

warning "Backend update (October 2026)" GitHub Models inference was retired on July 30, 2026. Any references below are retained only for historical context while this lab is migrated. Use Lab 013 ? Free LLM backends to configure Foundry Local or Ollama before continuing.

Lab 037: CI/CD for AI Agents with GitHub Actions

Level: L300 Path: Pro Code Time: ~45 min ? Cost: GitHub Free ? GitHub Actions free tier (2000 min/month)

What You'll Learn

Why AI agents need CI/CD pipelines (they're different from regular software) Automated prompt regression tests ? detect when a prompt change breaks behavior LLM-as-judge evaluation in CI ? use a model to grade outputs Safe deployment patterns: shadow mode, canary releases A complete GitHub Actions workflow for an AI agent

Introduction

Shipping agent updates is risky. A small prompt change can silently break the agent's behavior. Unlike traditional software, there's no compiler or type checker ? the "bug" is a subtle shift in reasoning quality.

CI/CD for AI agents needs:

1. Regression tests for expected behaviors 2. Quality gates (automated evals) before merging 3. Safe deployment strategies that limit blast radius

Prerequisites

GitHub account (free) GITHUB_TOKEN (Personal Access Token or Actions secret) Python project with a simple agent (we'll create one)

? Supporting Files

note "Download these files before starting the lab" Save all files to a lab-037/ folder in your working directory.

File | Description | Download | ai-agent-ci.yml | CI/CD workflow template | ? Download |

Lab Exercise

Step 1: Create a testable agent module

[diagram or code sample]

agent.py ? a simple customer service agent:

[diagram or code sample]

Step 2: Behavioral tests (fast, deterministic)

These tests check binary behaviors ? things that should always be true.

tests/test_behaviors.py:

[diagram or code sample]

Step 3: Golden test cases (JSON)

evals/test_cases.json:

[diagram or code sample]

Step 4: LLM-as-judge evaluation

tests/test_evals.py:

[diagram or code sample]

Step 5: GitHub Actions workflow

.github/workflows/agent-ci.yml:

[diagram or code sample]

CI Strategy Summary

Test type | Runs on | Speed | Cost | Behavioral (pytest) | Every PR | Fast (~10s) | No LLM calls | Golden cases | Every PR | Medium (~30s) | Minimal (structured output) | LLM-as-judge | Main branch only | Slow (~2min) | ~10 LLM calls | Deploy | Main, tests pass

| ? | ? |

tip "Prompt versioning" Store your prompts in version-controlled files (prompts/system.txt), not hardcoded in Python. This makes diffs meaningful and enables rollback.

? Knowledge Check

question "Q1 (Run the Lab): Open lab-037/ai-agent-ci.yml. How many jobs are defined in the workflow?"

Open the file and count the number of job-name: entries under the jobs: key.

??? success "? Reveal Answer" 7 jobs

The workflow defines: unit-tests, integration-tests, security-scan, agent-evaluation, docker-build, deploy-staging, and deploy-production. Each job has a distinct responsibility in the agent deployment pipeline.

question "Q2 (Run the Lab): Which job in ? ai-agent-ci.yml does NOT run on pull requests ? only on pushes to the main branch?"

Look at the if: condition on each job, or the workflow-level trigger vs per-job conditions.

??? success "? Reveal Answer" integration-tests

Integration tests call real external APIs (GitHub Models, etc.) and could fail due to rate limits or cost if run on every PR. The integration-tests job has if: github.event_name == 'push' && github.ref == 'refs/heads/main' ? it only runs on pushes to main, not on every pull request. Unit tests run on all events.

question "Q3 (Multiple Choice): The deploy-production job in the workflow requires manual approval. What GitHub Actions feature enables this?"

- A) A manual-approval: true flag in the job definition - B) An environment: block referencing a protected environment with required reviewers configured - C) A wait-for-approval step using the GitHub API - D) Setting runs-on: manual instead of runs-on: ubuntu-latest

??? success "? Reveal Answer" Correct: B ? A protected GitHub Environment with required reviewers

The deploy-production job specifies environment: production. In GitHub repository settings, you configure the production environment to require 1+ specific reviewers before the job runs. When the workflow reaches this job, GitHub sends a notification to the reviewers, pauses the workflow, and waits. Only after approval does the deployment proceed. This is the standard GitHub Actions pattern for human-in-the-loop production deployments.

Next Steps

Evaluate at scale with Azure AI: ? Lab 035 ? Agent Evaluation with Azure AI Eval SDK Deploy your agent to Azure Container Apps: ? Lab 028 ? Deploy MCP to Azure Container Apps Supporting kit

The site build generates stable PDF and ZIP downloads. The ZIP contains the lab guide and every supporting file declared for this lab.

Download generated PDF Download generated ZIP

Supporting file: ai-agent-ci.yml

AI Agent CI/CD Pipeline Lab 037: CI/CD for AI Agents (GitHub Actions) This workflow demonstrates a complete CI/CD pipeline for AI agents: 1. Unit tests (mock LLM responses) 2. Integration tests (real GitHub Models API) 3. Agent evaluation (groundedness, coherence, safety) 4. Security scan (prompt injection patterns) 5. Deploy to staging then production

Required secrets (Settings ? Secrets ? Actions): GITHUB_TOKEN - auto-provided by GitHub Actions AZURE_CREDENTIALS - Service principal JSON for Azure deployment AZURE_OPENAI_KEY -

For production evaluation (L300+)

name: AI Agent CI/CD

on: push: branches: [main] paths: - 'src/' - 'tests/' - 'eval/' pull_request: branches:

[main] paths: - 'src/' - 'tests/'

env: PYTHON_VERSION: "3.11" AGENT_SERVICE_NAME: "outdoorgear-agent" STAGING_URL:

"https://outdoorgear-agent-staging.azurecontainerapps.io"

===== JOBS

===== jobs:

----- # JOB 1: Lint

+ Unit Tests (no API calls, always free) #

----- unit-tests:

name: "? Unit Tests" runs-on: ubuntu-latest

steps: - uses: actions/checkout@v4

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

python-version: \${{ env.PYTHON_VERSION }} cache: "pip"

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

pytest-cov pytest-asyncio ruff mypy

- name: Lint with ruff run: ruff check src/ tests/

- name: Type check with mypy run: mypy src/ --ignore-missing-imports

- name: Run unit tests (mocked LLM) run: | pytest tests/unit/ \ -v \ --cov=src \

--cov-report=xml \ --cov-report=term-missing \ -m "not integration"

- name: Upload coverage report uses: codecov/codecov-action@v4 with: file: coverage.xml

fail_ci_if_error: false

----- # JOB 2:

Integration Tests (real API calls ? uses GitHub Models free tier) #

integration-tests: name: "? Integration Tests" runs-on: ubuntu-latest needs: unit-tests #

Only run on main branch pushes, not on every PR to save API quota if: github.event_name ==

'push' && github.ref == 'refs/heads/main'

steps: - uses: actions/checkout@v4

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

python-version: \${{ env.PYTHON_VERSION }} cache: "pip"

- name: Install dependencies run: pip install -r requirements.txt pytest pytest-asyncio

- name: Run integration tests env: GITHUB_TOKEN: \${{ secrets.GITHUB_TOKEN }} #

Auto-provided ? no extra secret needed! LLM_BASE_URL:

"https://models.inference.ai.azure.com" LLM_MODEL: "gpt-4o-mini" run: | pytest

tests/integration/ \ -v \ -m "integration" \ --timeout=60

----- # JOB 3:

Security Scan (prompt injection patterns) #

----- security-scan:

name: "? Security Scan" runs-on: ubuntu-latest needs: unit-tests

steps: - uses: actions/checkout@v4

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

python-version: \${{ env.PYTHON_VERSION }} cache: "pip"

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

- name: Bandit security scan (Python code) run: bandit -r src/ -f json -o

bandit-report.json || true

- name: Check for known vulnerable dependencies run: safety check --json >

```

safety-report.json || true
- name: Scan system prompt for injection vulnerabilities run: | python
tests/security/scan_prompts.py \ --prompt-files src/prompts/ \ --output
security-prompt-scan.json
- name: Upload security reports uses: actions/upload-artifact@v4 if: always() with: name:
security-reports path: | bandit-report.json safety-report.json security-prompt-scan.json
# ----- # JOB 4:
Agent Evaluation (requires Azure ? skip if no credentials) #
-----
agent-evaluation: name: "? Agent Evaluation" runs-on: ubuntu-latest needs:
integration-tests if: | github.event_name == 'push' && github.ref == 'refs/heads/main' &&
secrets.AZURE_OPENAI_KEY != "
steps: - uses: actions/checkout@v4
- name: Set up Python ${ env.PYTHON_VERSION } uses: actions/setup-python@v5 with:
python-version: ${ env.PYTHON_VERSION } cache: "pip"
- name: Install evaluation dependencies run: pip install azure-ai-evaluation promptflow
- name: Run agent evaluation suite env: AZURE_OPENAI_KEY: ${ secrets.AZURE_OPENAI_KEY }
AZURE_OPENAI_ENDPOINT: ${ vars.AZURE_OPENAI_ENDPOINT } EVAL_MODEL: "gpt-4o" run: | python
eval/run_evaluation.py \ --dataset eval/eval_dataset.jsonl \ --output-dir eval-results/ \
--metrics groundedness,coherence,relevance,safety \ --fail-threshold 0.8 # Fail if any
metric drops below 80%
- name: Parse evaluation results id: eval_results run: | python eval/parse_results.py
eval-results/summary.json >> $GITHUB_STEP_SUMMARY
- name: Upload evaluation results uses: actions/upload-artifact@v4 if: always() with: name:
eval-results path: eval-results/
- name: Comment evaluation summary on PR if: github.event_name == 'pull_request' uses:
actions/github-script@v7 with: script: | const fs = require('fs'); const summary =
fs.readFileSync('eval-results/summary.json', 'utf8'); const data = JSON.parse(summary);
const body = ## ? Agent Evaluation Results\n\n + | Metric | Score | Threshold | Status |\n
+ |-----|-----|-----|-----|\n + Object.entries(data.metrics).map(([k, v]) =>
| ${k} | ${v.score.toFixed(2)} | 0.80 | ${v.score >= 0.8 ? '?' : ''} | ).join("\n");
github.rest.issues.createComment({ issue_number: context.issue.number, owner:
context.repo.owner, repo: context.repo.repo, body });
# ----- # JOB 5:
Build Docker image # -----
---- build: name: "? Build Image" runs-on: ubuntu-latest needs: [unit-tests, security-scan]
if: github.event_name == 'push' && github.ref == 'refs/heads/main' outputs: image_tag: ${
steps.meta.outputs.tags } image_digest: ${ steps.build.outputs.digest }
steps: - uses: actions/checkout@v4
- name: Set up Docker Buildx uses: docker/setup-buildx-action@v3
- name: Log in to GitHub Container Registry uses: docker/login-action@v3 with: registry:
ghcr.io username: ${ github.actor } password: ${ secrets.GITHUB_TOKEN }
- name: Extract metadata id: meta uses: docker/metadata-action@v5 with: images: ghcr.io/${
github.repository }/${ env.AGENT_SERVICE_NAME } tags: | type=sha,prefix=sha-
type=raw,value=latest,enable=${is_default_branch}
- name: Build and push Docker image id: build 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

```

----- # JOB 6:

Deploy to Staging # -----

--- deploy-staging: name: "? Deploy to Staging" runs-on: ubuntu-latest needs: build

environment: name: staging url: \${ env.STAGING_URL }

steps: - name: Azure Login uses: azure/login@v2 with: creds: \${ secrets.AZURE_CREDENTIALS }

- name: Deploy to Azure Container Apps (Staging) uses:

azure/container-apps-deploy-action@v1 with: appSourcePath: \${ github.workspace } acrName:

outdoorgearacr containerAppName: \${ env.AGENT_SERVICE_NAME }-staging resourceGroup:

rg-outdoorgear-staging imageToDeploy: ghcr.io/\${ github.repository }/\${

env.AGENT_SERVICE_NAME }:sha-\${ github.sha }

- name: Smoke test staging run: | sleep 30 # Wait for deployment curl -f "\${

env.STAGING_URL }/health" || exit 1 echo "? Staging deployment healthy"

----- # JOB 7:

Deploy to Production (manual approval required) #

deploy-production: name: "? Deploy to Production" runs-on: ubuntu-latest needs:

[deploy-staging, agent-evaluation] if: | always() && needs.deploy-staging.result ==

'success' && (needs.agent-evaluation.result == 'success' || needs.agent-evaluation.result

== 'skipped') environment: name: production url:

"https://outdoorgear-agent.azurecontainerapps.io"

steps: - name: Azure Login uses: azure/login@v2 with: creds: \${ secrets.AZURE_CREDENTIALS }

- name: Deploy to Azure Container Apps (Production) uses:

azure/container-apps-deploy-action@v1 with: appSourcePath: \${ github.workspace } acrName:

outdoorgearacr containerAppName: \${ env.AGENT_SERVICE_NAME } resourceGroup:

rg-outdoorgear-prod imageToDeploy: ghcr.io/\${ github.repository }/\${

env.AGENT_SERVICE_NAME }:sha-\${ github.sha }

- name: Production health check run: | sleep 30 curl -f

"https://outdoorgear-agent.azurecontainerapps.io/health" || exit 1 echo "? Production deployment successful"

- name: Create GitHub deployment status uses: actions/github-script@v7 with: script: |

github.rest.repos.createDeploymentStatus({ owner: context.repo.owner, repo:

context.repo.repo, deployment_id: context.payload.deployment?.id, state: 'success',

description: 'Deployed to production', environment_url:

'https://outdoorgear-agent.azurecontainerapps.io' });