Executive Summary

Kubernetes, known as K8s, has redefined cloud computing by revolutionizing the way applications are deployed, managed, and scaled. It stemmed from Google's expertise in handling large container workloads with Borg and Omega, signifying a major industry shift towards containerization and microservices.

Borg and Omega, Google's early systems, were the precursors to Kubernetes. Borg was vital for task scheduling and resource allocation, while Omega addressed Borg's monolithic challenges by introducing flexible scheduling. Kubernetes was birthed in 2014 from the lessons of Borg and Omega, embracing an open design and combining it with popular container technologies like Docker.

Docker's introduction in 2013 popularized container technology. The adoption of containers spurred a move to microservices architecture, where applications are segmented into small, independent services. This rise of containerized microservices necessitated a tool like Kubernetes for efficient management at scale. Kubernetes evolved to manage containers effectively, drawing from Google's experiences with Borg and Omega.

Kubernetes architecture includes:

- Kubernetes Master
    - API Server: Entry point for cluster
    - Controller Manager: Detects cluster state changes (e.g., pods dying) and tries to restore the cluster back to its original state
    - Scheduler: Decide which node the next pod will be spun up on but does NOT spin up the pod itself (kubelet does this)
    - ETCD: Brain of the cluster, Key-Value store of the cluster state
- Nodes: Worker machines, physical or virtual.
- Pods: Basic units housing one or several containers.
- Services: Defines a set of Pods and access policies.
- Deployment: Manages Pods and describes an application's life cycle.

Kubernetes excels in orchestration, with capabilities like:

- Scheduling and Resource Allocation: Auto-scheduling based on resource availability.
- Health Checks and Self-healing: Auto-restarting or rescheduling of failed Pods.
- Load Balancing and Service Discovery: Distributes network traffic for stability.
- Scaling and Rollouts: Supports automated/manual scaling and application updates.
- Configuration and Secret Management: Safely manages configurations and sensitive information.
- Networking: Manages IP addresses, subnets, and port allocations.

Demo Objective

To effectively demonstrate deployment on Kubernetes Cluster I developed a MLOps pipeline to be deployed on Kubernetes Cluster.

- **Step1**: Deploy a Machine learning model from Jupyter notebook into MLFlow Server as backend to track multiple experiments and deploy models
- **Step2**: Access the deployed model via Stereamlit Frontend (which is deployed on Kubernetes Cluster)
- **Step3:** Showcase the cluster health metrics using Grafana and Prometheus. Grafana is a interactive dashboard and Prometheus is used for metrics measurement of the cluster.