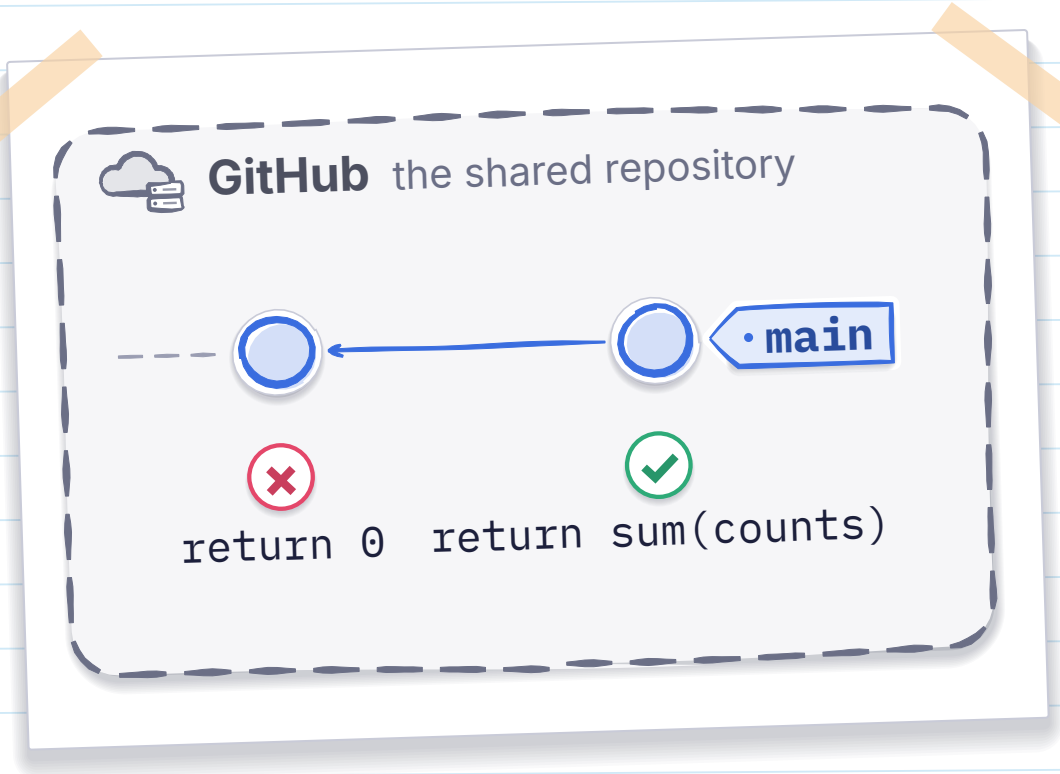
Prof. Aviv Zohar

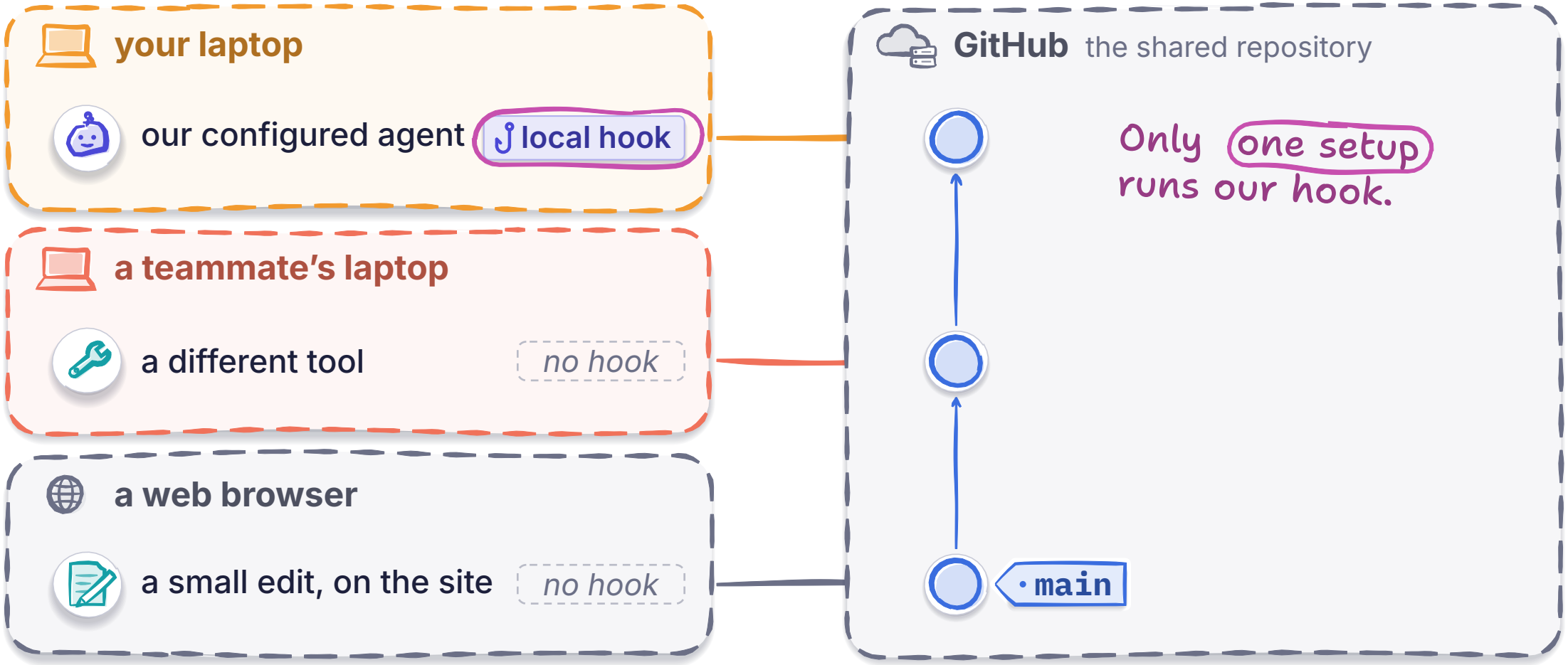
Semester A, 2026–7

CI with GitHub Actions

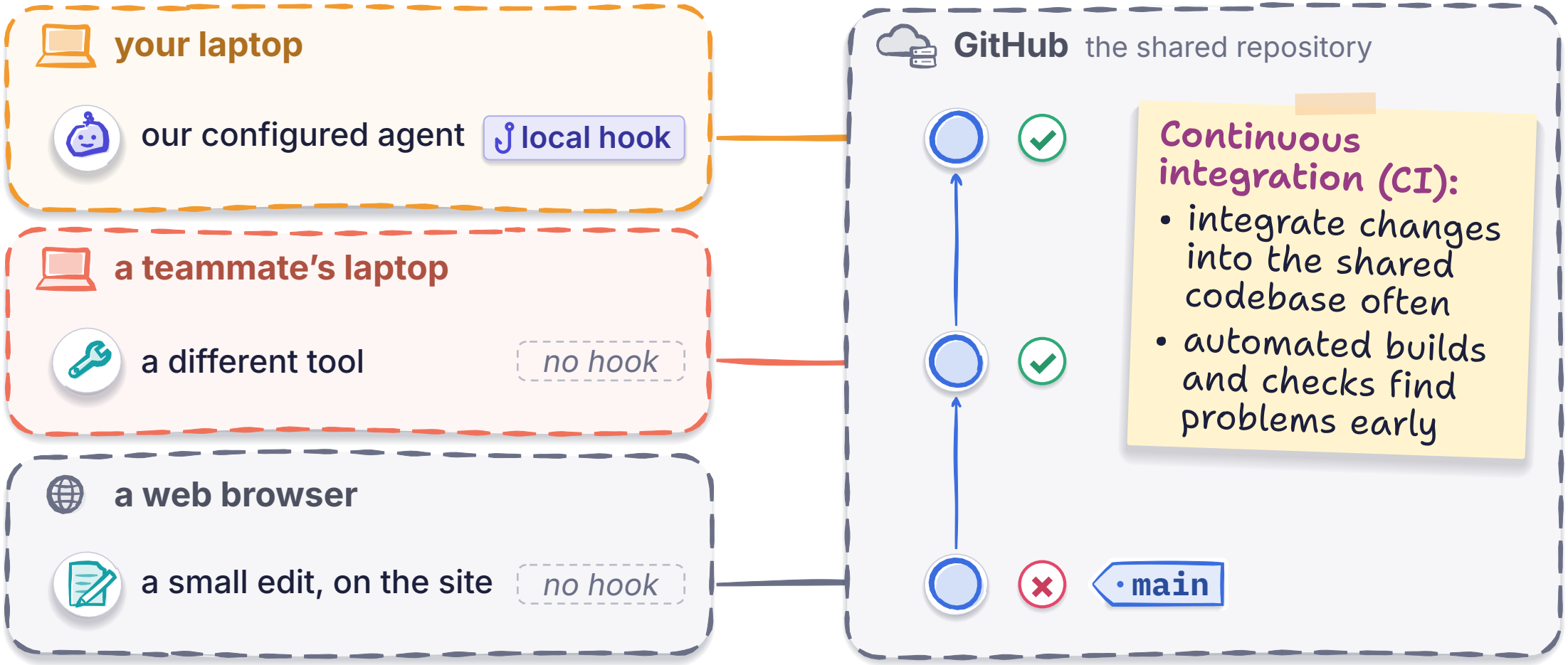
Run shared checks
when code is pushed



Changes reach the project from many setups

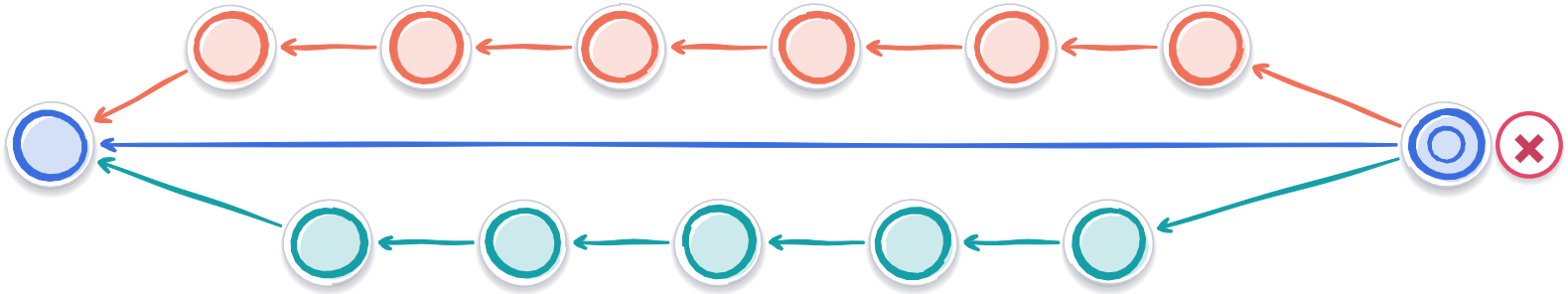


Changes reach the project from many setups



Integrate small changes, often

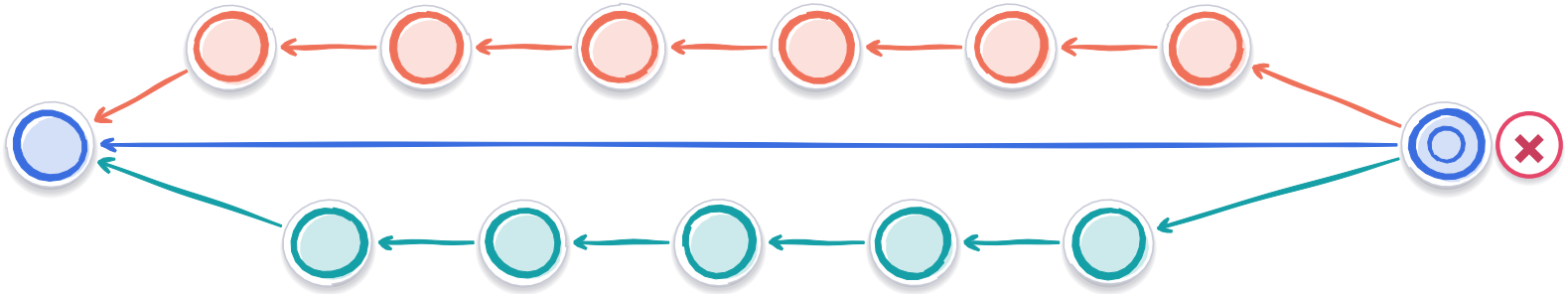
Separate work, combined at the end



Many changes to investigate, and assumptions that do not fit together.

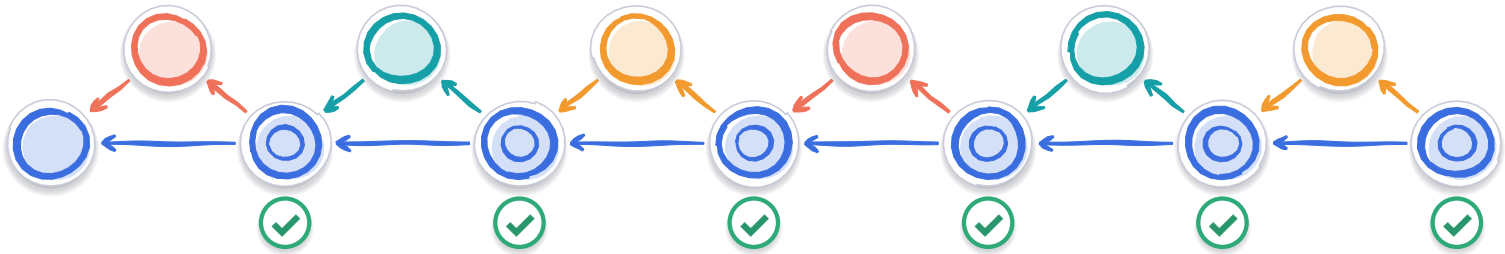
Integrate small changes, often

Separate work, combined at the end



Many changes to investigate, and assumptions that do not fit together.

Continuous integration



Feedback on each change while it is fresh.

Continuous delivery: keep the software ready to release. Integrating a change is not releasing it.

What GitHub's runners cost

Runners: the machines that execute our jobs.

Public repositories

Free

standard GitHub-hosted runners

Private repositories · GitHub Free

2,000 minutes a month

shared across the owner's private repositories

Useful for small personal projects. Larger runners and use beyond the allowance cost extra.

Other plans have other allowances.

GitHub Actions billing documentation, checked 30 September 2026

One mistaken change, pushed to main

src/counts.py our change

```
1  def total_items(counts: list[int]) -> int:
-      return sum(counts)
+      return 0
```

tests/test_counts.py

```
4  def test_total_items() -> None:
5      assert total_items([2, 3]) == 5
```

For [2, 3] the test expects 5 but the change returns 0.



illustrative ids and run

One mistaken change, pushed to main

src/counts.py our change

```
1  def total_items(counts: list[int]) -> int:
-      return sum(counts)
+      return 0
```

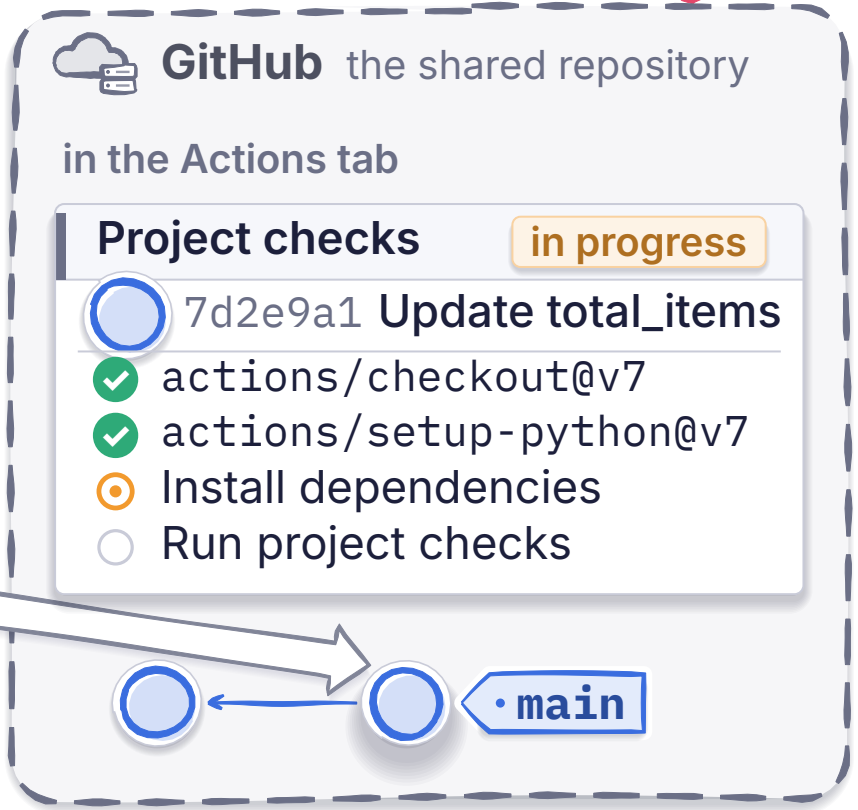
tests/test_counts.py

```
4  def test_total_items() -> None:
5      assert total_items([2, 3]) == 5
```

shell

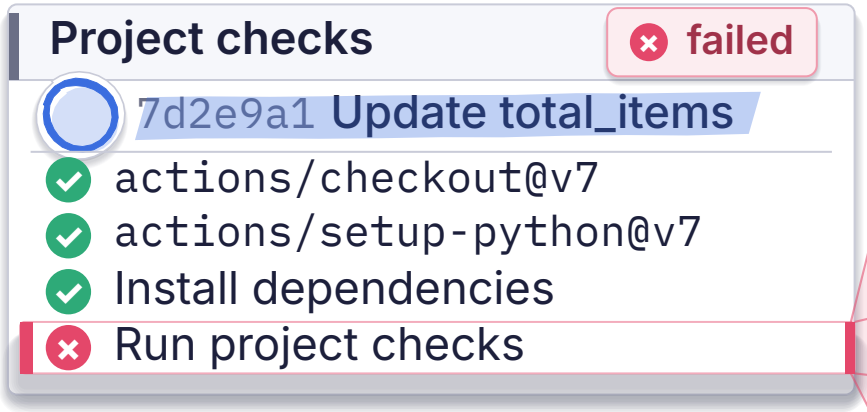
```
~/demo $ git commit -am "Update total_items"
~/demo $ git push origin main
```

For [2, 3] the test expects 5 but the change returns 0.



illustrative ids and run

The failure is reported on the pushed commit



The result belongs to this commit. Everyone can read the command and its output.

log · Run project checks

```
Run bash tools/check.sh
    def test_total_items() -> None:
>       assert total_items([2, 3]) == 5
E       assert 0 == 5
E        +  where 0 = total_items([2, 3])
tests/test_counts.py:5: AssertionError
FAILED tests/test_counts.py::test_total_items
1 failed in 0.01s
```

The test expects 5 and receives 0.

run illustrative · test output from the example project, cropped

The failure is reported on the pushed commit

Project checks — failed

`7d2e9a1` Update total_items

- ✓ actions/checkout@v7
- ✓ actions/setup-python@v7
- ✓ Install dependencies
- ✗ Run project checks

log · Run project checks

```
Run bash tools/check.sh
    def test_total_items() -> None:
>       assert total_items([2, 3]) == 5
E       assert 0 == 5
E        +  where 0 = total_items([2, 3])
tests/test_counts.py:5: AssertionError
FAILED tests/test_counts.py::test_total_items
1 failed in 0.01s
```

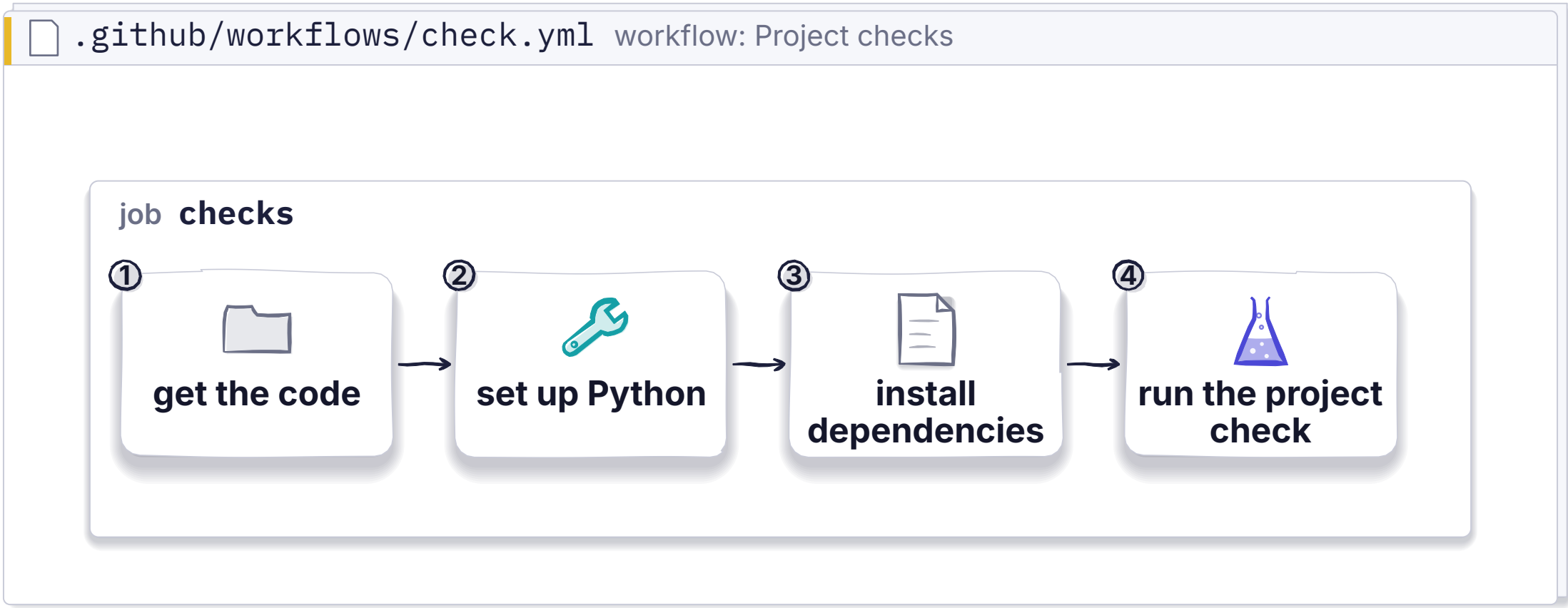
in the repository

- **.github/workflows/check.yml**
- src/counts.py
- tests/test_counts.py
- tools/check.sh

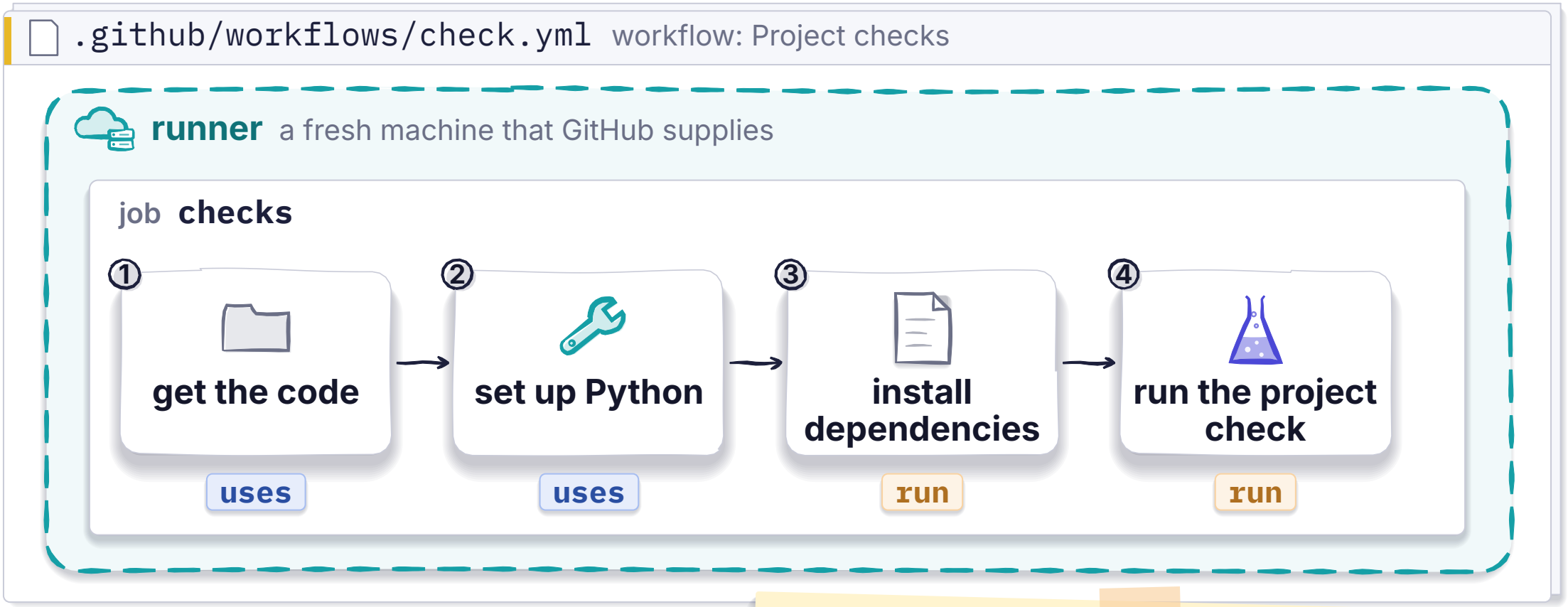
Tells GitHub Actions when to run and what to execute.

run illustrative · test output from the example project, cropped

A workflow holds jobs; a job holds steps



A workflow holds jobs; a job holds steps



uses: an action, reusable code for a task.
run: a shell command. The steps run in order on the runner.

The trigger and the job, in YAML

.github/workflows/check.yml

```yaml
name: Project checks
on:
  pull_request:
  push:
    branches: [main]
permissions:
  contents: read
jobs:
  checks:
    runs-on: ubuntu-latest
    steps:
      ⋮
```

the workflow's name
the events that start it:
every pull request
a push of code
to main only

YAML: a key and a colon introduce a value or a nested section. The indentation shows what belongs inside.

This file is committed and pushed with the project. Each later push to main starts another run.

The trigger and the job, in YAML

.github/workflows/check.yml

```yaml
name: Project checks
on:
  pull_request:
  push:
    branches: [main]
permissions:
  contents: read
jobs:
  checks:
    runs-on: ubuntu-latest
    steps:
```

the token may read the contents

our job's name

a hosted Ubuntu runner

next: the job's steps

The steps, in YAML

.github/workflows/check.yml continued

```yaml
    steps:
      - uses: actions/checkout@v7
      - uses: actions/setup-python@v7
        with:
          python-version: "3.12"
      - name: Install dependencies
        run: python -m pip install -r requirements-dev.txt
      - name: Run project checks
        run: bash tools/check.sh
```

Each `-` begins another step.

`uses` names an action: checkout gets the repository's files, setup-python installs Python.

`@v7` the action's own version

`"3.12"` the Python version, as in the local example

The steps, in YAML

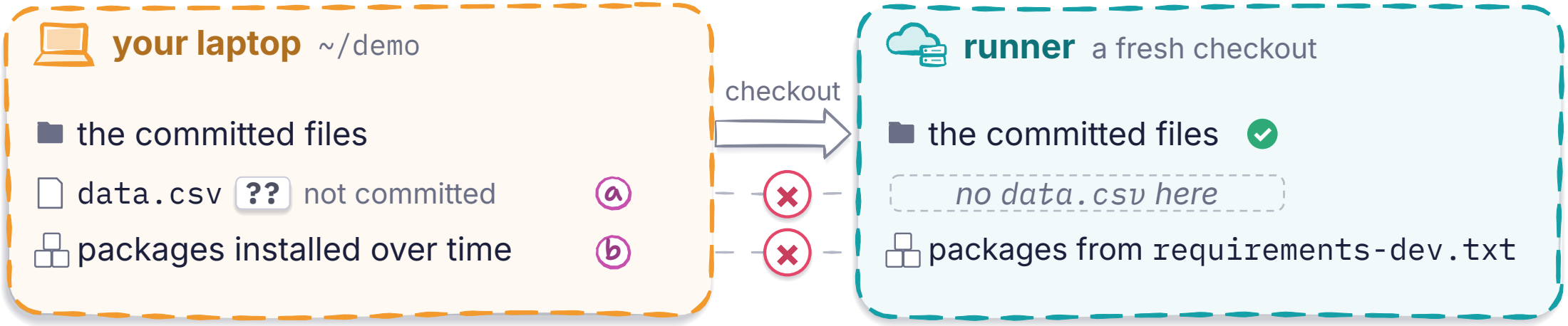
.github/workflows/check.yml continued

```yaml
    steps:
      - uses: actions/checkout@v7
      - uses: actions/setup-python@v7
        with:
          python-version: "3.12"
      - name: Install dependencies
        run: python -m pip install -r requirements-dev.txt
      - name: Run project checks
        run: bash tools/check.sh
```

`run` executes a shell command on the runner: install the dependencies, then run the checking script we use locally.

a checker fails → exit 1 → the step fails → GitHub reports the job as failed

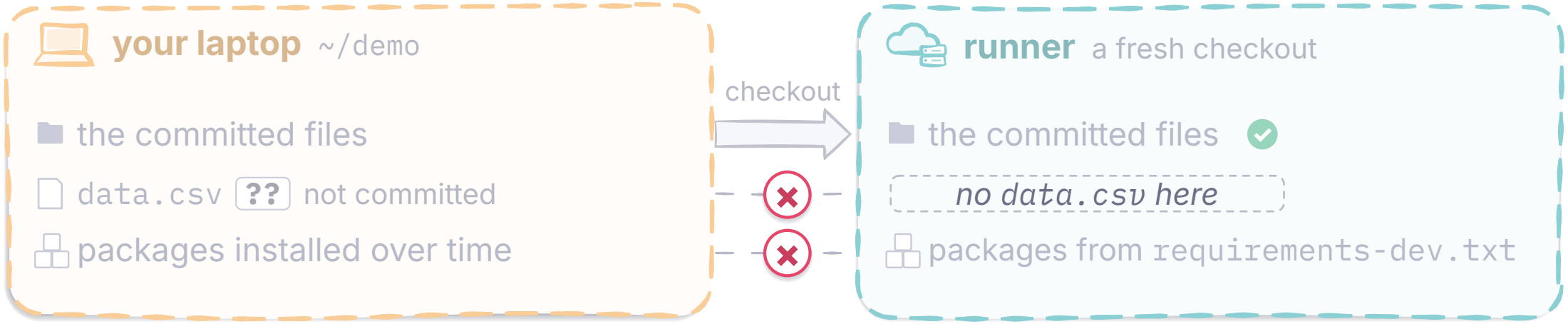
The runner has only what the workflow sets up



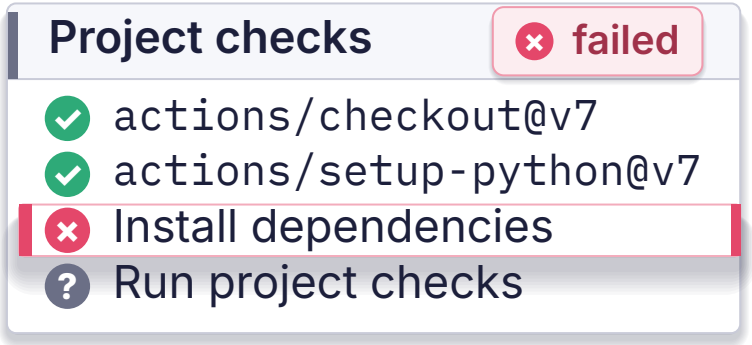
ⓐ Never committed, so the runner does not have `data.csv`.

ⓑ A package you installed by hand must also be in the runner's setup.

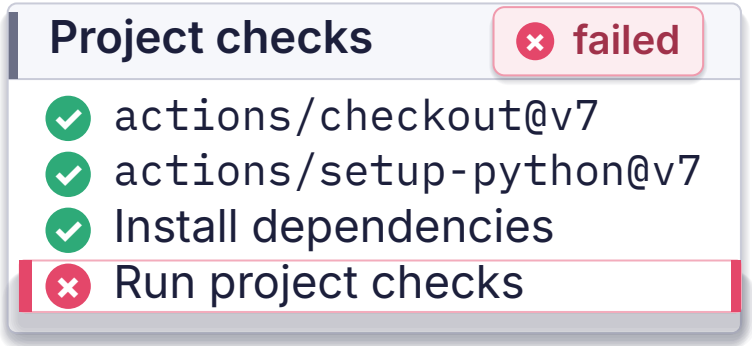
The runner has only what the workflow sets up



on GitHub, in the Actions tab: open the run



Installing failed: the checks never started.



A test ran and got an unexpected result.

or in a terminal

```shell
$ gh run view --log-failed
```

Reproduce it, correct it, check again

① reproduce on your laptop

shell

```
~/demo $ bash tools/check.sh
E       assert 0 == 5
1 failed in 0.01s
```

The test ran and found the wrong answer: 0, not 5.

② correct the function

src/counts.py the correction

```
-       return 0
+       return sum(counts)
```

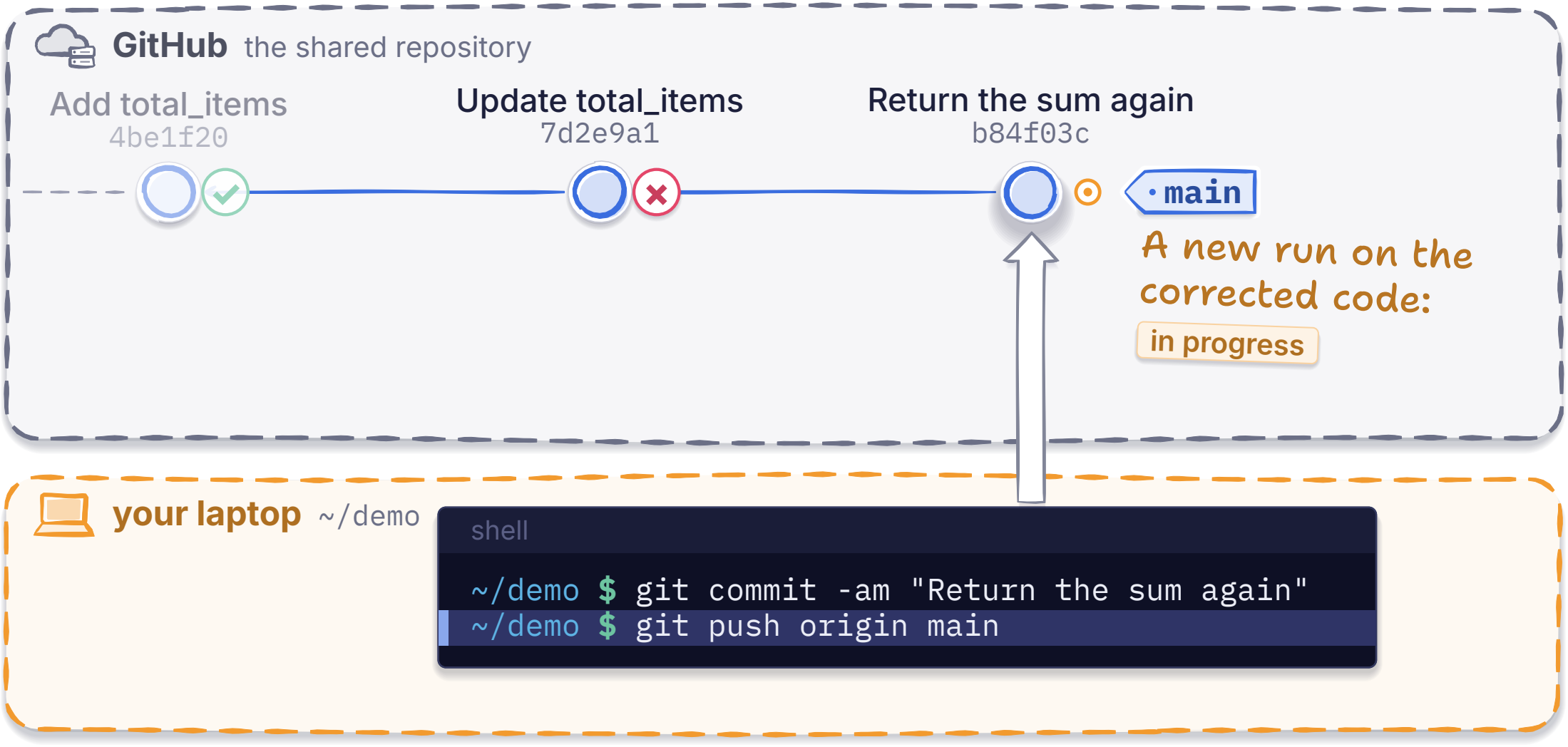
③ run the checks again

shell

```
~/demo $ bash tools/check.sh
1 passed in 0.01s
Success: no issues found in 1 source file
All checks passed!
7 files already formatted
```

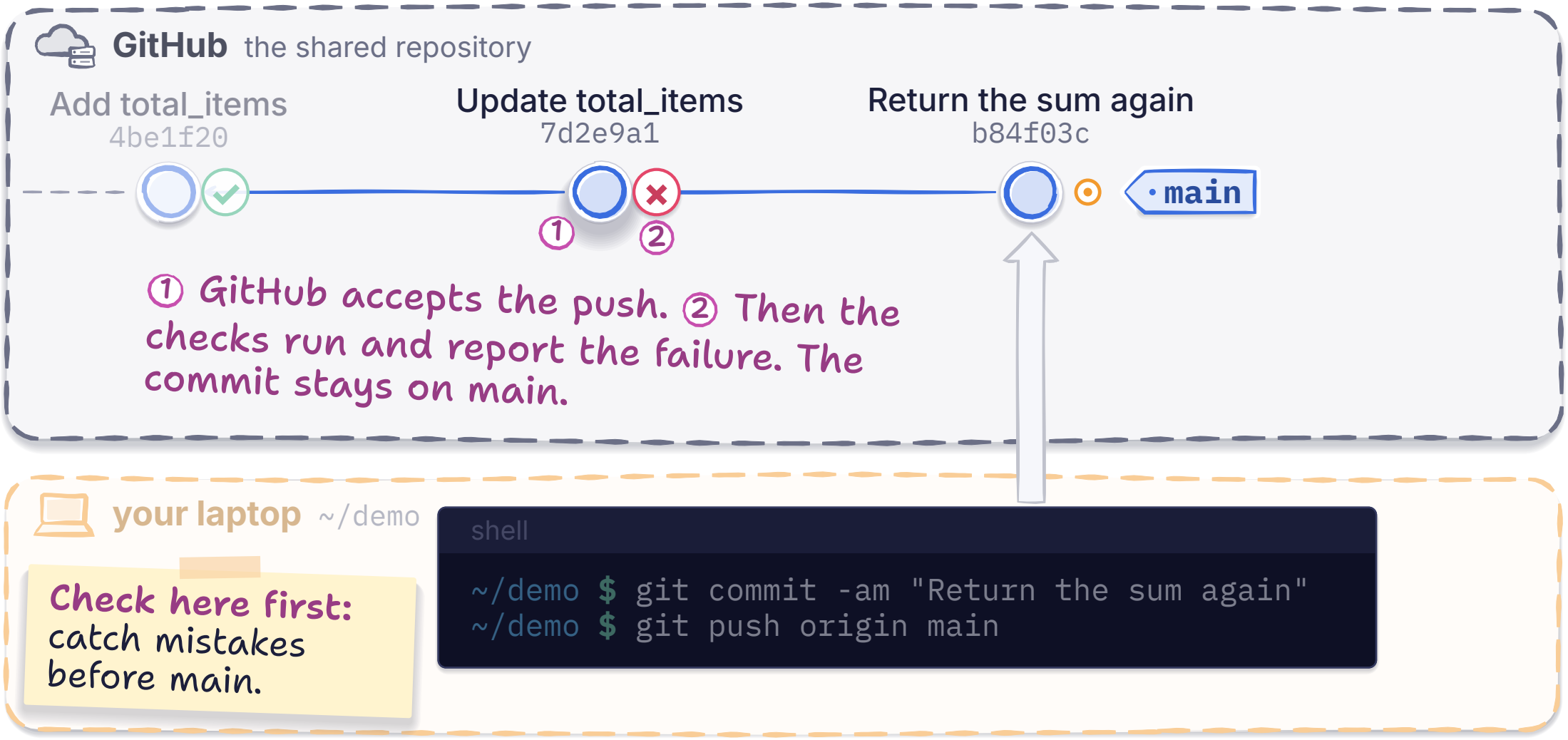
Review the diff as well: it shows what made the test pass.

A new push starts a new run



illustrative ids

A failed check does not undo a push



illustrative ids

To keep main working, require the check first

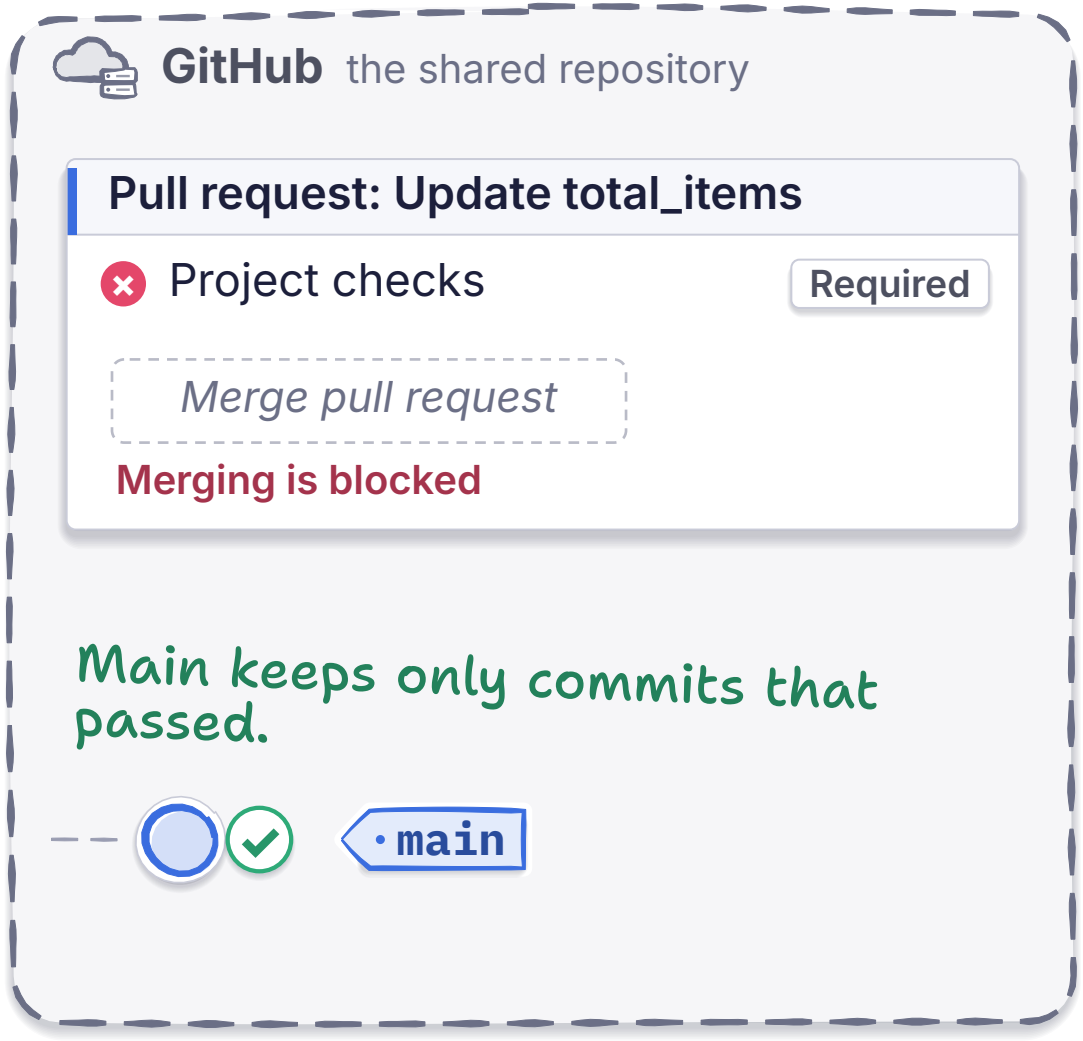
Ruleset for `main`

- Require a pull request before merging
- Require status checks to pass

  Project checks

```shell
~/demo $ git push origin main
! [remote rejected] main -> main
  (push declined due to repository
   rule violations)
```

A direct push to main is refused.



settings and pull request drawn

A pass means only what the checks cover

tools/check.sh a change to the checks

```
1 set -e
2 python -m pytest -q
- python -m mypy --strict src
3 python -m ruff check .
4 python -m ruff format --check .
```

Project checks passed

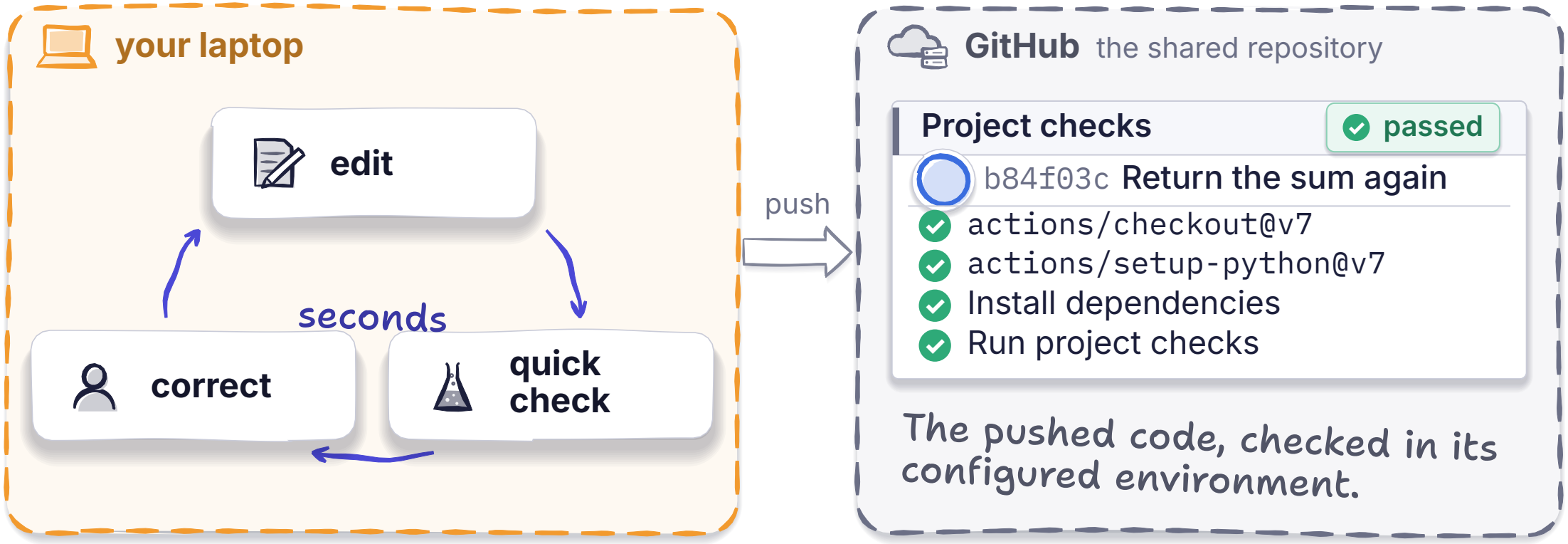
It passes because it no longer checks the types.

What a passing run depends on

- .github/workflows/check.yml
- tools/check.sh
- tests/

Review changes to these along with changes to the application code.

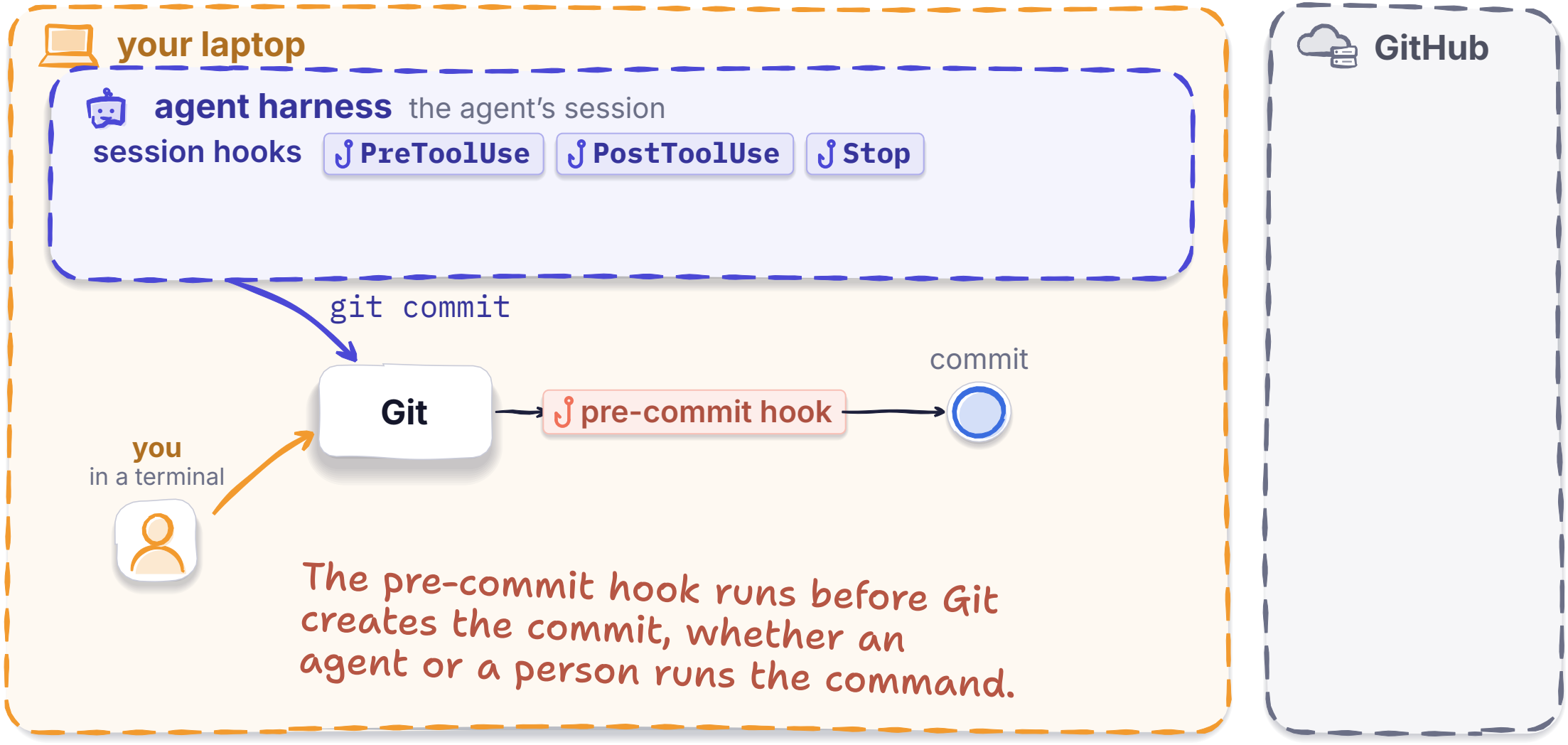
Fast feedback here, a shared result there



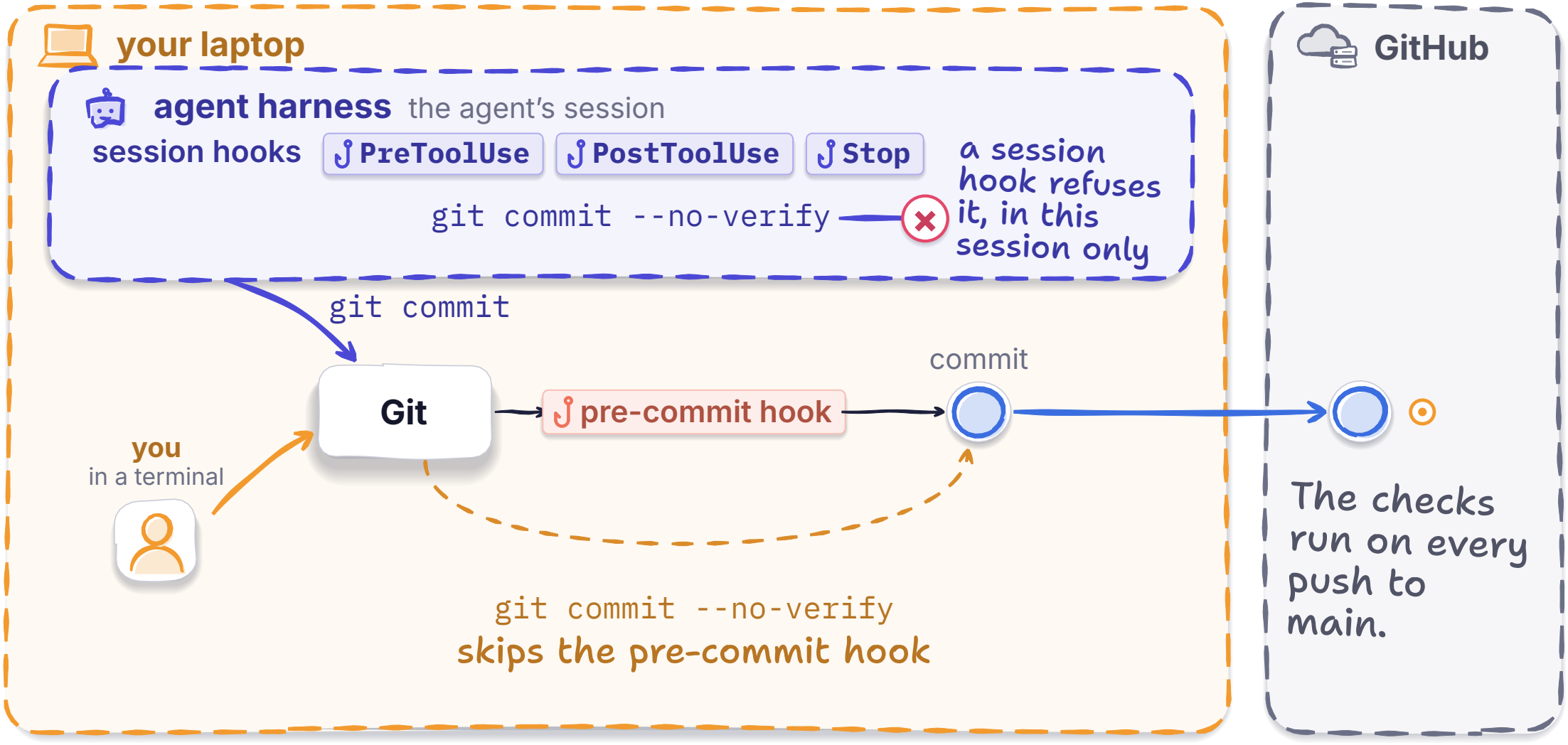
Some checks run twice, on purpose: the shared result does not depend on a report from one laptop.

illustrative run

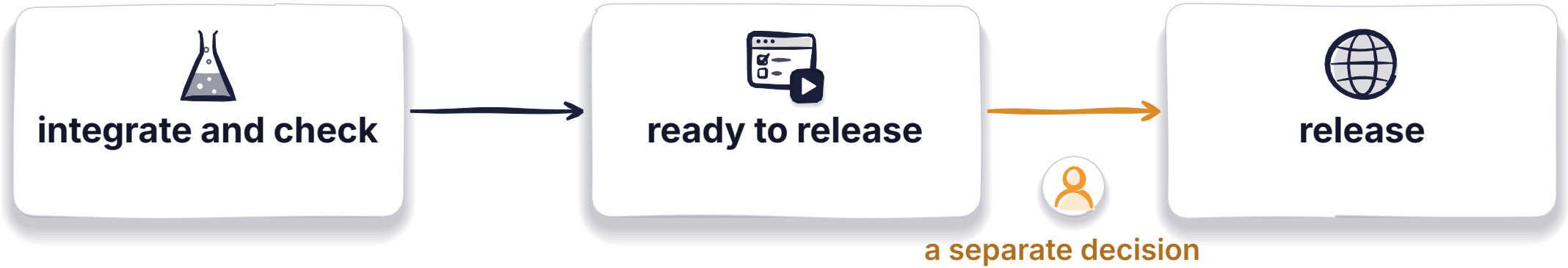
Git hooks run when Git commits, whoever asks



Git hooks run when Git commits, whoever asks

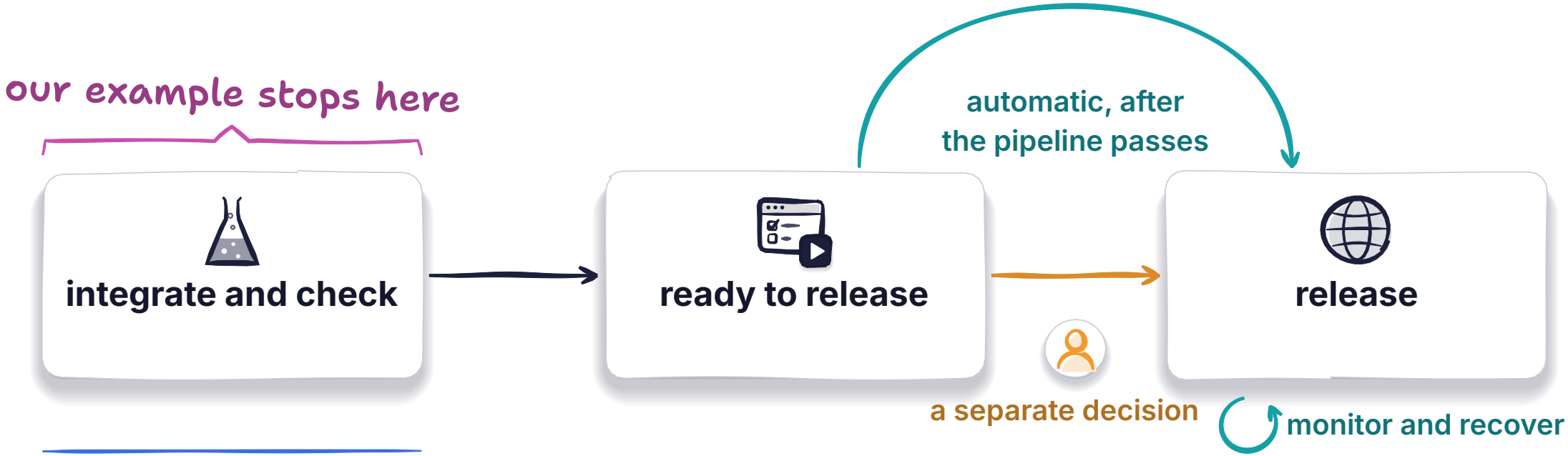


After the checks: delivery and deployment



A release workflow publishes only in a job with `needs: checks`.

After the checks: delivery and deployment



A release workflow publishes only in a job with `needs: checks`.

See you in the next video.