Unit Tests + Continuous Integration

Machine Learning in Production / AI Engineering - Recitation 7

Overview

- Unit Tests
- Continuous Integration (CI)
- CI Pipeline
- Demo
- CI Pipeline Qualities
- References

Unit Tests

- Code Testing Capability to prevent software bugs or unexpected output
- Unit Test frameworks: PyTest (Recommended), Unittest, and more
- Useful for CI/CD pipelines
- Good Practices:
  - Give Descriptive Test Names
  - Each test should be independent
  - Tests should have good code coverage (test all functions)
  - Test each function automatically on every code push

Unit Tests - Limitations

- It alone does not cover integration problems
- Failures can go undetected if you don’t explicitly write **ALL** tests
- Can get hard to retroactively apply. Adopt Unit Tests from start.

Continuous Integration (CI)

- Sequence checks a system has to go through before it can be deployed
- Usually followed by continuous deployment (CD) stages
- Flow
  - Code commit triggers a new pipeline run
  - Pipeline executes
  - If the CI pipeline passes, CD pipeline starts
- Goals:
  - Reduce the time taken from feature push to deployment
  - Another goal is to automate tests activities

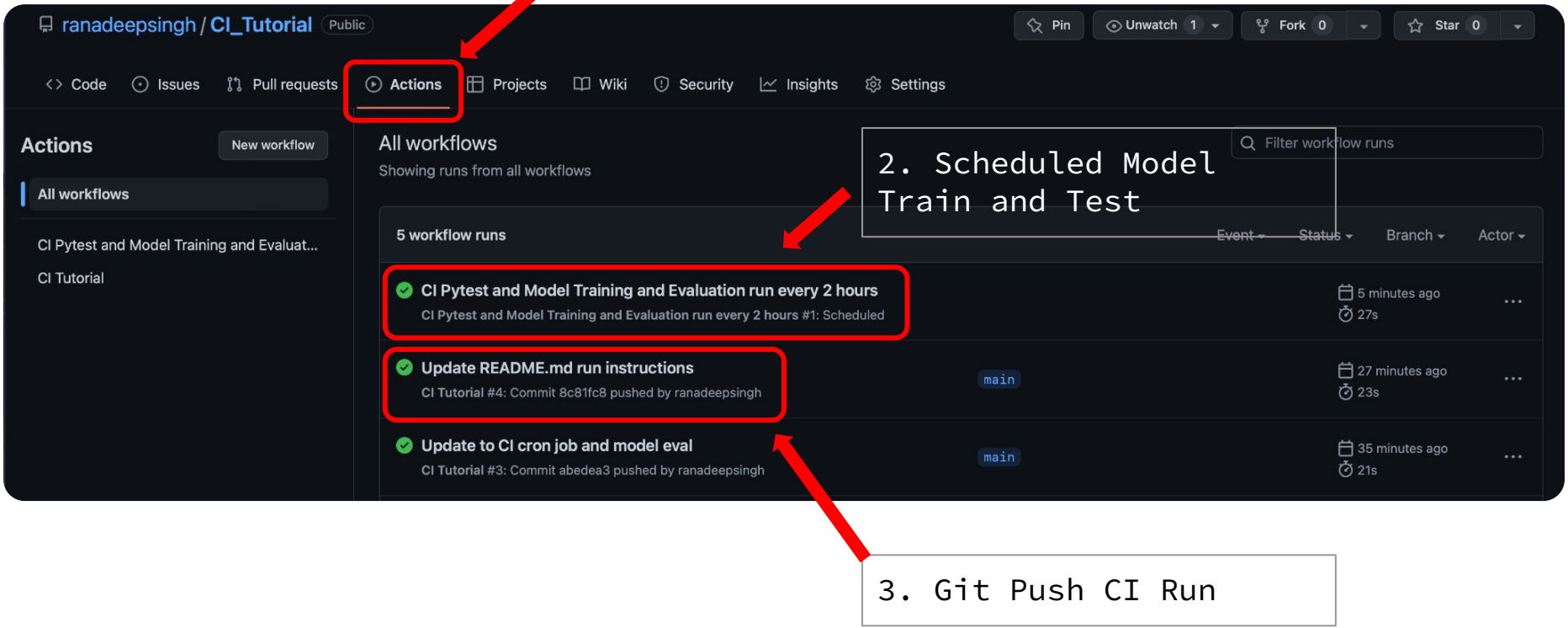
CI Pipeline

- Defined set of stages which run in an automated fashion once triggered
- Pipeline stages:
  - Push code → Set up environment → Build code → Static checks → Unit tests → Integration tests → Packaging the software → …
- For machine learning, you may have more stages such as:
  - Data quality check, offline model evaluation, data collection, data cleaning/preprocessing, model serialization, telemetry data collection, etc.
- CI/CD tools: Github Actions (Recommended), Jenkins, TravisCI, CircleCI, and others.

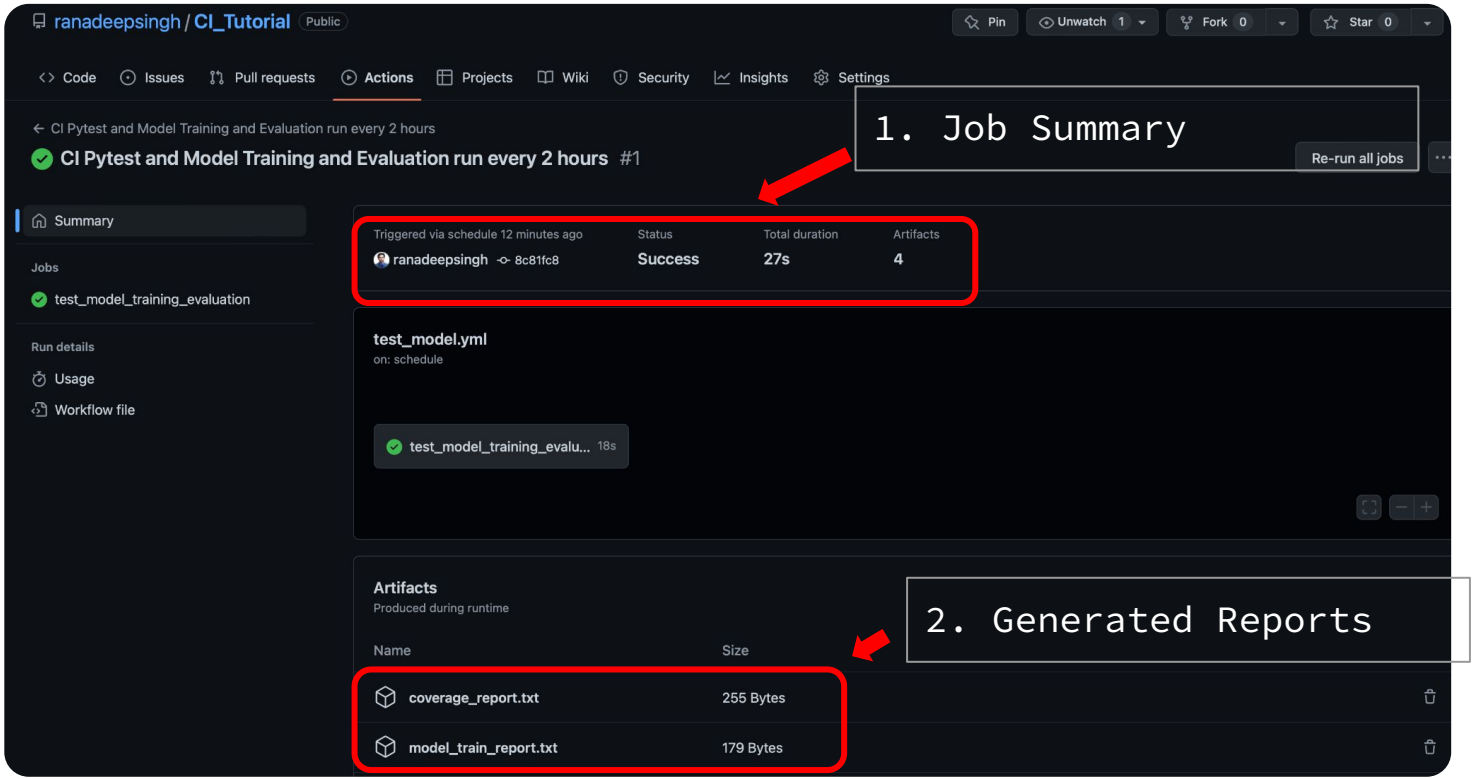
Demo

- Sample Code: https://github.com/ranadeepsingh/CI_Tutorial
- Contents
  - Code Walkthrough
  - PyTest Implementation
  - Model Training and Evaluation Tests
  - CI with GitHub Actions in action (pun intended)

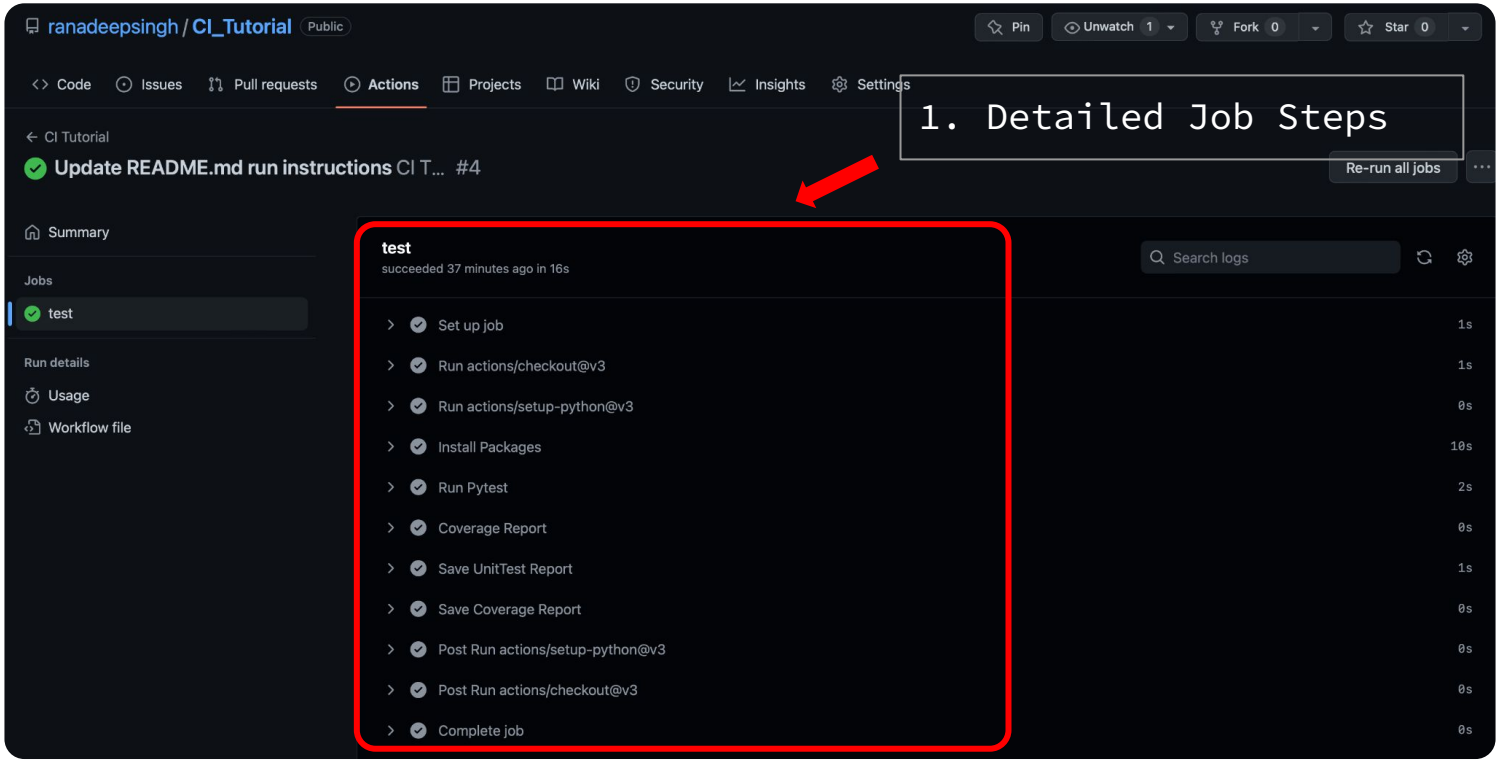
Demo


1. GitHub Actions Tab
ranadeepsingh / CI_Tutorial
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Actions
New workflow
All workflows
CI Pytest and Model Training and Evaluat...
CI Tutorial
All workflows
Showing runs from all workflows
Filter workflow runs
2. Scheduled Model Train and Test
5 workflow runs
Event
Status
Branch
Actor
CI Pytest and Model Training and Evaluation run every 2 hours
CI Pytest and Model Training and Evaluation run every 2 hours #1: Scheduled
5 minutes ago
27s
Update README.md run instructions
CI Tutorial #4: Commit 8c81fc8 pushed by ranadeepsingh
main
27 minutes ago
23s
Update to CI cron job and model eval
CI Tutorial #3: Commit abedea3 pushed by ranadeepsingh
main
35 minutes ago
21s
3. Git Push CI Run

Demo


ranadeepsingh / CI_Tutorial
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
CI Pytest and Model Training and Evaluation run every 2 hours
CI Pytest and Model Training and Evaluation run every 2 hours #1
Re-run all jobs
1. Job Summary
Summary
Jobs
test_model_training_evaluation
Run details
Usage
Workflow file
Triggered via schedule 12 minutes ago
ranadeepsingh 8c81fc8
Status
Success
Total duration
27s
Artifacts
4
test_model.yml
on: schedule
test_model_training_evalu... 18s
Artifacts
Produced during runtime
Name
Size
coverage_report.txt
255 Bytes
model_train_report.txt
179 Bytes
2. Generated Reports

Demo


ranadeepsingh / CI_Tutorial
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
1. Detailed Job Steps
CI Tutorial
Update README.md run instructions CI T... #4
Re-run all jobs
Summary
Jobs
test
Run details
Usage
Workflow file
test
succeeded 37 minutes ago in 16s
Search logs
Set up job 1s
Run actions/checkout@v3 1s
Run actions/setup-python@v3 0s
Install Packages 10s
Run Pytest 2s
Coverage Report 0s
Save UnitTest Report 1s
Save Coverage Report 0s
Post Run actions/setup-python@v3 0s
Post Run actions/checkout@v3 0s
Complete job 0s

CI Pipeline Qualities

- Repeatable - Consistent results across runs; consecutive runs are independent
- Fault-tolerant - Fail gracefully if any stage fails, ie. system remains operational
- Correct - Performs what is expected of it given some inputs
- Robust - Able to handle noise in any inputs the pipeline expects
- Testable - Stages of the pipeline should be independently testable
- Traceable - Possible to trace any error to its source quickly
- Performant - Possible to move through the pipeline quickly

References

- Demo Code: https://github.com/ranadeepsingh/CI_Tutorial
- PyTest: https://docs.pytest.org/en/7.1.x/
- GitHub Actions: https://docs.github.com/en/actions
- Circle CI: https://circleci.com
- Circle CI Tutorial: https://www.youtube.com/watch?v=4dp4JFpOpX0