name: CI/CD Pipeline

on:

push:

branches: [ main, develop ]

pull\_request:

branches: [ main ]

release:

types: [ created ]

env:

PYTHON\_VERSION: '3.11'

NODE\_VERSION: '18'

jobs:

# ====================

# Linting and Code Quality

# ====================

lint:

name: Lint Code

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: ${{ env.PYTHON\_VERSION }}

- name: Install dependencies

run: |

python -m pip install --upgrade pip

pip install flake8 black mypy pylint

- name: Run Black formatter check

run: black --check .

- name: Run Flake8 linter

run: flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics

- name: Run MyPy type checker

run: mypy sportai\_main\_app\_file.py --ignore-missing-imports

# ====================

# Security Scanning

# ====================

security:

name: Security Scan

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- name: Run Trivy vulnerability scanner

uses: aquasecurity/trivy-action@master

with:

scan-type: 'fs'

scan-ref: '.'

format: 'sarif'

output: 'trivy-results.sarif'

- name: Upload Trivy results to GitHub Security

uses: github/codeql-action/upload-sarif@v2

with:

sarif\_file: 'trivy-results.sarif'

- name: Run Bandit security linter

run: |

pip install bandit

bandit -r . -f json -o bandit-report.json

continue-on-error: true

# ====================

# Testing

# ====================

test:

name: Run Tests

runs-on: ubuntu-latest

strategy:

matrix:

python-version: ['3.8', '3.9', '3.10', '3.11']

services:

postgres:

image: postgres:15

env:

POSTGRES\_PASSWORD: testpass

POSTGRES\_DB: sportai\_test

options: >-

--health-cmd pg\_isready

--health-interval 10s

--health-timeout 5s

--health-retries 5

ports:

- 5432:5432

redis:

image: redis:7

options: >-

--health-cmd "redis-cli ping"

--health-interval 10s

--health-timeout 5s

--health-retries 5

ports:

- 6379:6379

steps:

- uses: actions/checkout@v4

- name: Set up Python ${{ matrix.python-version }}

uses: actions/setup-python@v4

with:

python-version: ${{ matrix.python-version }}

- name: Cache pip packages

uses: actions/cache@v3

with:

path: ~/.cache/pip

key: ${{ runner.os }}-pip-${{ hashFiles('requirements.txt') }}

restore-keys: |

${{ runner.os }}-pip-

- name: Install dependencies

run: |

python -m pip install --upgrade pip

pip install -r requirements.txt

pip install pytest pytest-cov pytest-asyncio

- name: Run tests with coverage

env:

DATABASE\_URL: postgresql://postgres:testpass@localhost:5432/sportai\_test

REDIS\_HOST: localhost

REDIS\_PORT: 6379

SECRET\_KEY: test-secret-key

run: |

pytest tests/ --cov=. --cov-report=xml --cov-report=html

- name: Upload coverage to Codecov

uses: codecov/codecov-action@v3

with:

file: ./coverage.xml

flags: unittests

name: codecov-umbrella

# ====================

# Build Docker Image

# ====================

docker:

name: Build Docker Image

runs-on: ubuntu-latest

needs: [lint, security, test]

if: github.event\_name == 'push' && github.ref == 'refs/heads/main'

steps:

- uses: actions/checkout@v4

- name: Set up Docker Buildx

uses: docker/setup-buildx-action@v3

- name: Log in to Docker Hub

uses: docker/login-action@v3

with:

username: ${{ secrets.DOCKER\_USERNAME }}

password: ${{ secrets.DOCKER\_PASSWORD }}

- name: Extract metadata

id: meta

uses: docker/metadata-action@v5

with:

images: sportai/suite

tags: |

type=ref,event=branch

type=ref,event=pr

type=semver,pattern={{version}}

type=semver,pattern={{major}}.{{minor}}

type=sha

- name: Build and push Docker image

uses: docker/build-push-action@v5

with:

context: .

push: true

tags: ${{ steps.meta.outputs.tags }}

labels: ${{ steps.meta.outputs.labels }}

cache-from: type=gha

cache-to: type=gha,mode=max

# ====================

# Deploy to Staging

# ====================

deploy-staging:

name: Deploy to Staging

runs-on: ubuntu-latest

needs: [docker]

if: github.event\_name == 'push' && github.ref == 'refs/heads/main'

environment:

name: staging

url: https://staging.sportai.com

steps:

- uses: actions/checkout@v4

- name: Deploy to staging server

env:

STAGING\_HOST: ${{ secrets.STAGING\_HOST }}

STAGING\_USER: ${{ secrets.STAGING\_USER }}

STAGING\_KEY: ${{ secrets.STAGING\_KEY }}

run: |

echo "$STAGING\_KEY" > deploy\_key

chmod 600 deploy\_key

ssh -o StrictHostKeyChecking=no -i deploy\_key $STAGING\_USER@$STAGING\_HOST '

cd /opt/sportai &&

docker-compose pull &&

docker-compose up -d --remove-orphans &&

docker system prune -f

'

# ====================

# Deploy to Production

# ====================

deploy-production:

name: Deploy to Production

runs-on: ubuntu-latest

needs: [docker]

if: github.event\_name == 'release'

environment:

name: production

url: https://app.sportai.com

steps:

- uses: actions/checkout@v4

- name: Deploy to production server

env:

PROD\_HOST: ${{ secrets.PROD\_HOST }}

PROD\_USER: ${{ secrets.PROD\_USER }}

PROD\_KEY: ${{ secrets.PROD\_KEY }}

run: |

echo "$PROD\_KEY" > deploy\_key

chmod 600 deploy\_key

ssh -o StrictHostKeyChecking=no -i deploy\_key $PROD\_USER@$PROD\_HOST '

cd /opt/sportai &&

docker-compose -f docker-compose.yml -f docker-compose.prod.yml pull &&

docker-compose -f docker-compose.yml -f docker-compose.prod.yml up -d --remove-orphans &&

docker system prune -f

'

- name: Verify deployment

run: |

sleep 30

curl -f https://app.sportai.com/health || exit 1

- name: Notify deployment

uses: 8398a7/action-slack@v3

with:

status: ${{ job.status }}

text: 'Production deployment completed for version ${{ github.event.release.tag\_name }}'

webhook\_url: ${{ secrets.SLACK\_WEBHOOK }}

if: always()