Google Developer Experts

CI/CD and Containerization for Machine Learning

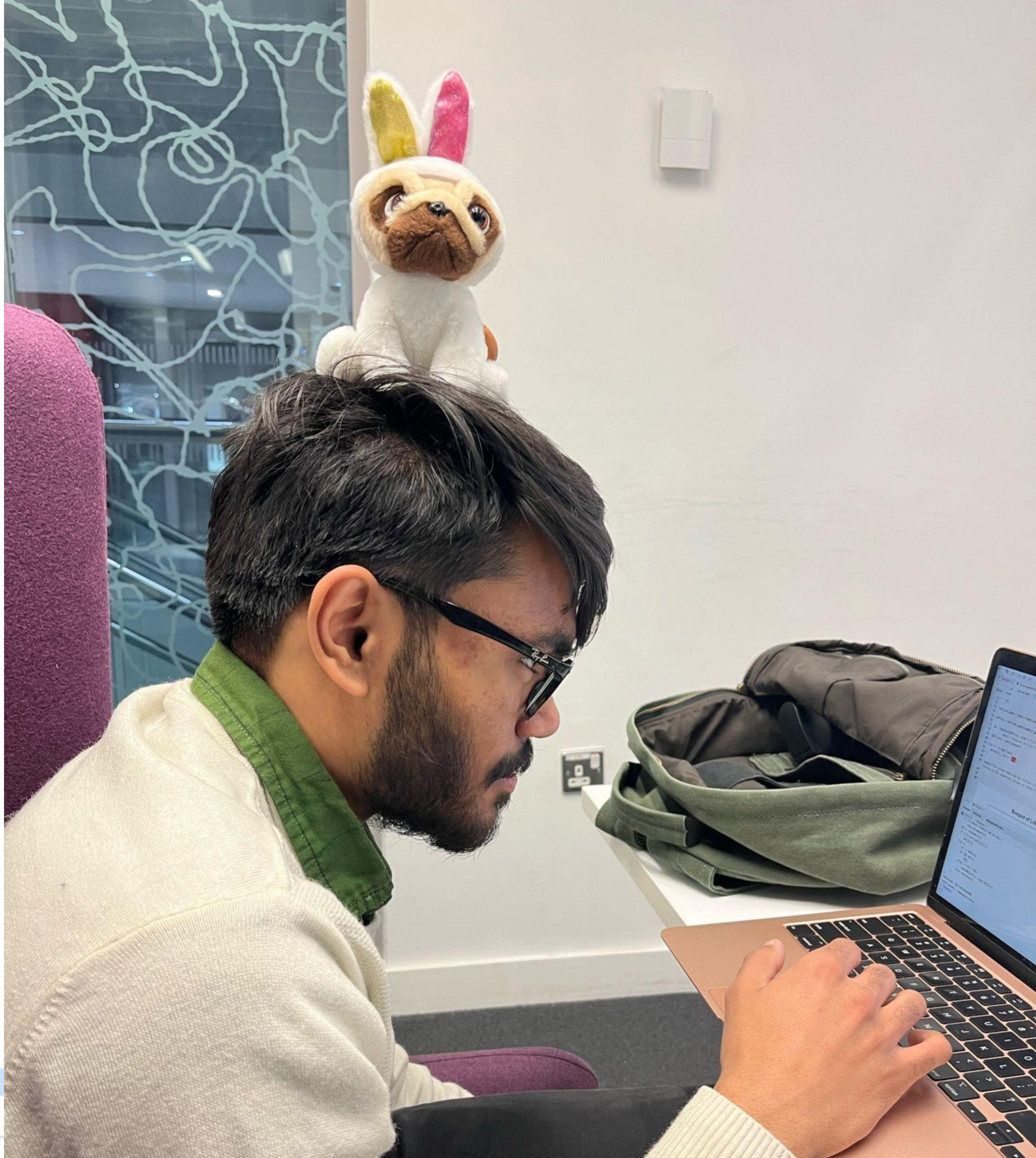
Saurav Maheshkar
Docker Bangalore August Meetup
@MaheshkarSaurav

👋, Hello my name is Saurav Maheshkar

📍 Manchester, United Kingdom

- Research Machine Learning Engineer at Re:course AI
- Interested in Geometric and Representation Learning
- Kaggle 4x Master
- Open Source and Coffee

Experts



Source Code and Workflows

https://github.com/SauravMaheshkar/python-template

Outline

- Why care about Code Quality in ML
- CI/CD in ML: Why and How ?
- Example Github Actions Workflows
- Containerization in ML: Why and How ?
- Example Containerfile and Github Actions Workflows

Code Quality in ML 💅🎨🧐🐳

- Code Quality ensures reproducibility and interpretability
- Reduces deployment time
- Increases Readability and facilitates debugging
- Availability of libraries: pytest, black, ruff, mypy, pre-commit

Pre-commit

```yaml
repos:
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v4.4.0
    hooks:
      - id: end-of-file-fixer
      - id: trailing-whitespace
      - id: check-yaml
      - id: check-toml
      - id: check-json
      - id: check-merge-conflict
      - id: requirements-txt-fixer
      - id: detect-private-key
  - repo: https://github.com/psf/black
    rev: 23.3.0
    hooks:
      - id: black
  - repo: https://github.com/pre-commit/mirrors-mypy
    rev: v1.4.1
    hooks:
      - id: mypy
  - repo: https://github.com/astral-sh/ruff-pre-commit
    rev: v0.0.277
    hooks:
      - id: ruff
```

CI/CD in ML 🔁

Why ?

- Makes managing model registry easier
- Makes deployment easier
- Enables for better artifact management
- Facilitates debugging

```yaml
name: "Build and Tests"

on:
  push:
    branches: [main]
    paths:
      - "**.py"
      - ".devcontainer/requirements.txt"
      - ".github/workflows/python.yml"
  pull_request:
    branches: [main]
    paths:
      - "**.py"
      - ".devcontainer/requirements.txt"
      - ".github/workflows/python.yml"
  release:
    types: [created]
  schedule:
    - cron: "0 0 * * 0"
```

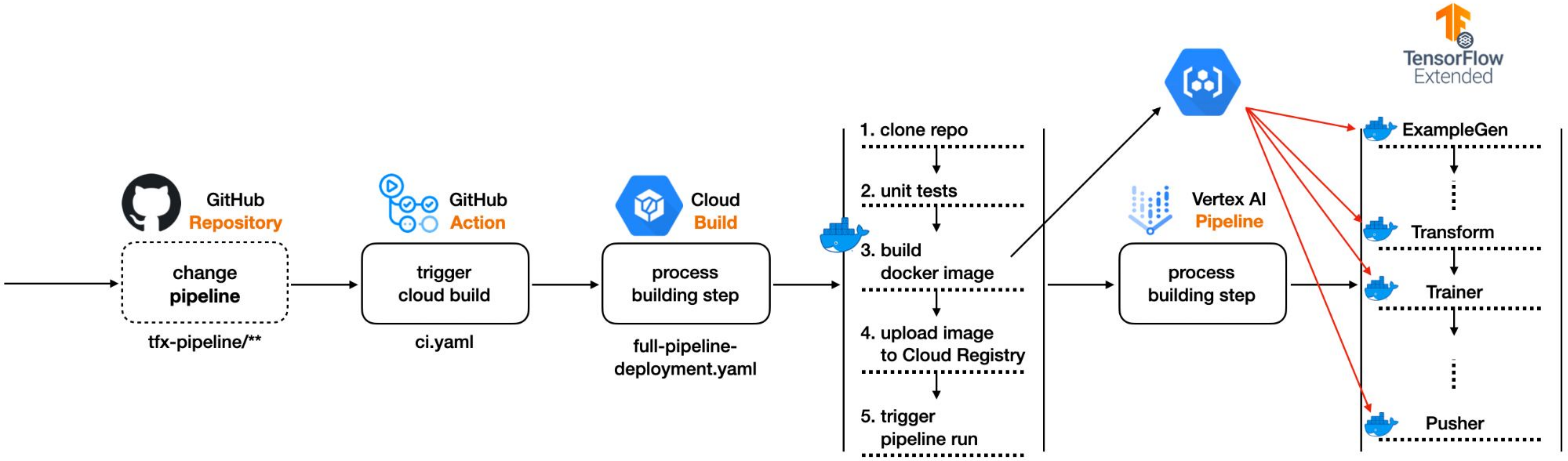
```yaml
jobs:
  build:
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        python-version: ["3.8", "3.9", "3.10", "3.11"]
        os: [ubuntu-latest, windows-latest, macos-latest]
    steps:
      - uses: actions/checkout@v3
      - name: Setup Python ${{ matrix.python-version }}
        uses: actions/setup-python@v4
        with:
          python-version: ${{ matrix.python-version }}
          cache: "pip"
          cache-dependency-path: ".devcontainer/requirements.txt"
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip wheel setuptools
          python -m pip install -r .devcontainer/requirements.txt
      - name: Ruff
        run: |
          ruff check src
      - name: Test with PyTest
        run: |
          pytest -v .
```

```
# Clone the repository.
- name: 'gcr.io/cloud-builders/git'
  args: ['clone', '--single-branch', '--branch',
         '$_BRANCH', '$_REPO_URL',
         '--depth', '1',
         '--verbose']
  id: 'Clone Repository'

# Run datasource_utils unit tests.
- name: '$_CICD_IMAGE_URI'
  entrypoint: 'pytest'
  args: ['src/tests/datasource_utils_tests.py', '-s']
  dir: 'mlops-with-vertex-ai'
  env:
  - 'PROJECT=$_PROJECT'
  - 'BQ_LOCATION=$_BQ_LOCATION'
  - 'BQ_DATASET_NAME=$_BQ_DATASET_NAME'
  - 'BQ_TABLE_NAME=$_BQ_TABLE_NAME'
  id: 'Unit Test Datasource Utils'
  waitFor: ['Clone Repository']
```

```
# Run model unit tests.
- name: '$_CICD_IMAGE_URI'
  entrypoint: 'pytest'
  args: ['src/tests/model_tests.py', '-s']
  dir: 'mlops-with-vertex-ai'
  id: 'Unit Test Model'
  waitFor: ['Clone Repository']
  timeout: 1800s

# Test e2e pipeline using local runner.
- name: '$_CICD_IMAGE_URI'
  entrypoint: 'pytest'
  args: ['src/tests/pipeline_deployment_tests.py::test_e2e_pipeline', '-s']
  dir: 'mlops-with-vertex-ai'
  env:
  - 'PROJECT=$_PROJECT'
  - 'REGION=$_REGION'
  - 'MODEL_DISPLAY_NAME=$_MODEL_DISPLAY_NAME'
  - 'DATASET_DISPLAY_NAME=$_DATASET_DISPLAY_NAME'
  - 'GCS_LOCATION=$_TEST_GCS_LOCATION'
  - 'TRAIN_LIMIT=$_CI_TRAIN_LIMIT'
  - 'TEST_LIMIT=$_CI_TEST_LIMIT'
  - 'UPLOAD_MODEL=$_CI_UPLOAD_MODEL'
  - 'ACCURACY_THRESHOLD=$_CI_ACCURACY_THRESHOLD'
  id: 'Local Test E2E Pipeline'
  waitFor: ['Unit Test Datasource Utils', 'Unit Test Model']
  timeout: 1800s
```

```yaml
# Build the image that encapsulates the pipeline.
- name: 'gcr.io/cloud-builders/docker'
  args: ['build', '-t', '$_TFX_IMAGE_URI', '.']
  dir: 'mlops-with-vertex-ai'
  id: 'Build TFX Image'
  waitFor: ['Local Test E2E Pipeline']

# Compile the pipeline.
- name: '$_CICD_IMAGE_URI'
  entrypoint: 'python'
  args: ['build/utils.py',
          '--mode', 'compile-pipeline',
          '--pipeline-name', '$_PIPELINE_NAME'
          ]
  dir: 'mlops-with-vertex-ai'
  env:
  - 'PROJECT=$_PROJECT'
  - 'REGION=$_REGION'
  - 'MODEL_DISPLAY_NAME=$_MODEL_DISPLAY_NAME'
  - 'DATASET_DISPLAY_NAME=$_DATASET_DISPLAY_NAME'
  - 'GCS_LOCATION=$_GCS_LOCATION'
  - 'TFX_IMAGE_URI=$_TFX_IMAGE_URI'
  - 'BEAM_RUNNER=$_BEAM_RUNNER'
  - 'TRAINING_RUNNER=$_TRAINING_RUNNER'
  id: 'Compile Pipeline'
  waitFor: ['Local Test E2E Pipeline']
```

```
# Upload compiled pipeline to GCS.
- name: 'gcr.io/cloud-builders/gsutil'
  args: ['cp', '$_PIPELINE_NAME.json', '$_PIPELINES_STORE']
  dir: 'mlops-with-vertex-ai'
  id:  'Upload Pipeline to GCS'
  waitFor: ['Compile Pipeline']


# Push TFX Image to Container Registy.
images: ['$_TFX_IMAGE_URI']
```



[workflow 1: build the whole pipeline]

Source: deep-diver/Model-Training-as-a-CI-CD-System

Containerization in ML 🐳

Why ?

- Common shared container for development
- Platform agnostic inference
- Easier sharing of models
- Scalable

```
# Use an Ubuntu Base Image
FROM ubuntu:22.04 AS builder

# Helpers
ARG DEBIAN_FRONTEND=noninteractive
ENV PYTHONUNBUFFERED=1

# Essential Installs
RUN apt-get update && apt-get install -y --no-install-recommends \
  build-essential \
  gcc \
  gfortran \
  libopenblas-dev \
  python3 \
  python3-pip \
  python3-dev \
  python3-venv \
  && apt-get clean && rm -rf /var/lib/apt/lists/*

COPY .devcontainer/requirements.txt .
RUN python3 -m venv /opt/venv
ENV PATH="/opt/venv/bin:$PATH"
RUN pip3 install --no-cache-dir --upgrade pip setuptools wheel
RUN pip3 install --no-cache-dir -r requirements.txt
```

```
# Runner Image
FROM ubuntu:22.04 AS runner
RUN apt update && apt install -y --no-install-recommends \
  python3 \
  python3-pip \
  python3-dev \
  python3-venv \
  && apt-get clean && rm -rf /var/lib/apt/lists/*

COPY --from=builder /opt/venv /opt/venv
ENV PATH="/opt/venv/bin:$PATH"

RUN useradd --create-home user
WORKDIR /home/user
USER user

ENTRYPOINT ["/bin/bash"]
```

Using pre-built images

```
$ docker pull pytorch/pytorch:latest
$ docker pull pytorch/pytorch:1.9.1-cuda11.1-cudnn8-runtime
$ docker pull pytorch/pytorch:1.9.1-cuda11.1-cudnn8-devel

$ docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

```yaml
name: Containers CI

on:
  workflow_run:
    workflows: [Build and Tests]
    types:
      - completed
  release:
    types: [created]

jobs:
  build_cache_buildx:
    runs-on: ubuntu-latest
    steps:
      - name: Cleanup disk
        run: |
          sudo ls -l /usr/local/lib/
          sudo ls -l /usr/share/
          sudo du -sh /usr/local/lib/
          sudo du -sh /usr/share/
          sudo rm -rf /usr/local/lib/android
          sudo rm -rf /usr/share/dotnet
          sudo du -sh /usr/local/lib/
          sudo du -sh /usr/share/
```

```yaml
- name: Checkout
  uses: actions/checkout@v3

- name: Set up Docker Buildx
  uses: docker/setup-buildx-action@v2

- uses: docker/build-push-action@v4
  with:
    context: ./
    file: .devcontainer/Containerfile
    push: false

- name: Buildah Action
  uses: redhat-actions/buildah-build@v2
  with:
    image: python-dev
    tags: latest ${{ github.sha }}
    containerfiles: |
      .devcontainer/Containerfile
```

Resources and Examples

- **deep-diver/ml-deployment-k8s-tfserving:** This project shows how to serve an TF based image classification model as a web service with TFServing, Docker, and Kubernetes(GKE)
- **deep-diver/Model-Training-as-a-CI-CD-System:** Demonstration of Model Training as a CI/CD System in Vertex AI
- **deep-diver/mlops-hf-tf-vision-models:** MLOps for Vision Models (Tensorflow) from 🤗 Transformers with Tensorflow Extended (TFX)

Questions ?

Experts

Thank You!