**pipeline.yaml**

apiVersion: tekton.dev/v1

kind: Pipeline

metadata:

name: train-spark-model

spec:

# One shared workspace where the repo is checked out and where we keep artifacts

workspaces:

- name: shared

# Handy knobs

params:

- name: train\_file

description: Path to the training CSV relative to repo root

default: spark/data/training.csv

- name: output\_dir

description: Where to collect the trained models (relative to repo root)

default: models\_out

tasks:

# 1) Install Python dependencies (idempotent – will no-op if nothing to install)

- name: install-requirements

taskSpec:

workspaces:

- name: src

steps:

- name: pip-install

image: python:3.9

workingDir: $(workspaces.src.path)

env:

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

value: "1"

script: |

set -euxo pipefail

python --version

pip install --upgrade pip

if [ -f requirements.txt ]; then

# If already satisfied, pip just skips – safe to run every time

pip install -r requirements.txt

else

echo "No requirements.txt found"

fi

workspaces:

- name: src

workspace: shared

# 2) Run your training script (Spark in local[\*] mode)

- name: train-model

runAfter: [install-requirements]

taskSpec:

params:

- name: train\_file

- name: output\_dir

workspaces:

- name: src

steps:

- name: run-training

image: python:3.9

workingDir: $(workspaces.src.path)

env:

# Spark uses this; your code builds the SparkSession internally

- name: PYSPARK\_PYTHON

value: python3

- name: MODEL\_DIR

value: $(workspaces.src.path)/$(params.output\_dir)

script: |

set -euxo pipefail

# Make sure an output directory exists (PVC-backed)

mkdir -p "$MODEL\_DIR"

# OPTIONAL: force Spark master to local[\*] in your config if present

python - <<'PY'

import os, json

cfg\_path = os.path.join("spark","config","config.json")

try:

with open(cfg\_path, "r", encoding="utf-8") as f:

cfg = json.load(f)

cfg.setdefault("spark", {})["master"] = "local[\*]"

with open(cfg\_path, "w", encoding="utf-8") as f:

json.dump(cfg, f)

print("Set spark.master to local[\*] in", cfg\_path)

except Exception as e:

print("Config tweak skipped:", e)

PY

# Kick off training – your script imports SparkSession and runs training

python model\_training.py

# Collect any trained models your code wrote (best-effort copy)

# If your script already writes to MODEL\_DIR, this is a no-op.

find . -maxdepth 4 -type d \( -iname "Random\_forest" -o -iname "SVM" -o -iname "Logistic\_Regression" -o -iname "models" \) -print0 \

| xargs -0 -I {} cp -r {} "$MODEL\_DIR"/ || true

params:

- name: train\_file

value: $(params.train\_file)

- name: output\_dir

value: $(params.output\_dir)

workspaces:

- name: src

workspace: shared

**pipelinerun.yaml**

apiVersion: tekton.dev/v1

kind: PipelineRun

metadata:

generateName: train-spark-model-

spec:

pipelineRef:

name: train-spark-model

params:

# Adjust if your CSV lives elsewhere in the repo

- name: train\_file

value: spark/data/training.csv

- name: output\_dir

value: models\_out

workspaces:

- name: shared

volumeClaimTemplate:

spec:

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: 2Gi

timeouts:

pipeline: 60m