Testing

17-356/17-766
Software Engineering for Startups

https://cmu-17-356.github.io
Andrew Begel and Fraser Brown

Admin:

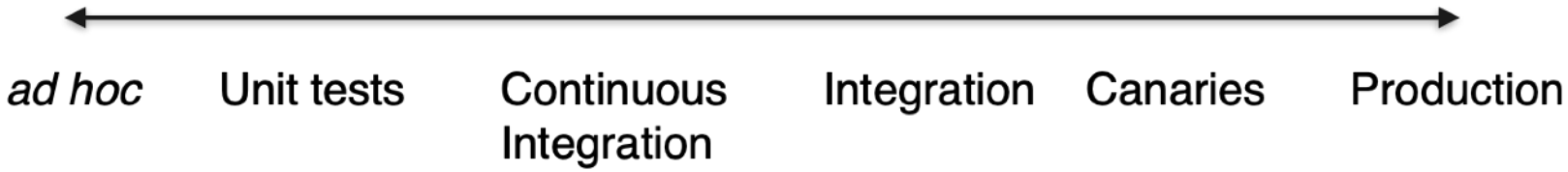
- HW4 is out
- Today: P1 check in for last 10-15 mins of class

Testing Myths

- “No tests” arguments
  - I’m a good programmer, I don’t need to prove myself.
  - I already tested this manually!
- Reality:
  - Tests aren’t about now, they are about from now on.
  - You need a fast, consistent, reliable signal for whether later changes have broken/disrupted your current feature.

Lifetime of Code

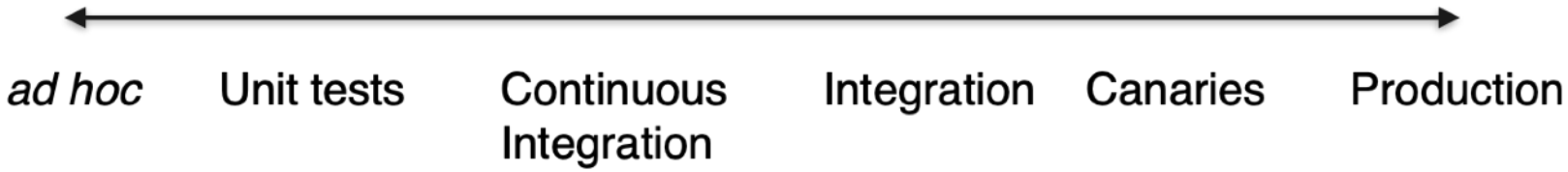
Ways to Test


ad hoc
Unit tests
Continuous Integration
Integration
Canaries
Production

Production Testing

- Most faithfully captures real-world scenarios: it is the real world!
- Relies upon users to detect and report problems.
- Expensive to fix bugs found in production.

Ways to Test

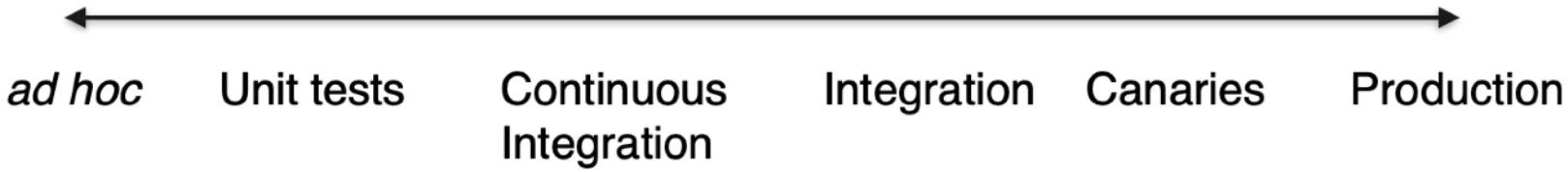

ad hoc
Unit tests
Continuous Integration
Integration
Canaries
Production

Canaries

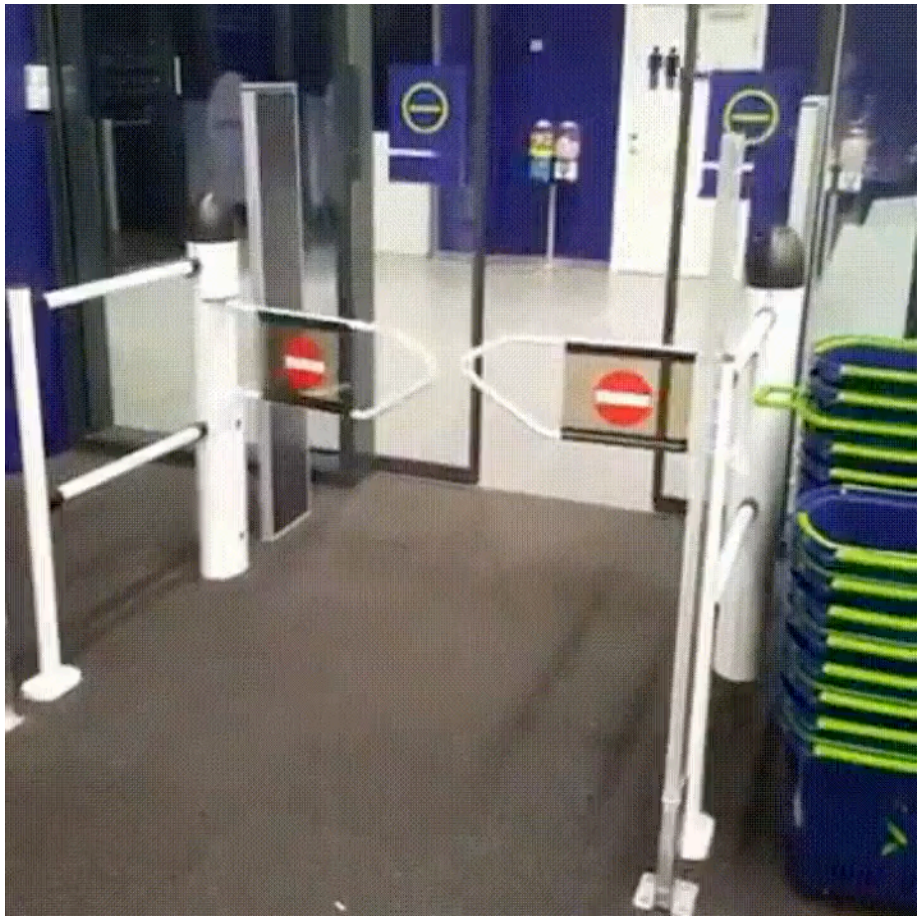
Canaries

- “Staged rollout”
- Still serves production traffic
  - But only a subset
- Sometimes more heavily instrumented
  - Lower performance, but better information about when something goes wrong.
- Still expensive to fix
  - But the users haven’t seen it yet.

Ways to Test


ad hoc
Unit tests
Continuous Integration
Integration
Canaries
Production

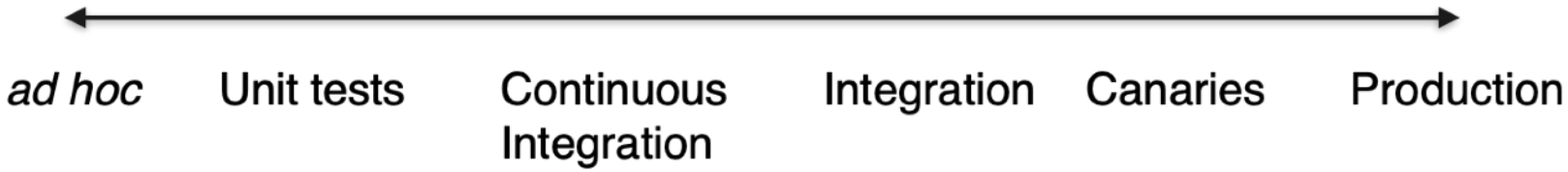
Integration Tests



Integration Tests

- Tests the combination of smaller components into large ones.
- Individual components can work correctly but fail when put together.
- Still expensive to fix failures detected here
  - Usually involves fixing interfaces between multiple components

Ways to Test


ad hoc
Unit tests
Continuous Integration
Integration
Canaries
Production


Unit tests: Fail
Dolores
RESTAURANT
Integration test: Pass

Unit Tests

- Designed to test a small unit of functionality
- Runs very quickly
  - An integral part of the edit-compile-test loop
- Small and self-contained: doesn't require interaction with external systems

Unit tests: “given input x, is output y?”

The act of writing a unit test is more an act of design than of **verification**.

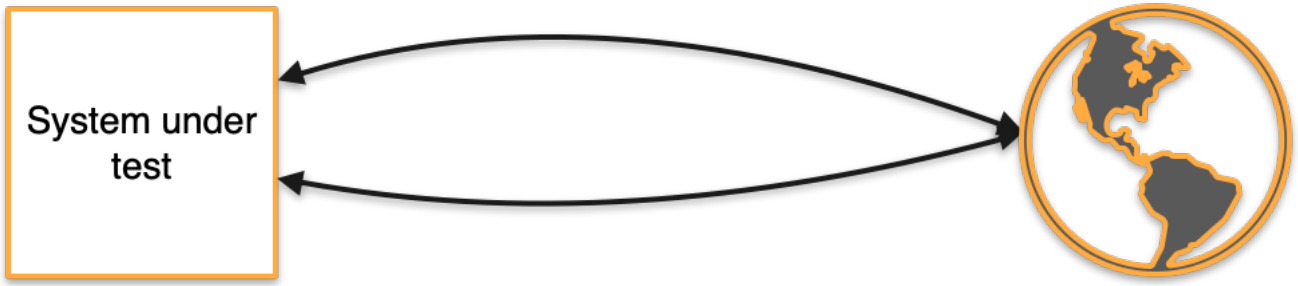
It is also more an act of documentation than of verification.

The act of writing a unit test closes a remarkable number of feedback loops, the least of which is the one pertaining to verification of function.

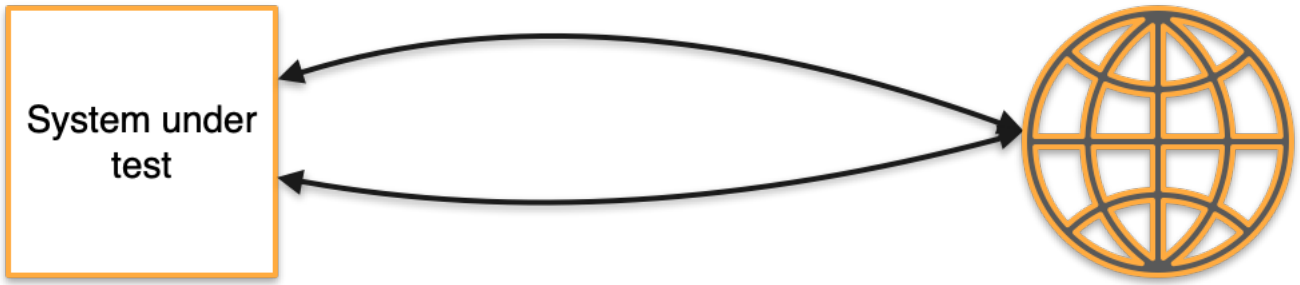
Tests should have a single reason to fail

```
@Test
public void testAddPositive() {

    // Instantiate the object to test
    Calculator tester = new Calculator();

    // the numbers used in the test
    Integer[] list = {1,2,3};

    assertEquals(null, tester.calculate(null,"+"));
    assertEquals(Integer.valueOf(6), tester.calculate(list,"+"));
}
```

Fakes and Mocks

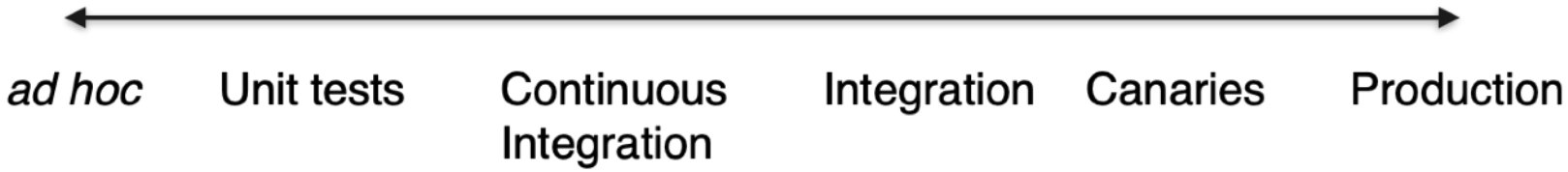


Fakes and Mocks



It’s possible to write code that’s very hard to unit test! How?

Ways to Test


ad hoc
Unit tests
Continuous Integration
Integration
Canaries
Production

ad hoc testing

- “Kicking the tires” basic testing
- Easy to setup, not guaranteed to catch much

If you find yourself running the same tests manually multiple times, it’s good sign that it’s time to invest in better testing infrastructure.

Test Driven Development (TDD)

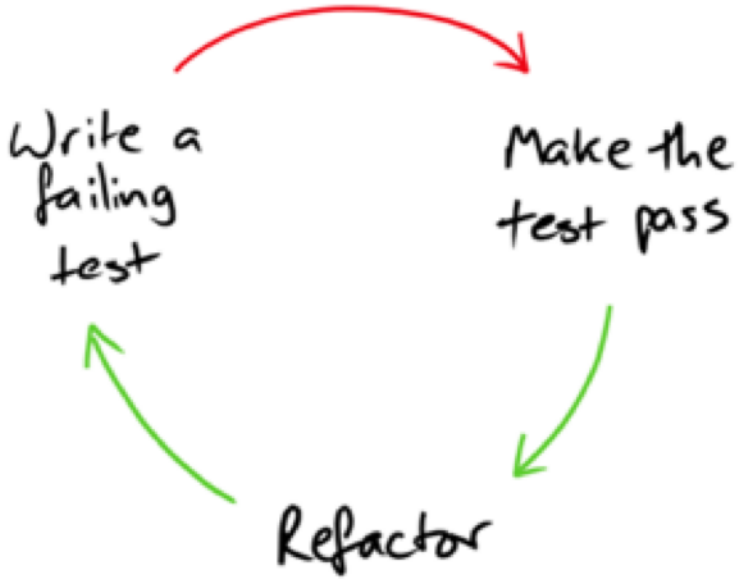
Three Simple Rules

1) You are not allowed to write any production code unless it is to make a failing unit test pass.

2) You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures.

3) You are not allowed to write any more production code than is sufficient to pass the one failing unit test.

Test Driven Development (TDD)



From Growing Object-Oriented Software by Nat Pryce and Steve Freeman
http://www.growing-object-oriented-software.com/figures.html

@sebrose

http://cucumber.io

Advantages of TDD

Clear place to start

Much less code thrown away, less wasted effort

Less Fear

Side Effect: Robust test suite

Disadvantages of strict TDD?

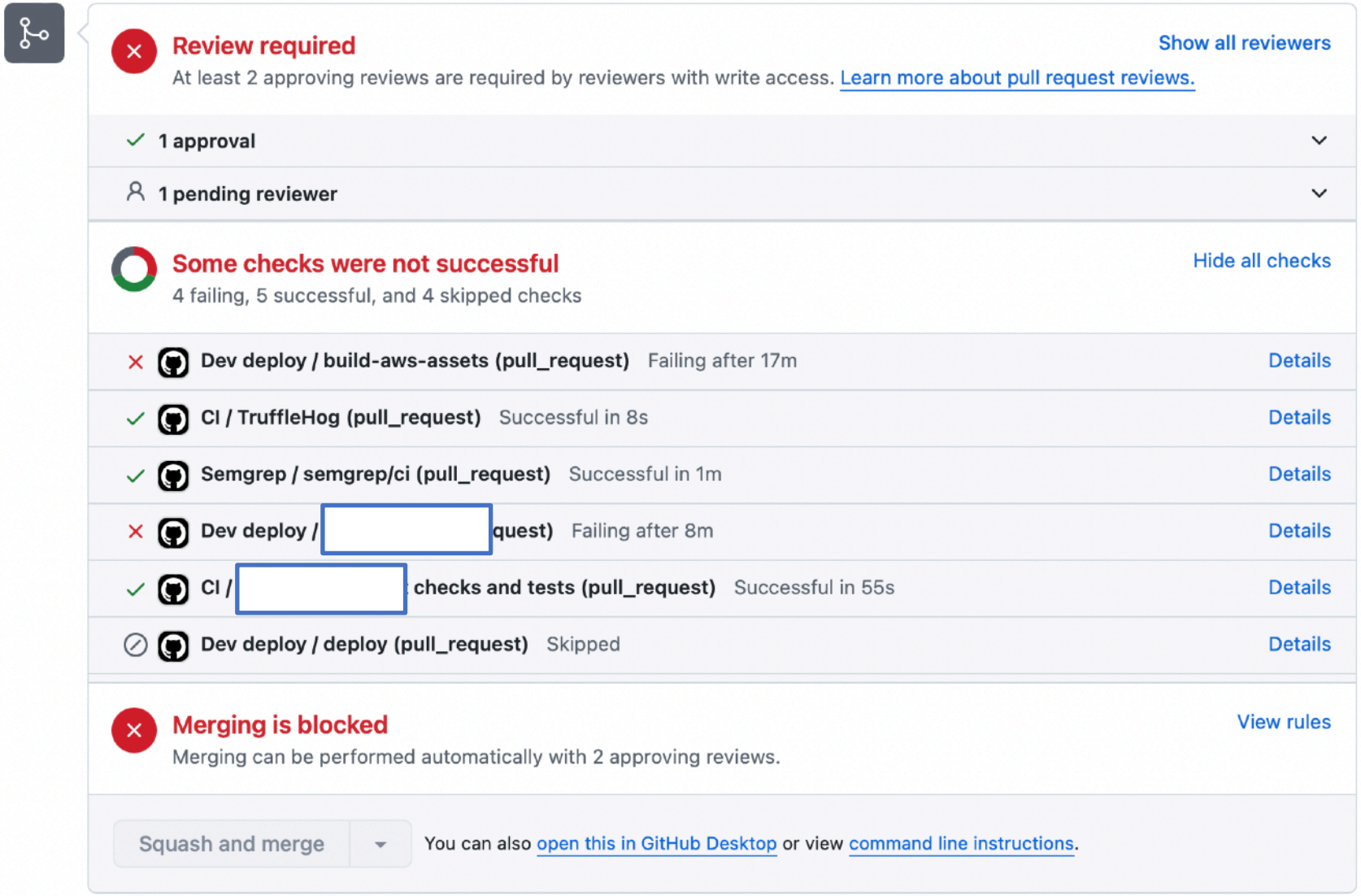
Continuous Integration

- Submitting pull requests early and often, and running tests over the resulting branch.
- Integrates directly with the version control system to run existing tests.
- Catches failures earlier, but fidelity is not a good as production.

When do you stop testing?

CI whirlwind tour

Tests must pass before you can merge



Github workflows

The components of GitHub Actions

You can configure a GitHub Actions **workflow** to be triggered when an **event** occurs in your repository, such as a pull request being opened or an issue being created. Your workflow contains one or more **jobs** which can run in sequential order or in parallel. Each job will run inside its own virtual machine **runner**, or inside a container, and has one or more **steps** that either run a script that you define or run an **action**, which is a reusable extension that can simplify your workflow.

https://docs.github.com/en/actions/about-github-actions/understanding-github-actions

Github workflows

- .github/workflows directory defines automated processes that run…

Github workflows

- .github/workflows directory defines automated processes that run…

```yaml
on:
  workflow_dispatch:
  pull_request:
    types: [synchronize, opened, reopened, ready_for_review]
    paths-ignore:
      - '**.md'
  push:
    branches: ["main"]
    paths-ignore:
      - '**.md'

name: CI
```

Github workflows

- .github/workflows directory defines automated processes that run…

```yaml
on:
  workflow_dispatch:
  pull_request:
    types: [synchronize, opened, reopened, ready_for_review]
    paths-ignore:
      - '**.md'
  push:
    branches: ["main"]
    paths-ignore:
      - '**.md'

name: CI
```

Github workflows

- .github/workflows directory defines automated processes that run…

```yaml
on:
  workflow_dispatch:
  pull_request:
    types: [synchronize, opened, reopened, ready_for_review]
    paths-ignore:
      - '**.md'
  push:
    branches: ["main"]
    paths-ignore:
      - '**.md'

name: CI
```

So much customization in the docs

`pull_request`

Webhook event payload	Activity types	`GITHUB_SHA`	`GITHUB_REF`
`pull_request`	- `assigned` - `unassigned` - `labeled` - `unlabeled` - `opened` - `edited` - `closed` - `reopened` - `synchronize` - `converted_to_draft` - `locked` - `unlocked` - `enqueued` - `dequeued` - `milestoned` - `demilestoned` - `ready_for_review` - `review_requested` - `review_request_removed` - `auto_merge_enabled` - `auto_merge_disabled`	Last merge commit on the `GITHUB_REF` branch	PR merge branch `refs/pull/PULL_REQUEST_NUMBER/merge`

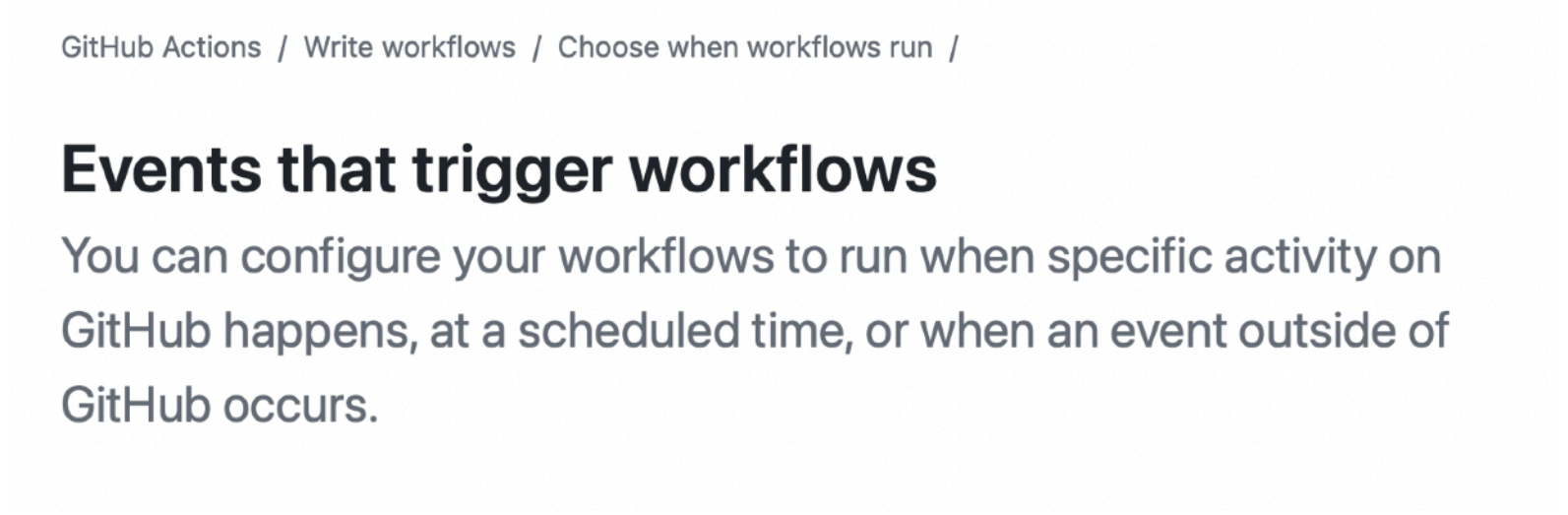
Github workflows

- .github/workflows directory defines automated processes that run…

```yaml
on:
  workflow_dispatch:
  pull_request:
    types: [synchronize, opened, reopened, ready_for_review]
    paths-ignore:
      - '**.md'
  push:
    branches: ["main"]
    paths-ignore:
      - '**.md'

name: CI
```

Github workflows

- .github/workflows directory defines automated processes that run…whenever: https://docs.github.com/en/actions/writing-workflows/choosing-when-your-workflow-runs/events-that-trigger-workflows



Github workflows

- .github/workflows directory defines automated processes that do…something…somewhere:

```yaml
jobs:
  TruffleHog:
    runs-on: ubuntu-latest
    steps:
      - name: Check out repo
        uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Run TruffleHog on changes in PR
        uses: trufflesecurity/trufflehog@v3.26.0
        with:
          path: ./
          base: ${{ github.event.repository.default_branch }}
          head: HEAD
          extra_args: --debug --only-verified
```

All this stuff actually has to run on a real machine

All this stuff actually has to run on a real machine. Two choices

- GH hosted runners
- Self-hosted runners

Choosing GitHub-hosted runners

If you use a GitHub-hosted runner, each job runs in a fresh instance of a runner image specified by `runs-on`.

The value for runs-on, when you are using a GitHub-hosted runner, is a runner label or the name of a runner group. The labels for the standard GitHub-hosted runners are shown in the following tables.

All this stuff actually has to run on a real machine. Two choices

- GH hosted runners: https://docs.github.com/en/actions/writing-workflows/choosing-where-your-workflow-runs/choosing-the-runner-for-a-job
- Self-hosted runners: https://docs.github.com/en/actions/writing-workflows/choosing-where-your-workflow-runs/choosing-the-runner-for-a-job

```yaml
jobs:
  TruffleHog:
    runs-on: ubuntu-latest
    steps:
      - name: Check out repo
        uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Run TruffleHog on changes in PR
        uses: trufflesecurity/trufflehog@v3.26.0
        with:
          path: ./
          base: ${{ github.event.repository.default_branch }}
          head: HEAD
          extra_args: --debug --only-verified
```

```yaml
jobs:
  TruffleHog:
    runs-on: ubuntu-latest
    steps:
      - name: Check out repo
        uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Run TruffleHog on changes in PR
        uses: trufflesecurity/trufflehog@v3.26.0
        with:
          path: ./
          base: ${{ github.event.repository.default_branch }}
          head: HEAD
          extra_args: --debug --only-verified
```

```
18    jobs:
19      TruffleHog:
20        runs-on: ubuntu-latest
21        steps:
22          - name: Check out repo
23            uses: actions/checkout@v4
24            with:
25              fetch-depth: 0
26
27          - name: Run TruffleHog on changes in PR
28            uses: trufflesecurity/trufflehog@v3.26.0
29            with:
30              path: ./
31              base: ${{ github.event.repository.default_branch }}
32              head: HEAD
33              extra_args: --debug --only-verified
```

Look at config info for tool you're using

TruffleHog Github Action

General Usage

```
on:
  push:
    branches:
      - main
  pull_request:

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout code
      uses: actions/checkout@v4
      with:
        fetch-depth: 0
    - name: Secret Scanning
      uses: trufflesecurity/trufflehog@main
      with:
        extra_args: --results=verified,unknown
```

In the example config above, we're scanning for live secrets in all PRs and Pushes to `main`. Only code changes in the referenced commits are scanned. If you'd like to scan an entire branch, please see the "Advanced Usage" section below.

Shallow Cloning

If you're incorporating TruffleHog into a standalone workflow and aren't running any other CI/CD tooling alongside TruffleHog, then we recommend using Shallow Cloning to speed up your workflow. Here's an example for how to

Advanced Usage: Scan entire branch

```
- name: scan-push
        uses: trufflesecurity/trufflehog@main
        with:
          base: ""
          head: ${{ github.ref_name }}
          extra_args: --results=verified,unknown
```

When you’re working on your projects, *documentation is your best friend.*

Learn how your tools work. Don’t just learn incantations.

```yaml
35            -tests:
36        runs-on: ubuntu-latest
37        name: Run              checks and tests
38        steps:
39
40          - name: Check out repo
41            uses: actions/checkout@v4
42            with:
43              fetch-depth: 1
44
45          - name: Install packages
46            working-directory: ./
47            run: npm ci
48
49          - name: Code generation
50            working-directory: ./
51            run: npm run compile
52
53          - name: Run linter
54            working-directory: ./
55            run: npm run lint
56
57          - name: Build
58            working-directory: ./
59            run: npm run build
60
61          - name: Run |       tests
62            working-directory: ./
63            run: npm test
64
65          - name: Code coverage
66            working-directory: ./
```

CI for a large product is much more complex. How/why?

Last 10-15 minutes: P1 checkins