**.tekton/ci-pipeline.yml**

apiVersion: tekton.dev/v1

kind: Pipeline

metadata:

name: py-train-ci

spec:

params:

- name: repo\_url

type: string

- name: revision

type: string

default: feature/ci-sandbox

workspaces:

- name: source

tasks:

# 1) Clone the repo at the requested revision onto the PVC

- name: clone

params:

- name: repo\_url

value: $(params.repo\_url)

- name: revision

value: $(params.revision)

workspaces:

- name: output

workspace: source

taskSpec:

params:

- name: repo\_url

- name: revision

workspaces:

- name: output

steps:

- name: git-clone

image: alpine/git:2.44

workingDir: $(workspaces.output.path)

script: |

#!/bin/sh

set -eu

git clone --depth 1 --branch "$(params.revision)" "$(params.repo\_url)" .

git status

# 2) Install requirements only when requirements.txt changed

- name: pip-install

runAfter: [clone]

workspaces:

- name: src

workspace: source

taskSpec:

workspaces:

- name: src

steps:

- name: install

# Debian image with python & coreutils (sha256sum) available

image: python:3.11-slim

workingDir: $(workspaces.src.path)

env:

# Silence /.docker permission warnings

- name: HOME

value: /tekton/home

- name: PIP\_DISABLE\_PIP\_VERSION\_CHECK

value: "1"

script: |

#!/bin/sh

set -eu

python -V

pip install --upgrade pip wheel

# If there's no requirements file, we're done.

if [ ! -f requirements.txt ]; then

echo "No requirements.txt at repo root—skipping."

exit 0

fi

# Compute checksum of requirements and compare with last run

mkdir -p .ci

REQ\_HASH\_NEW="$(sha256sum requirements.txt | awk '{print $1}')"

REQ\_HASH\_OLD=""

if [ -f .ci/requirements.sha256 ]; then

REQ\_HASH\_OLD="$(cat .ci/requirements.sha256 || true)"

fi

if [ "$REQ\_HASH\_NEW" = "$REQ\_HASH\_OLD" ]; then

echo "Dependencies up-to-date (hash match). Skipping pip install."

else

echo "Dependencies changed or first run. Installing…"

pip install -r requirements.txt

echo "$REQ\_HASH\_NEW" > .ci/requirements.sha256

fi

# 3) Train model with your Spark code (local[\*] session)

- name: train

runAfter: [pip-install]

workspaces:

- name: src

workspace: source

taskSpec:

workspaces:

- name: src

steps:

- name: run-training

image: python:3.11-slim

workingDir: $(workspaces.src.path)

env:

- name: HOME

value: /tekton/home

- name: DOCKER\_CONFIG

value: /tekton/home/.docker

# Critical: your code reads WORKSPACE to resolve paths

- name: WORKSPACE

value: $(workspaces.src.path)

script: |

#!/bin/sh

set -eu

echo "WORKSPACE=${WORKSPACE}"

echo "Repo tree:"

find . -maxdepth 2 -type f | sort | sed 's/^/ - /'

# Optional: show data file presence (adjust as needed)

test -f spark/data/Test\_12.csv && echo "Found spark/data/Test\_12.csv"

# Kick off the training. Your model\_training.py uses WORKSPACE to find config/data.

# It also creates artifact/ and logs under the repo dir (on the PVC).

python spark/src/model\_training.py

echo "Training finished."

**.tekton/ci-run.yaml**

apiVersion: tekton.dev/v1

kind: PipelineRun

metadata:

generateName: py-train-ci-run-

spec:

pipelineRef:

name: py-train-ci

# Give every step a writable HOME to avoid '/.docker' noise/errors

taskRunTemplate:

podTemplate:

env:

- name: HOME

value: /tekton/home

- name: DOCKER\_CONFIG

value: /tekton/home/.docker

params:

- name: repo\_url

value: https://github.com/CitiInternal/178144.jedi.jedi-ml # <- your repo

- name: revision

value: feature/ci-sandbox # <- branch to run

workspaces:

- name: source

volumeClaimTemplate:

spec:

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: 2Gi