Frequently Asked Questions (FAQ)

1. Why did you choose Go for building this ETL pipeline?

Go provides high performance, native concurrency, and is well-suited for building lightweight, scalable microservices.

2. How does KRaft mode in Kafka differ from Zookeeper-based Kafka?

KRaft (Kafka Raft) eliminates the need for Zookeeper by using an internal Raft-based consensus protocol, simplifying deployment and improving performance.

3. Why is Kubernetes essential for this setup?

Kubernetes provides scalability, self-healing, rolling updates, and resource efficiency for managing distributed components like Kafka, MongoDB, and microservices.

4. What are the resource requirements for running this pipeline locally?

Ideally, a machine with **16GB RAM**, **4-core CPU**, and **Docker + Minikube** installed.

5. Is this pipeline production-ready?

The architecture is extendable to production, but for real-world use, you'd need to add logging, error handling, authentication, CI/CD, and observability layers.

6. Can I run this pipeline on a public cloud?

Yes. You can easily deploy this setup on AWS, GCP, or Azure using managed Kubernetes (like EKS/GKE/AKS) and integrate with managed Kafka and MongoDB services.

7. What are some monitoring tools recommended for this setup?

You can use **Prometheus + Grafana** for app metrics, **Kubernetes Dashboard**, and **Kiali** if you integrate service mesh like Istio.

8. What kind of data is being used in the demo?

A dummy data generator in Go simulates event streams that mimic real-world sensor data or logs for ETL processing.

9. How do I scale the ETL application dynamically?

Kubernetes Horizontal Pod Autoscaler (HPA) is used based on CPU/memory usage or custom metrics from Prometheus.

10. Where can I access the source code and setup instructions?

GitHub Repo: https://github.com/golangda/go-k8s-data-pipeline