CI/CD & Automation — Follow-Along Guide

Week 11 · CS 203 · Software Tools and Techniques for AI

Prof. Nipun Batra · IIT Gandhinagar

Spring 2026

How to Use This Guide

- Open `cicd_followalong.sh` in your editor (left half of screen)
- Open a terminal (right half of screen)
- Copy-paste each command, one at a time
- **Type it yourself** — that's how you learn

Legend: `$` = command to type. `>>` = expected output. Blue boxes = projector slide.

Projector: Slides 2–3 — The Manual Quality Control Problem Nobody manually checks tests before merging. Humans forget. Machines don't.

Act 1: Setup — A Testable ML Project ~8 min

```
$ mkdir -p ~/cicd-demo && cd ~/cicd-demo
$ git init
$ python -m venv .venv && source .venv/bin/activate
$ pip install scikit-learn numpy pytest ruff pre-commit
$ mkdir -p src tests .github/workflows
```

Create `src/train.py` with `load_data()` and `train_model()` functions:

```
$ python src/train.py
>> Accuracy: 0.xxx
```

Projector: Slides 4–6 — Testing Pyramid + What to Test in ML Many unit tests (fast, cheap), fewer integration, few E2E. In ML: test shapes, types, pipeline completion — NOT accuracy.

Act 2: Writing Tests with pytest ~10 min

Create tests in `tests/test_data.py` and `tests/test_model.py`:

```python
def test_data_shapes():
    X_train, X_test, y_train, y_test = load_data()
    assert X_train.shape[1] == 8
    assert len(y_test) == 100

def test_predictions_valid():
    model = train_model()
    preds = model.predict(np.random.rand(10, 8))
    assert all(p in [0, 1] for p in preds)
```

```
$ pytest tests/ -v
>> tests/test_data.py::test_data_shapes PASSED
>> tests/test_data.py::test_data_types PASSED
>> tests/test_data.py::test_data_reproducibility PASSED
>> tests/test_model.py::test_model_trains PASSED
>> tests/test_model.py::test_predictions_valid PASSED
>> tests/test_model.py::test_model_reproducibility PASSED
>> tests/test_model.py::test_predict_proba_range PASSED
>> 7 passed
```

Tip

What to test in ML code: data shapes, data types, no NaN after preprocessing, model produces output, predictions in valid range, pipeline runs end-to-end. **What NOT to test in CI:** accuracy thresholds (that's experiment tracking).

Projector: Slides 8–10 — Pre-commit Hooks + Ruff Hooks run before `git commit`. If they fail, the commit is rejected. Ruff is 100x faster than flake8.

Act 3: Pre-commit Hooks ~10 min

First, see Ruff catch issues:

```
$ ruff check src/messy.py
>> src/messy.py:1:8: F401 `os` imported but unused
>> src/messy.py:3:8: F401 `json` imported but unused
$ ruff check --fix src/messy.py && ruff format src/messy.py
```

Set up pre-commit:

```yaml
# .pre-commit-config.yaml
repos:
  - repo: https://github.com/astral-sh/ruff-pre-commit
    rev: v0.4.4
    hooks:
      - id: ruff
        args: [--fix]
      - id: ruff-format
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v4.6.0
    hooks:
      - id: trailing-whitespace
      - id: end-of-file-fixer
      - id: check-yaml
      - id: check-added-large-files
```

```
$ pre-commit install
$ git add -A && git commit -m "Add bad code"
>> ruff.......Failed   (auto-fixes the file)
>> ruff-format.......Failed
$ git add -A && git commit -m "Add utility function"
>> All hooks passed
```

Warning

When a hook auto-fixes files, the commit is rejected but the fixes are applied. You need to `git add` again and commit a second time.

Projector: Slides 12–15 — GitHub Actions Workflow YAML in `.github/workflows/`. Triggers: push, pull_request. Jobs run on GitHub-hosted machines.

Act 4: GitHub Actions — CI on Every Push ~12 min

```yaml
# .github/workflows/ci.yml
name: CI
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.10" }
      - run: pip install ruff
      - run: ruff check .
  test:
    runs-on: ubuntu-latest
    needs: lint
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with: { python-version: "3.10" }
      - run: pip install -r requirements.txt && pip install pytest
      - run: pytest tests/ -v
```

Key concepts: `on` = trigger, `jobs` = parallel work, `needs` = dependencies, `steps` = commands.

Act 5: Push and Watch CI Run ~10 min

```
$ gh repo create cicd-demo --public --source=. --push
$ gh run watch
>>  lint   in 15s
>>  test   in 30s
>>  train  in 20s
```

Projector: Slide 16 — Branch Protection Require CI to pass before merging. Require PR reviews. Nobody pushes directly to main.

Act 6: Branch Protection + PR Workflow ~10 min

```
$ git checkout -b feature/add-evaluation
# ... add src/evaluate.py + tests/test_evaluate.py ...
$ pytest tests/ -v
>> 11 passed
$ git add -A && git commit -m "Add evaluation module with tests"
$ git push -u origin feature/add-evaluation
$ gh pr create --title "Add evaluation module" --body "..."
$ gh pr checks
>>  lint    test    train
$ gh pr merge --merge
```

Tip

The workflow: branch → code → test locally → push → CI runs → PR review → merge. This is how professional teams work.

Act 7: Makefile — Run CI Locally ~5 min

```
ci: lint test
    @echo "All checks passed!"
lint:
    ruff check .
test:
    pytest tests/ -v
```

```
$ make ci
>> All checks passed - safe to push!
```

Quick Reference

I want to...	Command
Run tests	`pytest tests/ -v`
Lint code	`ruff check .`
Auto-fix lint	`ruff check --fix .`
Format code	`ruff format .`
Install hooks	`pre-commit install`
Run hooks manually	`pre-commit run --all-files`
Create GitHub repo	`gh repo create name --public --source=. --push`
Watch CI run	`gh run watch`
Check PR status	`gh pr checks`

I want to…	Command
Run CI locally	`make ci`

Exam-relevant concepts:

- Testing pyramid: unit (many, fast) > integration > E2E (few, slow)
- CI = automated checks on every push; CD = automated build/deploy after CI
- What to test in ML: data shapes, pipeline completion, prediction validity
- What NOT to test in CI: accuracy thresholds (use experiment tracking)
- Pre-commit hooks catch issues at the earliest point — before bad code is committed