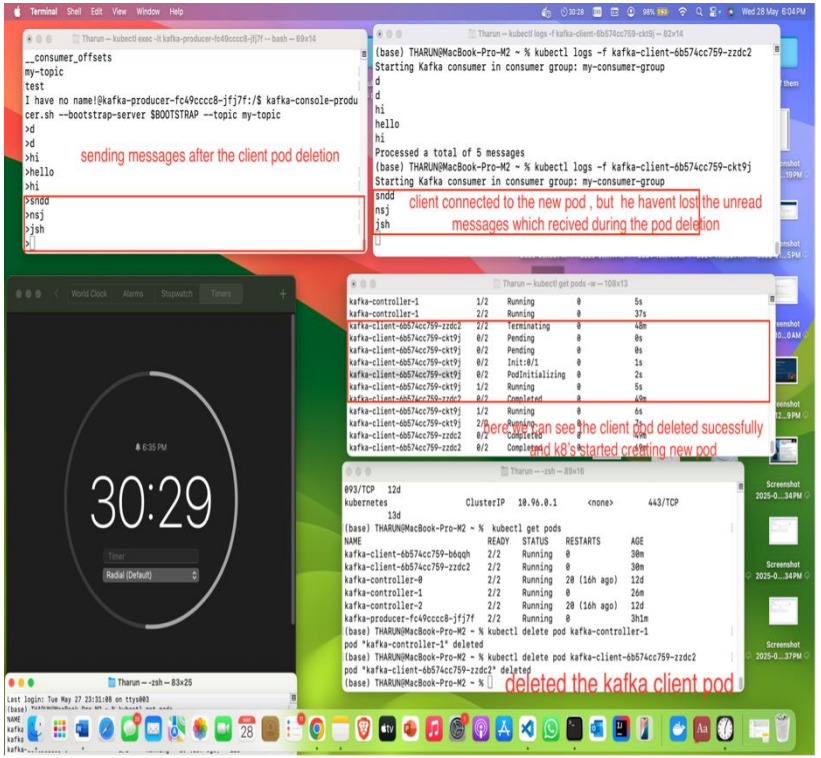




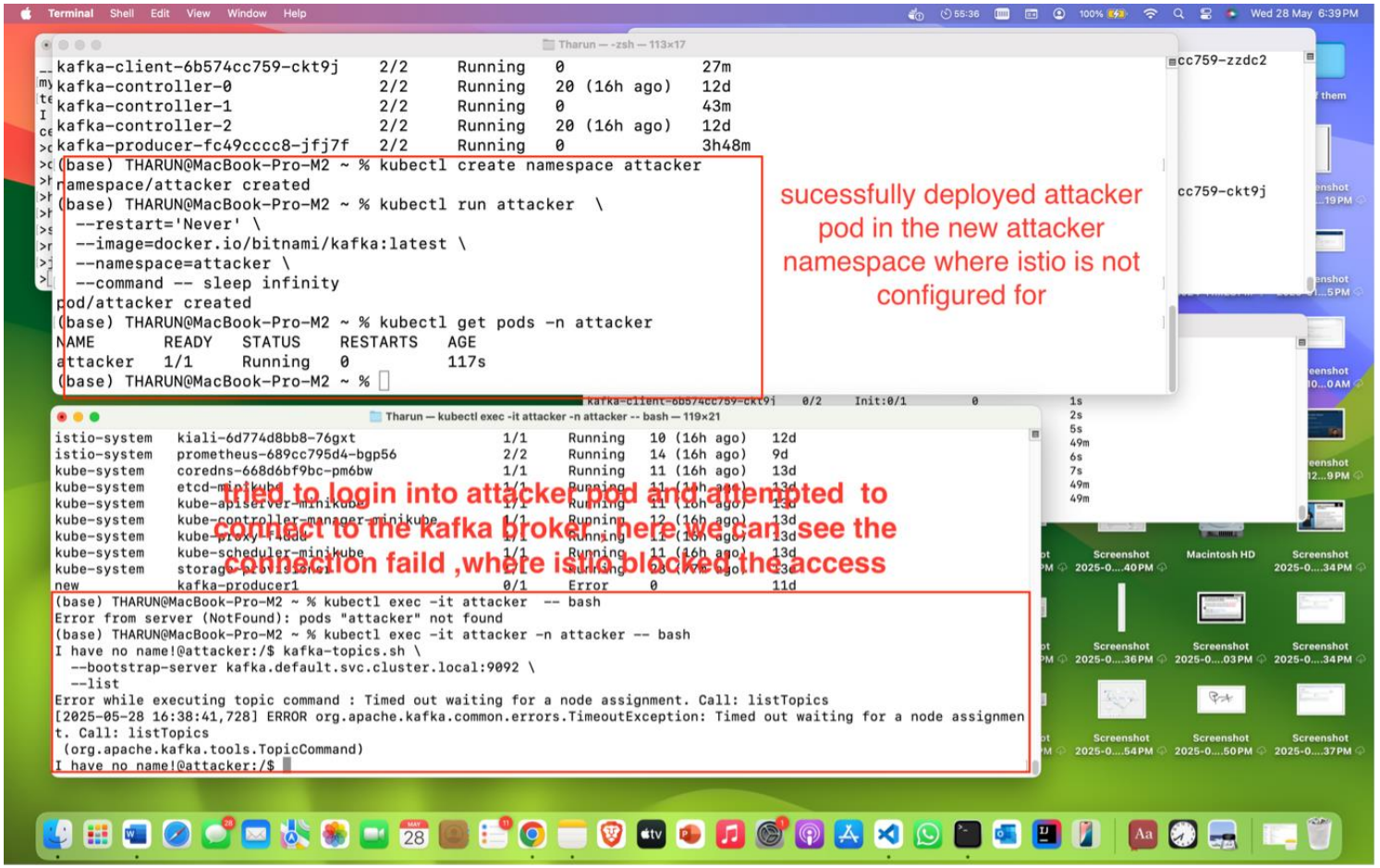
Self-Healing: Consumer pod crash → auto-recovery

- Before test
- After test





- Security: Unauthorized pod access blocked (403 error)



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