CS777 Term Paper Presentation

Exploration of K8s

Venkata Sandeep Yerra
Boston University

Evolution, Architecture and MLOps

Agenda

- Introduction
- Historical Background
- Evolution of Containers and Rise of Kubernetes
- Kubernetes Architecture
- Demo: MLOps
- Demo: Monitoring

Introduction

- What is a container?
  - Lightweight, portable box that has everything
    - App itself along with code
    - Libraries
    - Settings
- Why do we need a container
  - Differences in Operating systems and underlying infrastructure are abstracted
  - Much more light weight compared to Virtual machines which also ship operating system

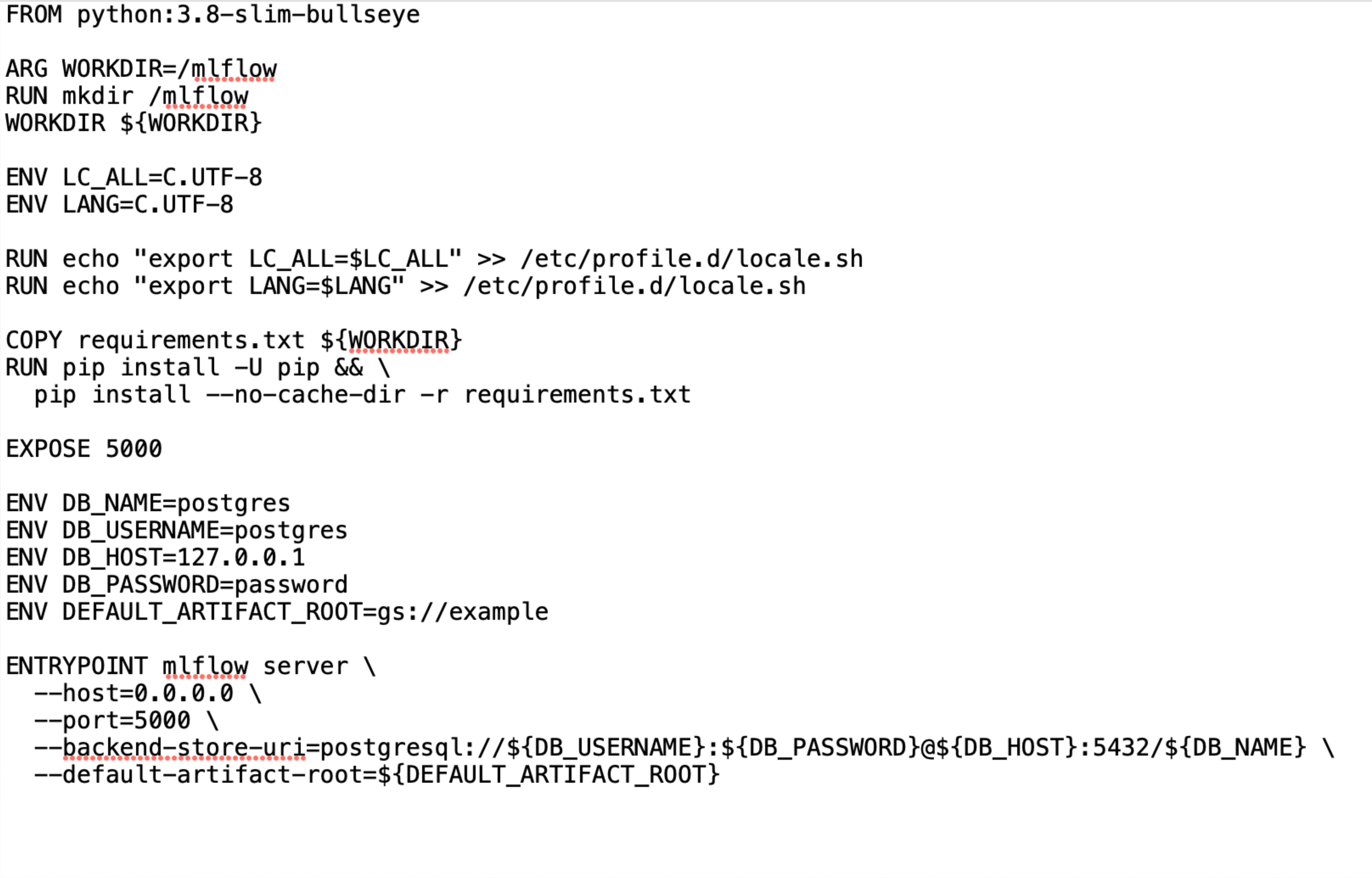
- What is Kubernetes
  - Framework to run distributed systems resiliently
- Why do we Kubernetes
  - Automates operational tasks of container management, deployment of applications, rolling changes, scaling, monitoring

Historical Background of Kubernetes

- The story of Kubernetes begins with its roots in a project at Google. Let's take a quick journey through its history
- **2003-2014: The Google Influence**
  - The concept underlying Kubernetes is largely based on Google's internal platform called Borg, which revolutionized the way software was deployed and managed at scale within Google since around 2003
  - Borg allowed Google to efficiently run containers – which are like lightweight, standalone packages of software – across their massive server fleets
  - **Omega** was a ground up software solution developed internally at google to be more flexible than Borg
- **2014: Birth of Kubernetes**
  - In mid-2014, Google decided to build an open-source version of their internal tools, leading to the birth of Kubernetes
  - Google partnered with the Linux Foundation to form the Cloud Native Computing Foundation (CNCF) to govern Kubernetes independently.
- **2015-Present: Growth and Adoption**
  - Kubernetes quickly gained popularity in the tech community for its efficiency in managing containerized applications
  - It became synonymous with microservices architectures and cloud-native technologies
- Today, Kubernetes is widely adopted by numerous companies, from startups to large enterprises, for orchestrating containers in production environment

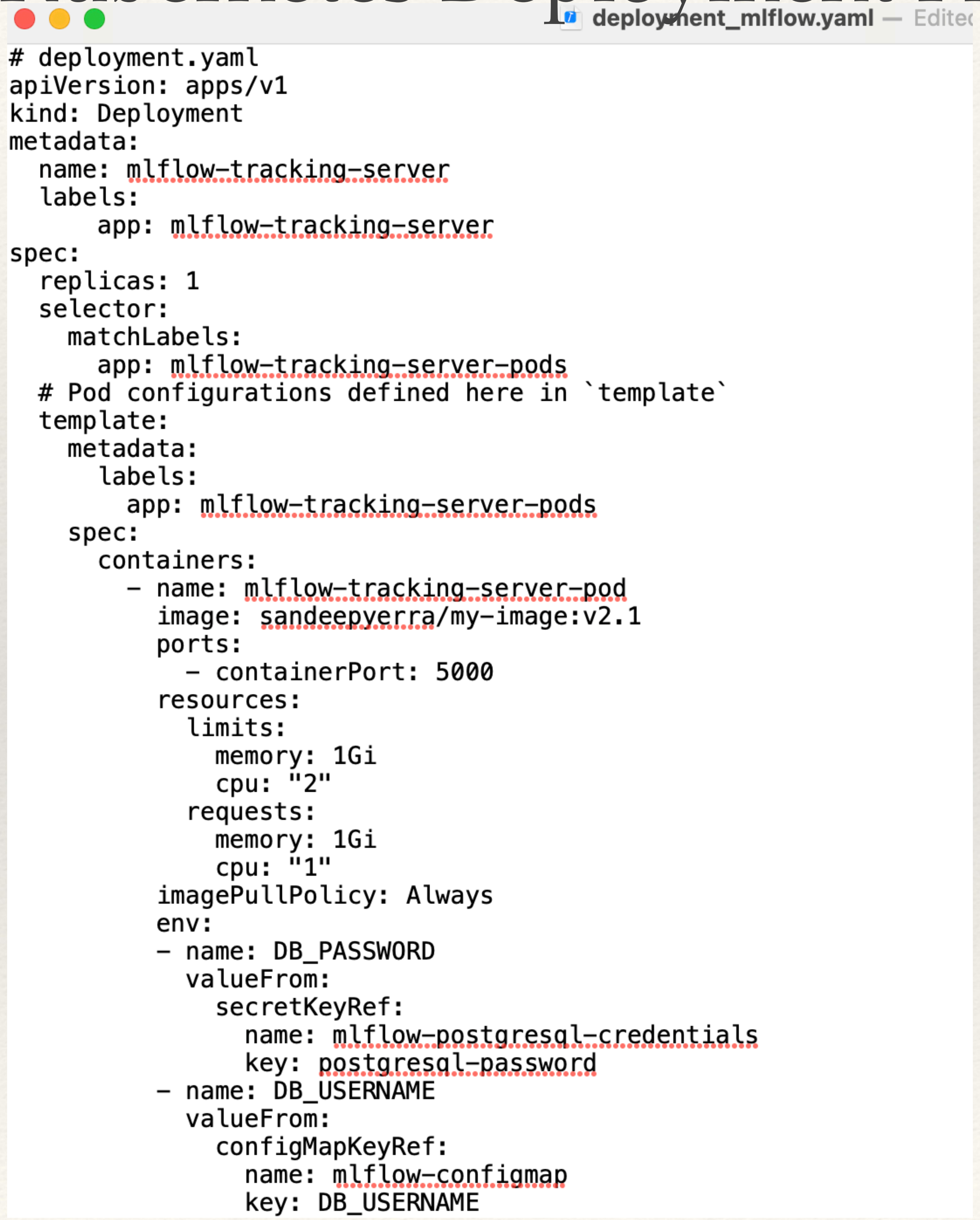
Examples

Dockerfile



```
FROM python:3.8-slim-bullseye

ARG WORKDIR=/mlflow
RUN mkdir /mlflow
WORKDIR ${WORKDIR}

ENV LC_ALL=C.UTF-8
ENV LANG=C.UTF-8

RUN echo "export LC_ALL=$LC_ALL" >> /etc/profile.d/locale.sh
RUN echo "export LANG=$LANG" >> /etc/profile.d/locale.sh

COPY requirements.txt ${WORKDIR}
RUN pip install -U pip && \
  pip install --no-cache-dir -r requirements.txt

EXPOSE 5000

ENV DB_NAME=postgres
ENV DB_USERNAME=postgres
ENV DB_HOST=127.0.0.1
ENV DB_PASSWORD=password
ENV DEFAULT_ARTIFACT_ROOT=gs://example

ENTRYPOINT mlflow server \
  --host=0.0.0.0 \
  --port=5000 \
  --backend-store-uri=postgresql://${DB_USERNAME}:${DB_PASSWORD}@${DB_HOST}:5432/${DB_NAME} \
  --default-artifact-root=${DEFAULT_ARTIFACT_ROOT}
```

Kubernetes Deployment File

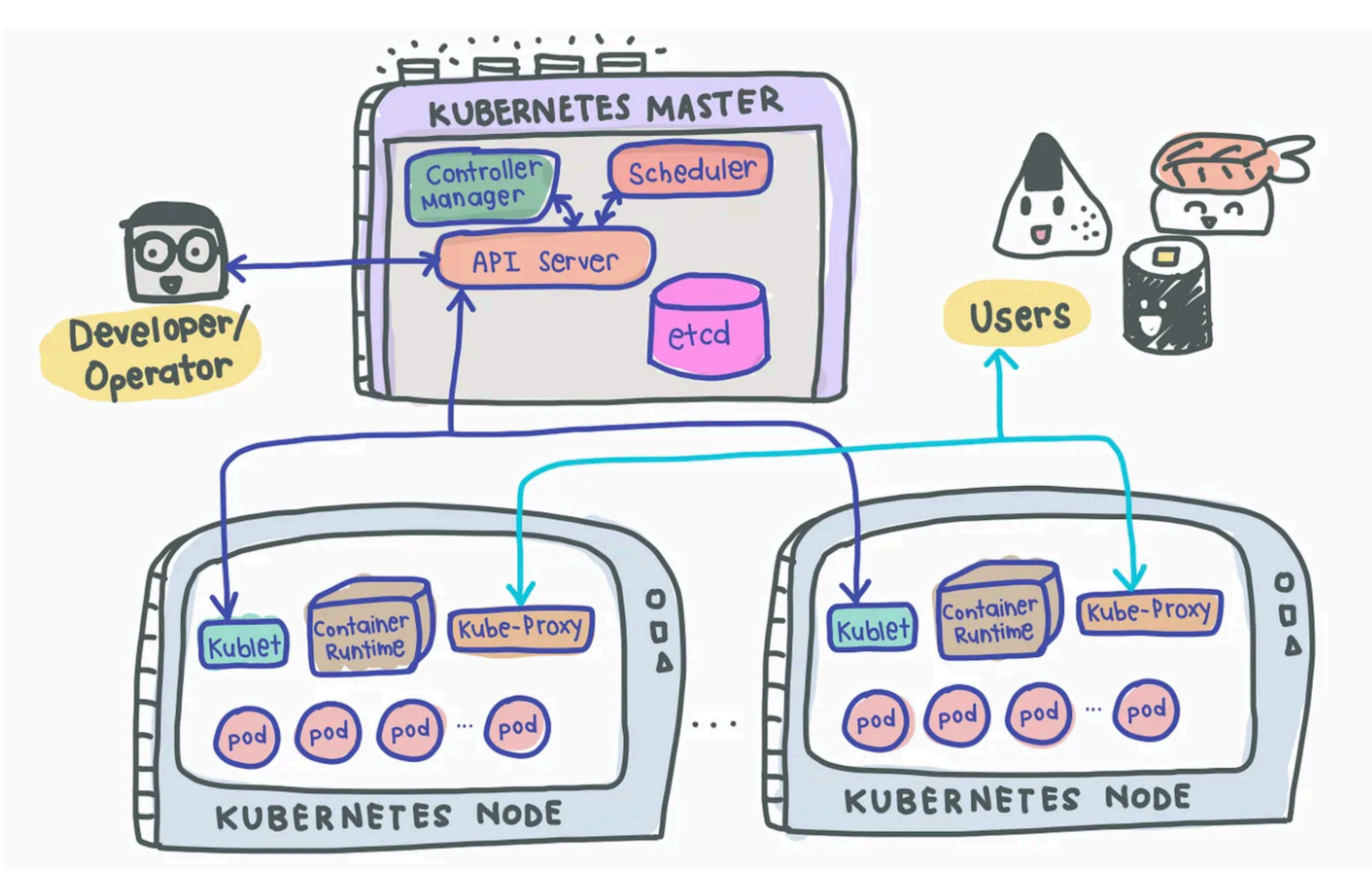


```
# deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mlflow-tracking-server
  labels:
      app: mlflow-tracking-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mlflow-tracking-server-pods
  # Pod configurations defined here in `template`
  template:
    metadata:
      labels:
        app: mlflow-tracking-server-pods
    spec:
      containers:
        - name: mlflow-tracking-server-pod
          image: sandeepverra/my-image:v2.1
          ports:
            - containerPort: 5000
          resources:
            limits:
              memory: 1Gi
              cpu: "2"
            requests:
              memory: 1Gi
              cpu: "1"
          imagePullPolicy: Always
          env:
          - name: DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mlflow-postgresql-credentials
                key: postgresql-password
          - name: DB_USERNAME
            valueFrom:
              configMapKeyRef:
                name: mlflow-configmap
                key: DB_USERNAME
```

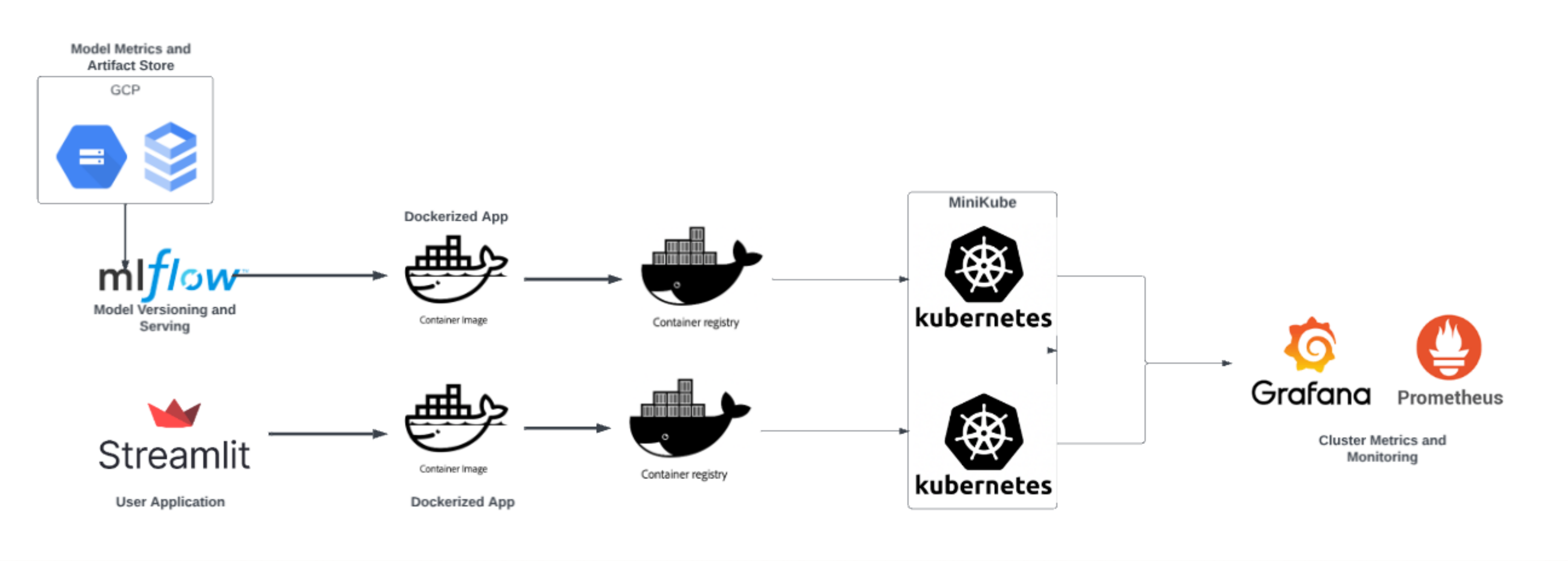
Kubernetes Architecture

Key Components

- Kubernetes Master
  - API Server
  - Controller Manager
  - Scheduler
  - ETCD
- Kubernetes Node
  - Kubelet
  - Container runtime
  - Kube-proxy
  - Pod



Demo Architecture



Demo Flow

- Flow of Demo
  - Kubernetes Features
    - Self-healing
    - Scaling
  - Model Development and Experiment Tracking
  - Model serving
  - Cluster Metrics Monitoring using Grafana and Prometheus

Thank You

I NEED TO KNOW WHY MOVING OUR APP TO THE CLOUD DIDN'T AUTOMATICALLY SOLVE ALL OUR PROBLEMS.
Dilbert.com @ScottAdamsSays
YOU WOULDN'T LET ME RE-ARCHITECT THE APP TO BE CLOUD-NATIVE.
JUST PUT IT IN CONTAINERS.
11-08-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel
YOU CAN'T SOLVE A PROBLEM JUST BY SAYING TECHY THINGS.
KUBERNETES.