MÓDULO 09: GITHUB ACTIONS AVANZADO

CI Robusto, Caching, Matrix Testing y Secrets

Guía MLOps v5.0: Senior Edition | DuqueOM | Noviembre 2025

MÓDULO 09: GitHub Actions Avanzado

De CI Básico a Pipeline Production-Ready

"Si tu CI tarda 30 minutos, nadie lo va a usar."

Duración	Teoría	Práctica
4-5 horas	20%	80%

Lo Que Lograrás en Este Módulo

1. **Diseñar** workflows de CI eficientes con caching
2. **Implementar** matrix testing para múltiples versiones
3. **Gestionar** secrets de forma segura
4. **Optimizar** tiempos de ejecución

9.1 Anatomía de un Workflow

```yaml
# .github/workflows/ci.yml
name: CI Pipeline  # Nombre del workflow

on:  # Triggers
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]
  workflow_dispatch:  # Trigger manual

env:  # Variables globales
  PYTHON_VERSION: "3.11"

jobs:
  job-name:
    runs-on: ubuntu-latest
    steps:
      - name: Step name
        uses: action/name@version
        with:
          parameter: value
```

9.2 Workflow CI Completo para MLOps

```yaml
# .github/workflows/ci.yml
name: CI Pipeline

on:
  push:
    branches: [main, develop]
    paths-ignore:
      - '**.md'
      - 'docs/**'
      - '.gitignore'
  pull_request:
    branches: [main]
```

```yaml
env:
  PYTHON_VERSION: "3.11"
  POETRY_VERSION: "1.7.0"

jobs:
  # ═══════════════════════════════════════════════════════════════
  # JOB 1: Lint y Type Check (Rápido, falla temprano)
  # ═══════════════════════════════════════════════════════════════
  lint:
    name: Lint & Type Check
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: ${{ env.PYTHON_VERSION }}

      - name: Install linters
        run: pip install ruff mypy

      - name: Run Ruff
        run: ruff check src/

      - name: Run Ruff Format Check
        run: ruff format --check src/

      - name: Run MyPy
        run: mypy src/ --ignore-missing-imports

  # ═══════════════════════════════════════════════════════════════
  # JOB 2: Tests con Coverage
  # ═══════════════════════════════════════════════════════════════
  test:
    name: Tests (Python ${{ matrix.python-version }})
    needs: lint  # Solo si lint pasa
    runs-on: ubuntu-latest

    strategy:
      fail-fast: false
      matrix:
        python-version: ["3.10", "3.11", "3.12"]

    steps:
      - uses: actions/checkout@v4

      - name: Set up Python ${{ matrix.python-version }}
        uses: actions/setup-python@v5
        with:
          python-version: ${{ matrix.python-version }}

      - name: Cache pip dependencies
        uses: actions/cache@v4
        with:
          path: ~/.cache/pip
          key: pip-${{ runner.os }}-${{ matrix.python-version }}-${{ hashFiles('requirements*.txt') }}
          restore-keys: |
            pip-${{ runner.os }}-${{ matrix.python-version }}-

      - name: Install dependencies
        run: |
          pip install --upgrade pip
          pip install -r requirements.txt
          pip install -r requirements-dev.txt
          pip install -e .

      - name: Run tests with coverage
        run: |
          pytest tests/ \
            -v \
            --cov=src/bankchurn \
            --cov-report=xml \
            --cov-report=term-missing \
            --cov-fail-under=80

      - name: Upload coverage to Codecov
        if: matrix.python-version == '3.11'
        uses: codecov/codecov-action@v3
        with:
          file: ./coverage.xml
          flags: unittests
          fail_ci_if_error: false

  # ═══════════════════════════════════════════════════════════════
  # JOB 3: Security Scan
  # ═══════════════════════════════════════════════════════════════
  security:
    name: Security Scan
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: ${{ env.PYTHON_VERSION }}

      - name: Install security tools
        run: pip install bandit safety

      - name: Run Bandit (code security)
        run: bandit -r src/ -ll --skip B101

      - name: Run Safety (dependency vulnerabilities)
        run: safety check -r requirements.txt --full-report
        continue-on-error: true  # No bloquear por vulns conocidas

  # ═══════════════════════════════════════════════════════════════
  # JOB 4: Build Docker (solo en main)
  # ═══════════════════════════════════════════════════════════════
  build:
    name: Build Docker Image
    needs: [test, security]
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'

    steps:
      - uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Login to GitHub Container Registry
        uses: docker/login-action@v3
        with:
          registry: ghcr.io
```

```
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Build and push
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          tags: |
            ghcr.io/${{ github.repository }}:${{ github.sha }}
            ghcr.io/${{ github.repository }}:latest
          cache-from: type=gha
          cache-to: type=gha,mode=max
```

9.3 Caching Avanzado

Cache de pip

```
- name: Cache pip dependencies
  uses: actions/cache@v4
  with:
    path: ~/.cache/pip
    key: pip-${{ runner.os }}-${{ hashFiles('**/requirements*.txt') }}
    restore-keys: |
      pip-${{ runner.os }}-
```

Cache de Poetry

```
- name: Cache Poetry virtualenv
  uses: actions/cache@v4
  with:
    path: |
      ~/.cache/pypoetry
      .venv
    key: poetry-${{ runner.os }}-${{ hashFiles('**/poetry.lock') }}
    restore-keys: |
      poetry-${{ runner.os }}-

- name: Install dependencies
  run: |
    poetry config virtualenvs.in-project true
    poetry install --no-interaction
```

Cache de Docker layers

```
- name: Build and push
  uses: docker/build-push-action@v5
  with:
    context: .
    push: true
    tags: ghcr.io/${{ github.repository }}:latest
    cache-from: type=gha
    cache-to: type=gha,mode=max
```

9.4 Matrix Testing

Múltiples Versiones de Python

```
strategy:
  fail-fast: false  # Continuar aunque uno falle
  matrix:
    python-version: ["3.10", "3.11", "3.12"]
    os: [ubuntu-latest, macos-latest]

runs-on: ${{ matrix.os }}
steps:
  - uses: actions/setup-python@v5
    with:
      python-version: ${{ matrix.python-version }}
```

Matrix con Exclusiones

```
strategy:
  matrix:
    python-version: ["3.10", "3.11", "3.12"]
    os: [ubuntu-latest, windows-latest]
    exclude:
      - os: windows-latest
        python-version: "3.10"  # Skip Python 3.10 en Windows
    include:
      - os: ubuntu-latest
        python-version: "3.11"
        experimental: true  # Variable custom
```

9.5 Gestión de Secrets

Configurar Secrets

```
# En GitHub: Settings → Secrets and variables → Actions
# Añadir secrets:
# - AWS_ACCESS_KEY_ID
# - AWS_SECRET_ACCESS_KEY
# - MLFLOW_TRACKING_URI
```

Usar Secrets en Workflows

```yaml
jobs:
  deploy:
    runs-on: ubuntu-latest
    env:
      AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
      AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}

    steps:
      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: us-east-1

      - name: Push to S3
        run: aws s3 cp ./models/ s3://my-bucket/models/ --recursive
```

Environments para Control de Secrets

```yaml
jobs:
  deploy-staging:
    runs-on: ubuntu-latest
    environment: staging  # Usa secrets de "staging"
    steps:
      - run: echo "Deploying to ${{ vars.DEPLOY_URL }}"

  deploy-production:
    runs-on: ubuntu-latest
    environment: production  # Usa secrets de "production", requiere approval
    needs: deploy-staging
    steps:
      - run: echo "Deploying to production"
```

9.6 Workflow para ML: Training Pipeline

```yaml
# .github/workflows/ml-training.yml
name: ML Training Pipeline

on:
  workflow_dispatch:
    inputs:
      experiment_name:
        description: 'Nombre del experimento'
        required: true
        default: 'bankchurn-experiment'
      n_estimators:
        description: 'Número de estimadores'
        required: false
        default: '100'

env:
  MLFLOW_TRACKING_URI: ${{ secrets.MLFLOW_TRACKING_URI }}

jobs:
  train:
    name: Train Model
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.11"

      - name: Cache dependencies
        uses: actions/cache@v4
        with:
          path: ~/.cache/pip
          key: pip-${{ hashFiles('requirements.txt') }}

      - name: Install dependencies
        run: |
          pip install -r requirements.txt
          pip install -e .

      - name: Pull data with DVC
        env:
          AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
          AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        run: |
          dvc pull

      - name: Train model
        run: |
          python src/bankchurn/main.py \
            --experiment-name "${{ github.event.inputs.experiment_name }}" \
            --n-estimators ${{ github.event.inputs.n_estimators }}

      - name: Upload model artifact
        uses: actions/upload-artifact@v4
        with:
          name: trained-model
          path: models/pipeline.pkl
          retention-days: 30
```

9.7 Reusable Workflows

Workflow Reutilizable

```yaml
# .github/workflows/reusable-test.yml
name: Reusable Test Workflow

on:
  workflow_call:
    inputs:
      python-version:
        required: true
        type: string
      coverage-threshold:
        required: false
        type: number
        default: 80

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - uses: actions/setup-python@v5
        with:
          python-version: ${{ inputs.python-version }}

      - run: pip install -r requirements-dev.txt

      - run: pytest --cov-fail-under=${{ inputs.coverage-threshold }}
```

Llamar Workflow Reutilizable

```yaml
# .github/workflows/ci.yml
jobs:
  test-3-11:
    uses: ./.github/workflows/reusable-test.yml
    with:
      python-version: "3.11"
      coverage-threshold: 85

  test-3-12:
    uses: ./.github/workflows/reusable-test.yml
    with:
      python-version: "3.12"
```

9.8 Ejercicio Integrador

Crea CI Pipeline Completo

1. **Lint job**: ruff + mypy
2. **Test job**: pytest con coverage
3. **Security job**: bandit + safety
4. **Build job**: Docker image (condicional)

Checklist

```
WORKFLOW BÁSICO:
[ ] Trigger en push y PR
[ ] Cache de dependencias configurado
[ ] Tests con coverage

AVANZADO:
[ ] Matrix testing (múltiples Python versions)
[ ] Security scan
[ ] Docker build condicional
[ ] Secrets configurados correctamente
```

Siguiente Paso

Con CI automatizado, es hora de crear **imágenes Docker optimizadas**.

Ir a Módulo 10: Docker Avanzado →

Módulo 09 completado. Tu CI ahora es rápido y confiable.