Azure Python CI/CD

Python CI/CD in an Azure DevOps Pipeline

Rik Watson 2021-07-02

❤️

Aims

What we aim to cover

- Discuss the structure of a basic Azure DevOps Pipeline and the tooling which runs alongside it
- Show example pipeline and run some tests through it
- There will be time for Q&A at the end but please ask questions
- Pipeline, repository & presentation will be made available at the end

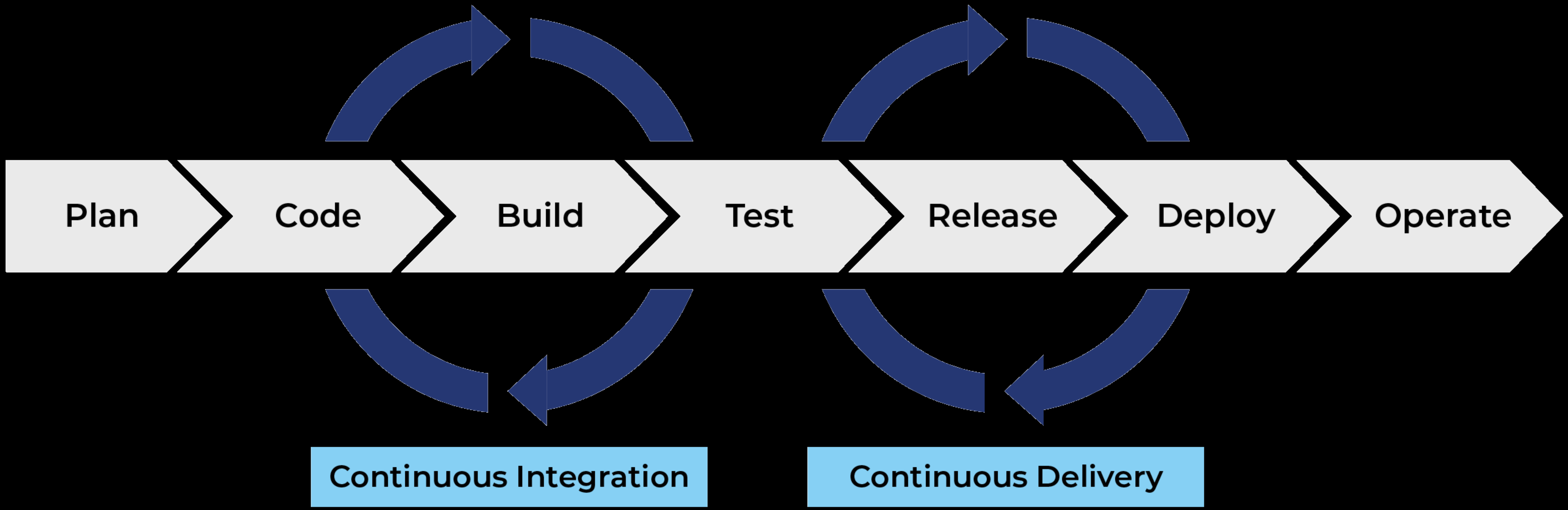
Pre Req’s

What we won’t be covering

- How to program in Python
- Python best practices - but we’ll be discussing how to automate them - just not what they are
- DVCS - assume some basic knowledge of git terminology
- TDD

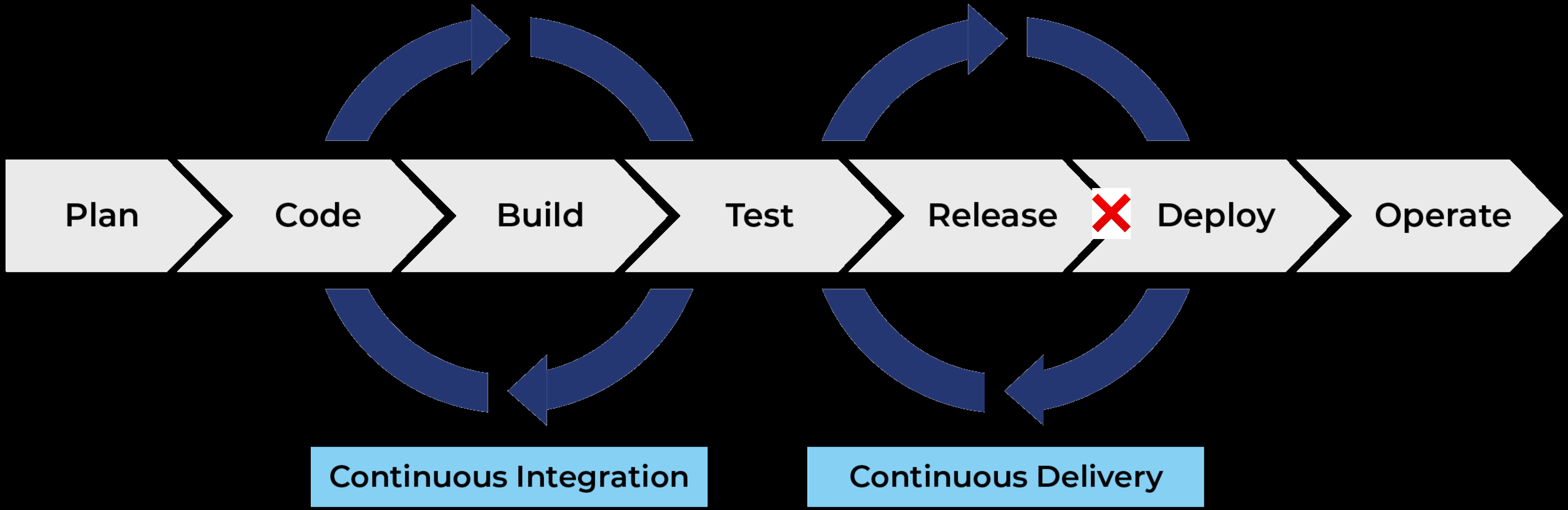
CI / CD

Continuous Integration / Continuous Deployment



CI / CDel

Continuous Integration / Continuous Delivery



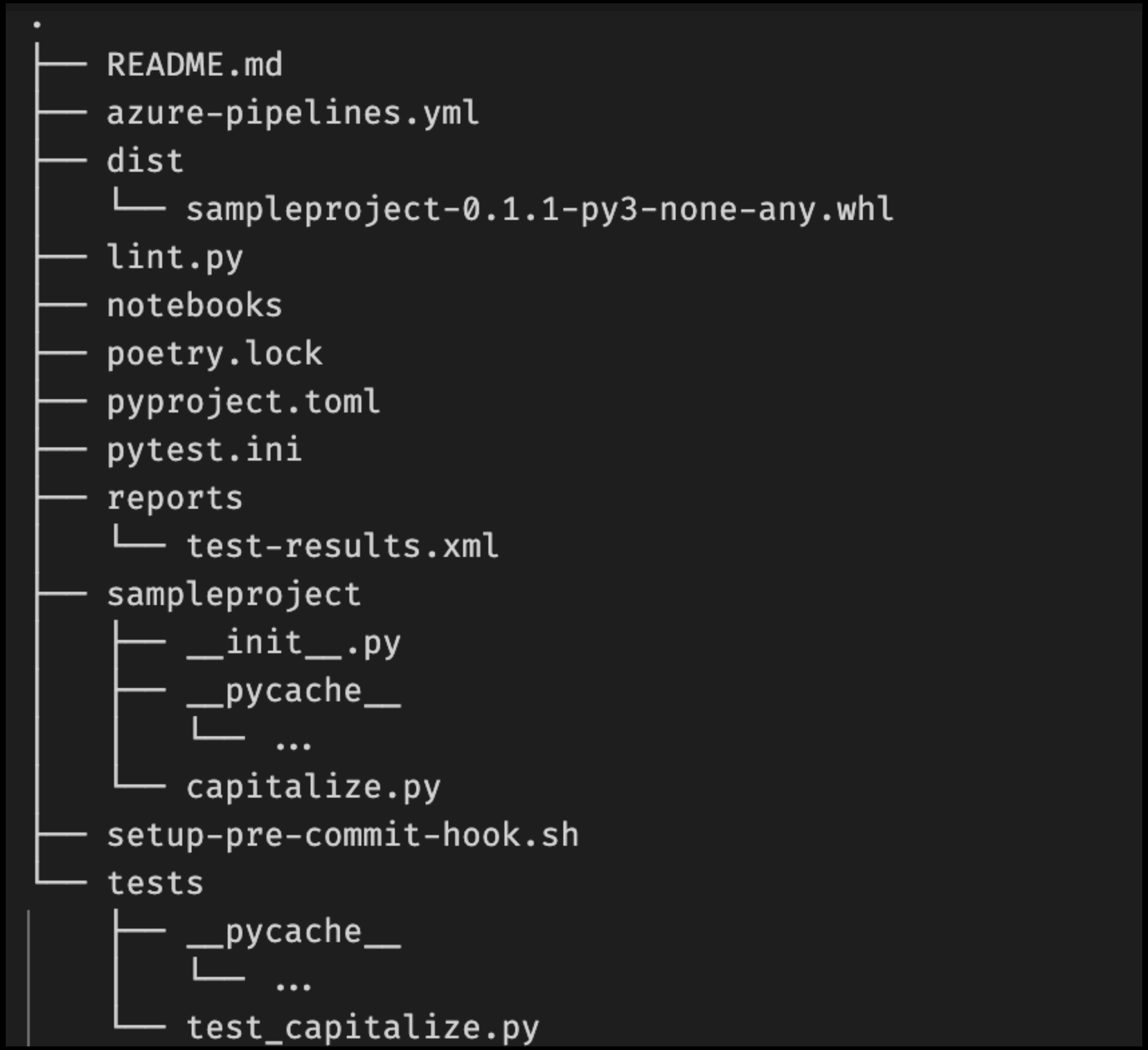
CI / CD / CDel ?

Continuous Integration / Continuous Deployment - Delivery

- Most teams start with CDel until they get the confidence to go ‘straight’ to production (Cont. Deployment)
- For Packages etc it’s common to have test and live artefact stores
- This will depend almost entirely on the project and how it is utilised
- The difference between CD and CDel lies in choice. With CD, code is automatically deployed after each successful main branch merge, while with CDel your organisation can choose whether it's released or not. Teams usually choose to stick with CDel & not progress to CD for business reasons

Directory Layout of example

Simplified



```
.
├── README.md
├── azure-pipelines.yml
├── dist
│   └── sampleproject-0.1.1-py3-none-any.whl
├── lint.py
├── notebooks
├── poetry.lock
├── pyproject.toml
├── pytest.ini
├── reports
│   └── test-results.xml
├── sampleproject
│   ├── __init__.py
│   ├── __pycache__
│   │   └── ...
│   └── capitalize.py
├── setup-pre-commit-hook.sh
└── tests
    ├── __pycache__
    │   └── ...
    └── test_capitalize.py
```

README.md

The heart of a project

- All repositories (and hence projects) should have a comprehensive README.md
- It should outline the tool chain used, how to set it up
- You should be able to take the README and build / test / deploy with little or no outside help

Tool chain

The right tool chain can make a project

- Although Python is famously an opinionated language it has a varied and flexible tool chain
- Remember you *MUST* use *EXACTLY* the same tools to build / test on your PC and your CI/CD platform
  - Tools like docker, pyenv & Poetry are your friends - use them
  - Use matrix builds to test multiple versions of language / tooling etc

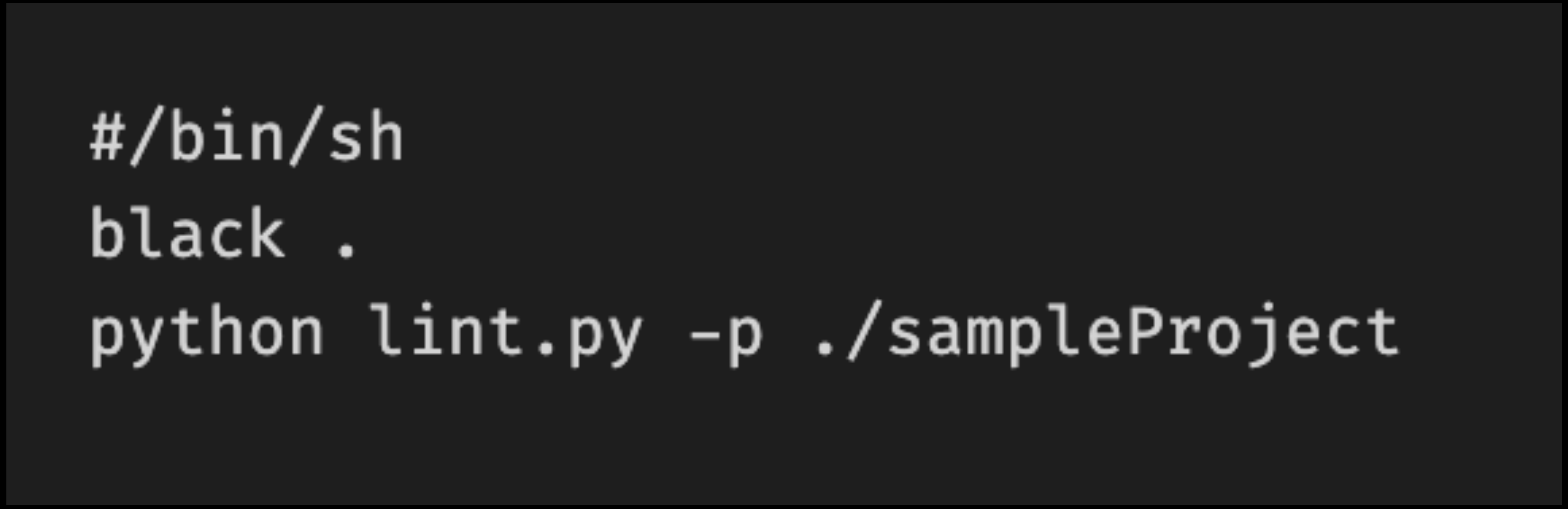
Use pre-commit hooks

And encourage others to do the same

```
#!/bin/bash
#
# Install a git pre-commit hook to format and lint Python files
#
# See https://rikwatson.github.io/python_lint for more details
#
echo $'#/bin/sh\nblack .\npython lint.py -p ./sampleProject'> .git/hooks/pre-commit

chmod +x .git/hooks/pre-commit
```

•



```
#/bin/sh
black .
python lint.py -p ./sampleProject
```

Azure Pipelines

Key Concepts

- A trigger tells a Pipeline to run.
- A pipeline is made up of one or more stages. A pipeline can deploy to one or more environments.
- A stage is a way of organising jobs in a pipeline and each stage can have one or more jobs.
- Each job runs on one agent. A job can also be agentless.
- Each agent runs a job that contains one or more steps.
- A step can be a task or script and is the smallest building block of a pipeline.
- A task is a pre-packaged script that performs an action, such as invoking a REST API or publishing a build artifact.
- An artifact is a collection of files or packages published by a run.

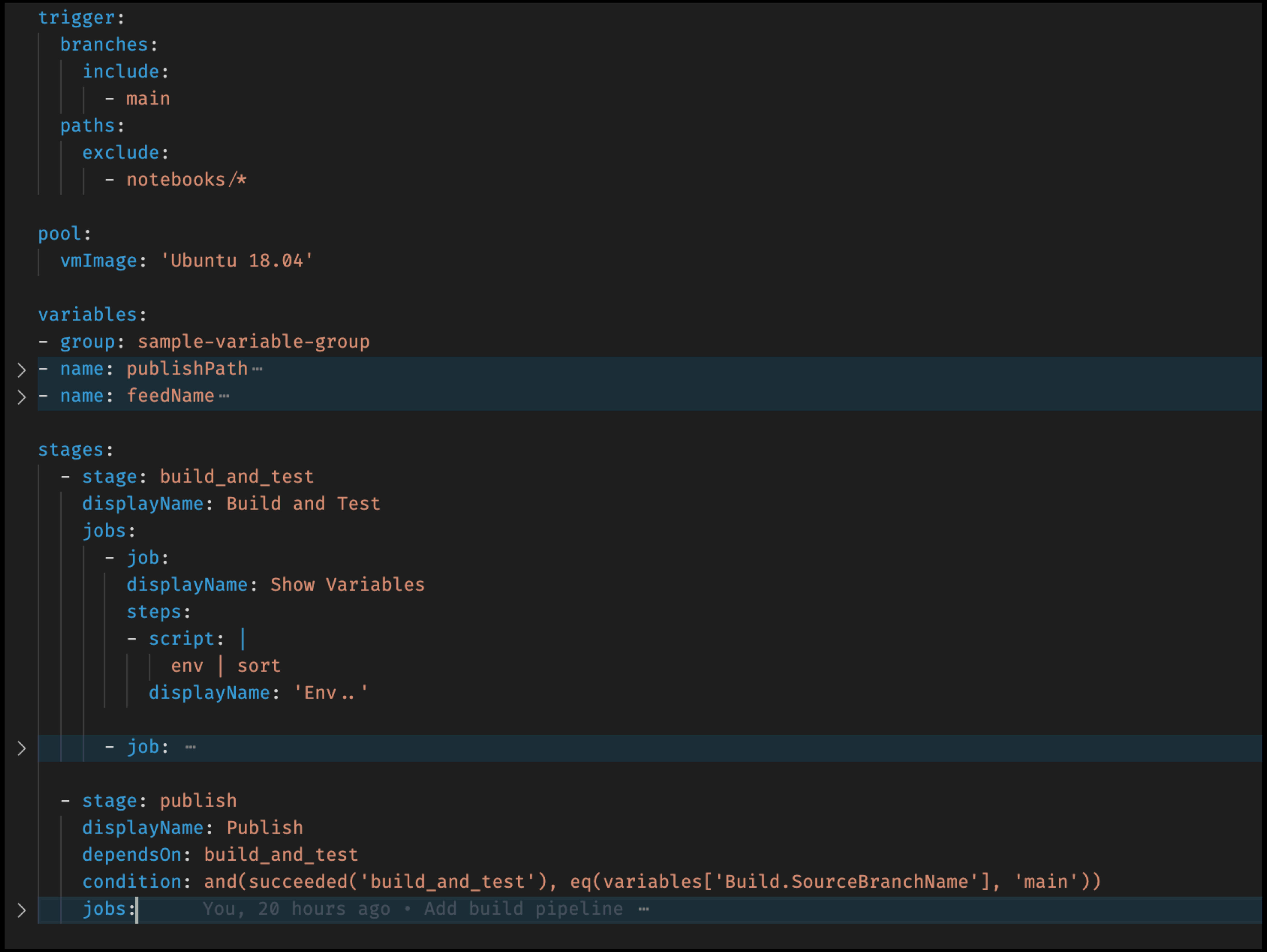
Pipeline as Code

YAML - YAML Ain't Markup Language

- Having the full CI/CD pipeline expressed as code has significant advantages
- Can be held in a DVCS
- `git blame` is your friend
- Spend time learning YAML and get a schema validator (yamale?)
    - Azure DevOps schema is here:
        - https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema

Azure Pipeline

Overview of structure



```yaml
trigger:
  branches:
    include:
      - main
  paths:
    exclude:
      - notebooks/*

pool:
  vmImage: 'Ubuntu 18.04'

variables:
- group: sample-variable-group
- name: publishPath⋯
- name: feedName⋯

stages:
  - stage: build_and_test
    displayName: Build and Test
    jobs:
      - job:
        displayName: Show Variables
        steps:
        - script: |
            env | sort
          displayName: 'Env..'

      - job: ⋯

  - stage: publish
    displayName: Publish
    dependsOn: build_and_test
    condition: and(succeeded('build_and_test'), eq(variables['Build.SourceBranchName'], 'main'))
    jobs:      You, 20 hours ago • Add build pipeline ⋯
```

Install dependencies

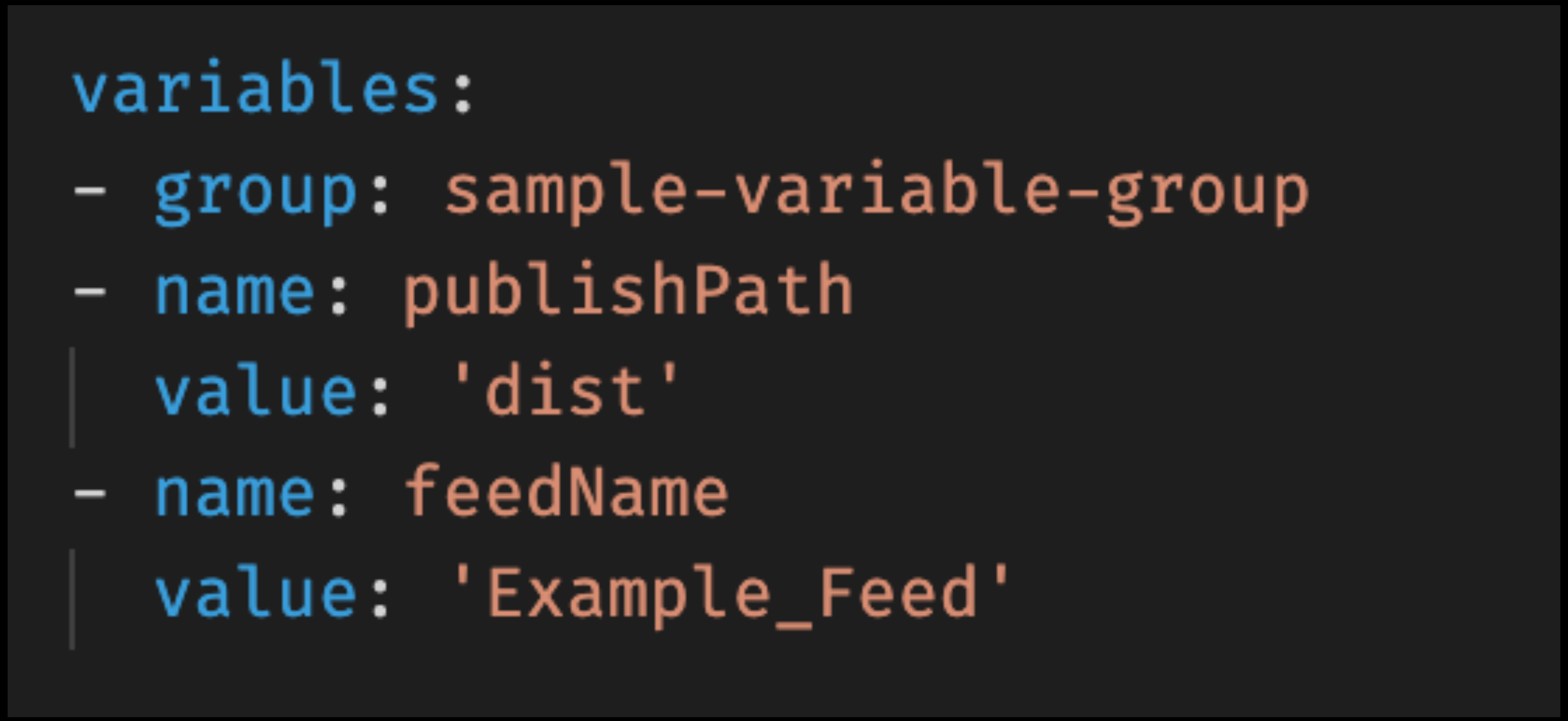
Should reflect your README

```
        - script: |
            python -m pip install -U pip
            python -m pip install 'pytest==6.2.4' 'poetry==1.1.7'
            poetry install --no-interaction
          displayName: 'Install dependencies'
```

•

Azure Pipeline

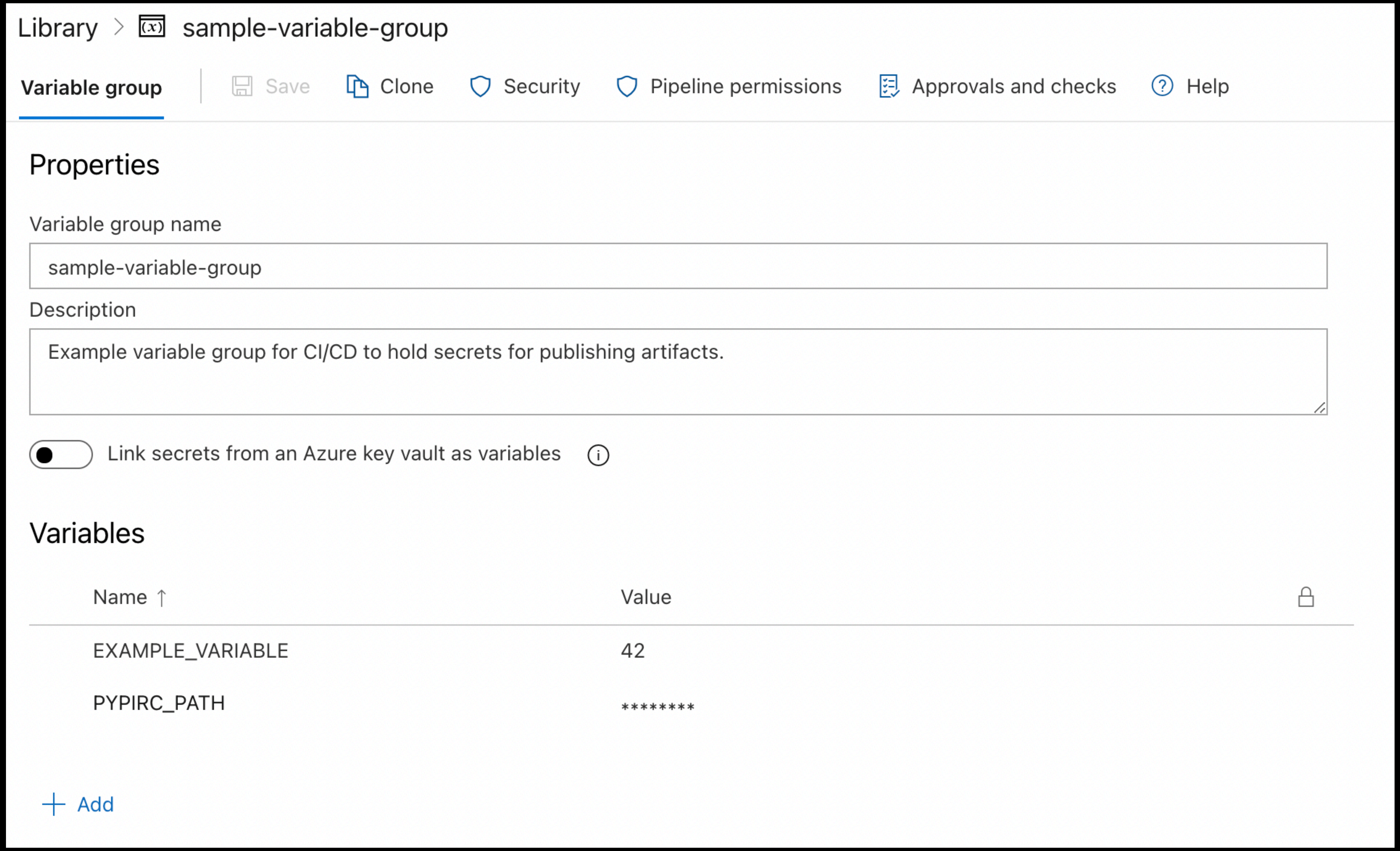
Variables



```yaml
variables:
- group: sample-variable-group
- name: publishPath
  value: 'dist'
- name: feedName
  value: 'Example_Feed'
```

Azure Pipeline

Variables



Properties

Variable group name: sample-variable-group

Description: Example variable group for CI/CD to hold secrets for publishing artifacts.

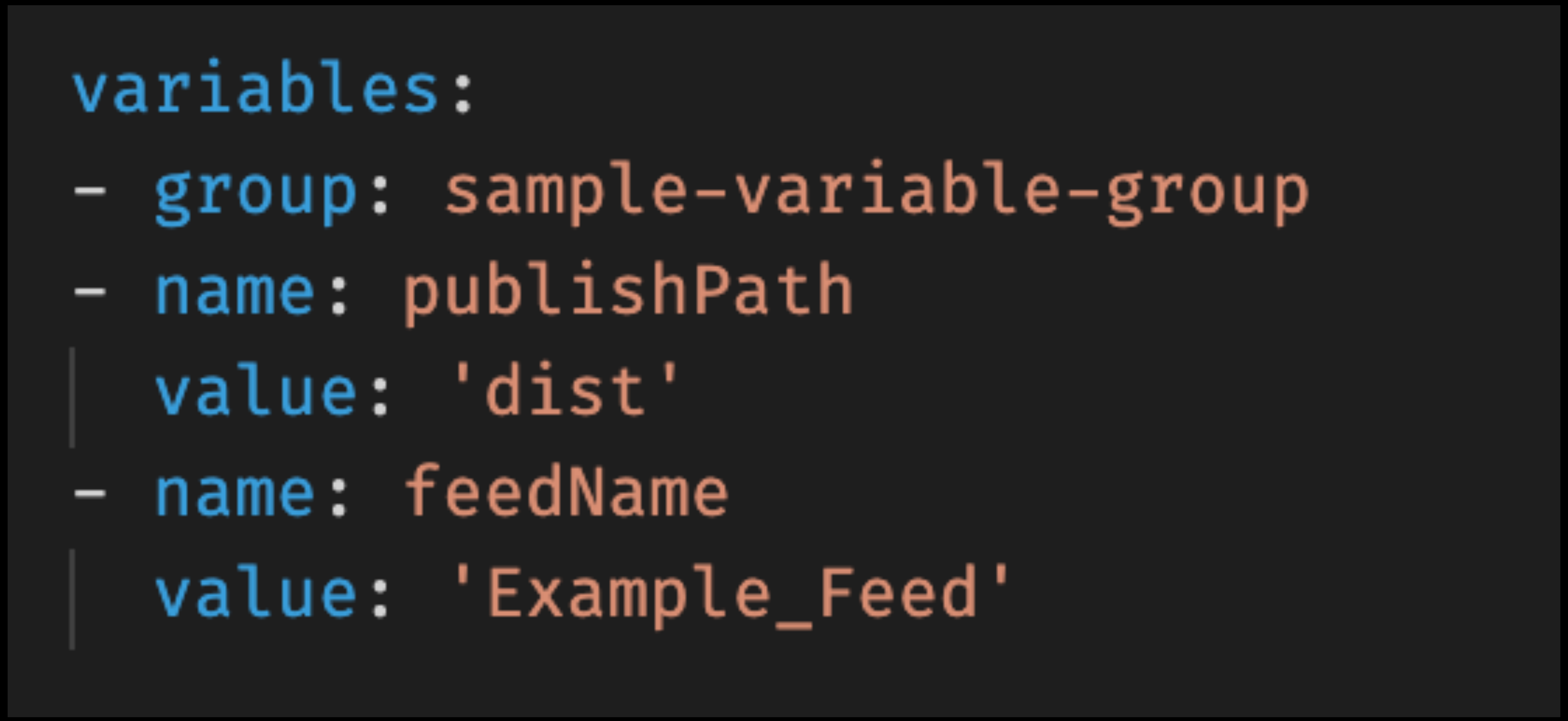
Link secrets from an Azure key vault as variables

Variables

Name ↑	Value
EXAMPLE_VARIABLE	42
PYPIRC_PATH	********

Azure Pipeline

Stages



```yaml
variables:
- group: sample-variable-group
- name: publishPath
  value: 'dist'
- name: feedName
  value: 'Example_Feed'
```

Matrix Builds

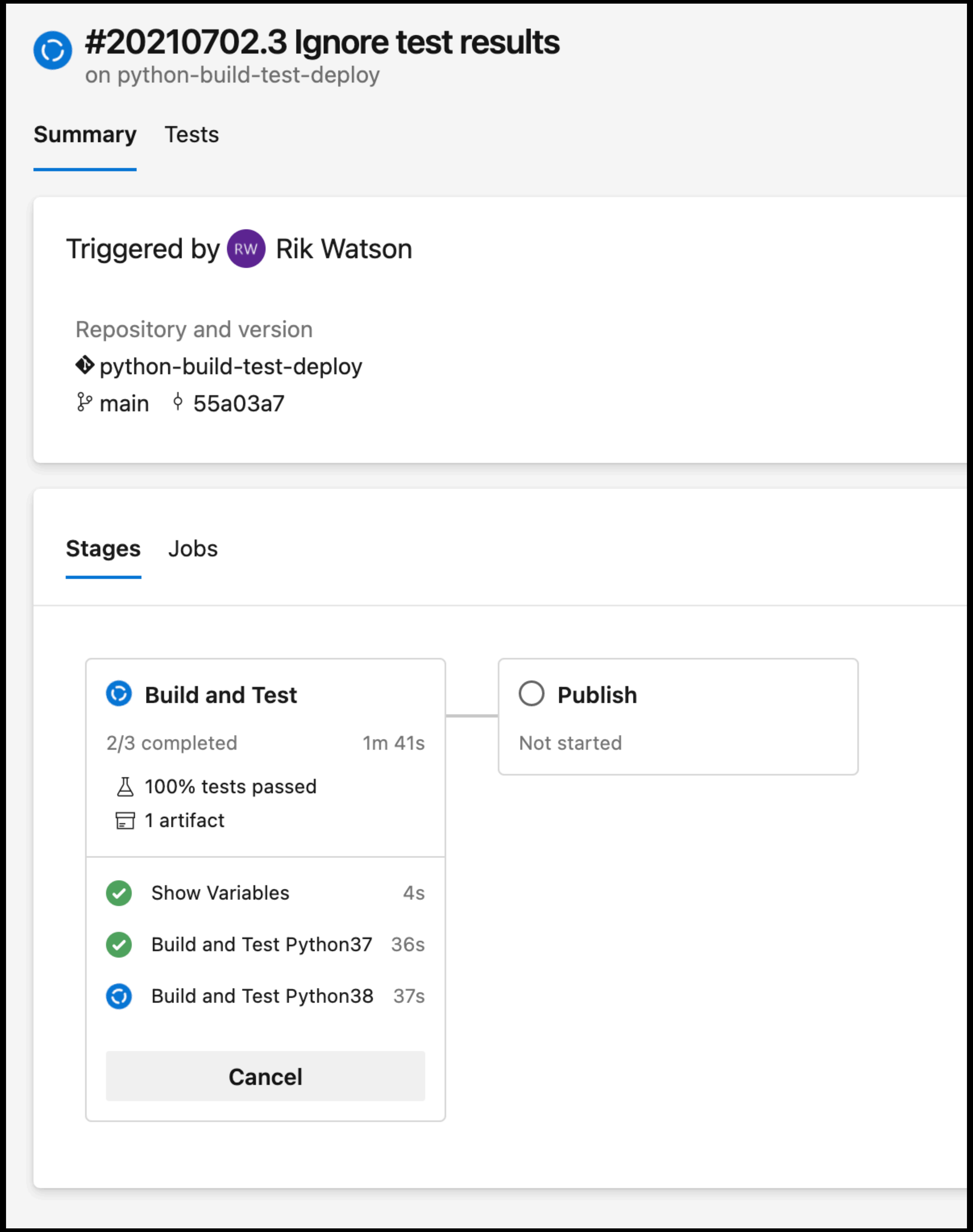
Nero would be proud

```
    - job:
      displayName: Build and Test
      strategy:
        matrix:
          Python37:
            python.version: '3.8'
          Python38:
            python.version: '3.9'
        maxParallel: 4
      steps:…
```

.

Azure Pipeline

Building



#20210702.3 Ignore test results
on python-build-test-deploy

Summary Tests

Triggered by RW Rik Watson

Repository and version
python-build-test-deploy
main 55a03a7

Stages Jobs

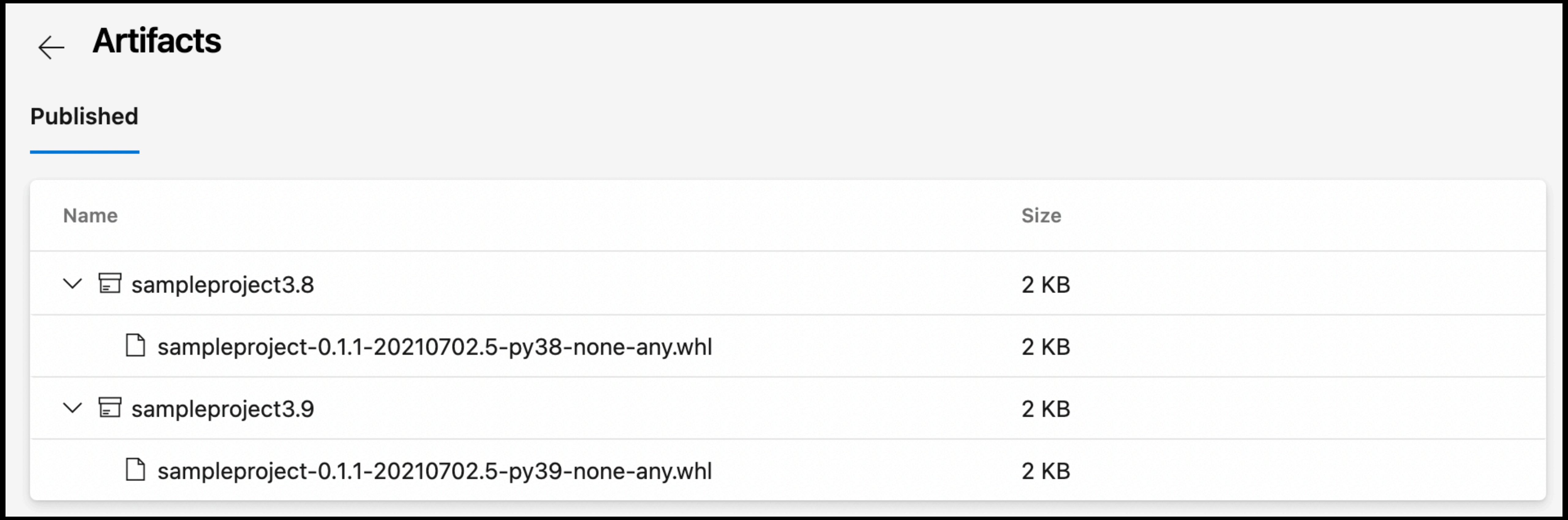
Build and Test
2/3 completed 1m 41s
100% tests passed
1 artifact

Show Variables 4s
Build and Test Python37 36s
Build and Test Python38 37s

Cancel

Publish
Not started

Azure Pipeline
Publishing artifacts locally



Artifacts

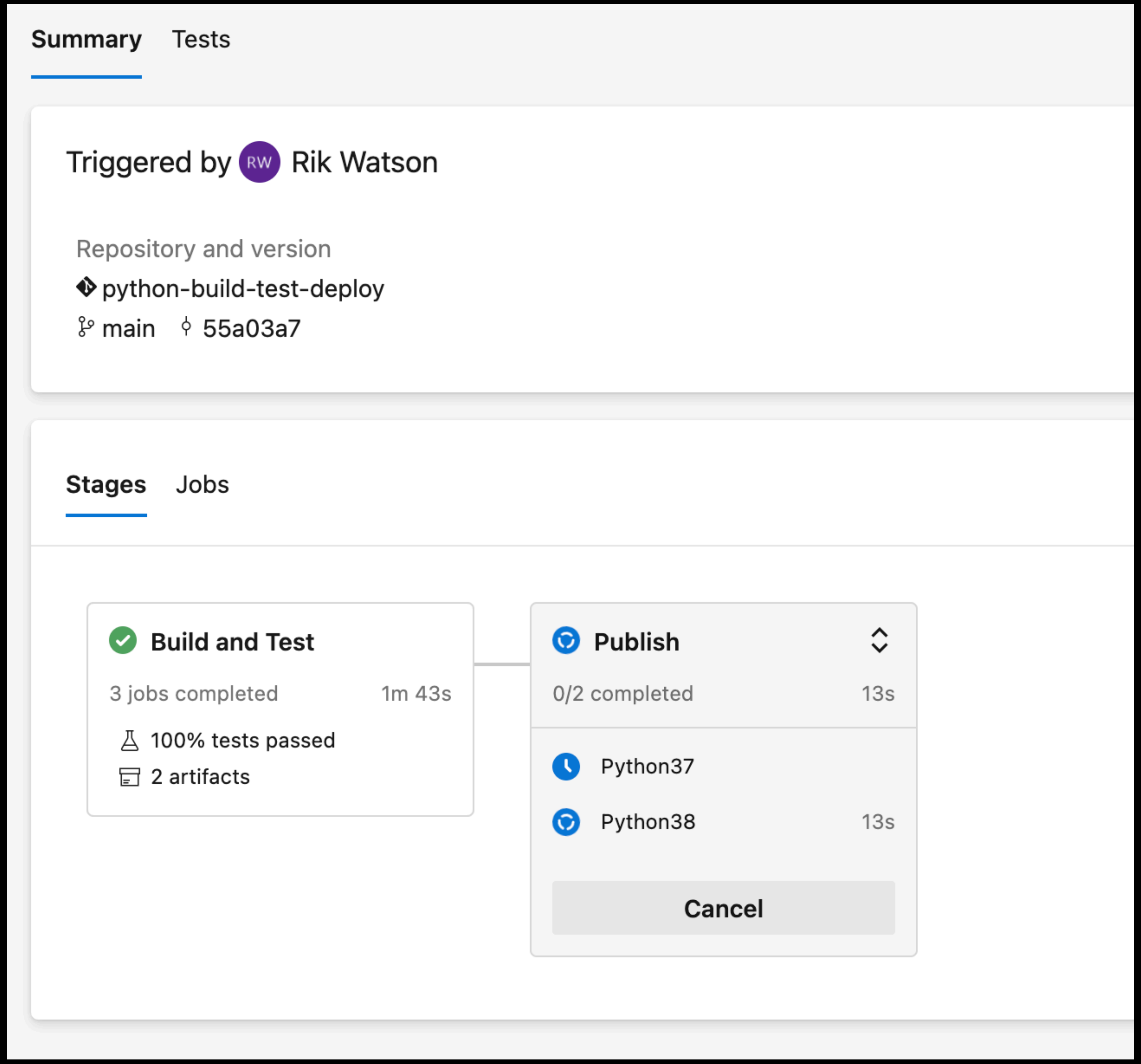
Published

Name	Size
sampleproject3.8	2 KB
sampleproject-0.1.1-20210702.5-py38-none-any.whl	2 KB
sampleproject3.9	2 KB
sampleproject-0.1.1-20210702.5-py39-none-any.whl	2 KB

Note: SemVer & Python version

Azure Pipeline

Publishing to an external feed



Summary Tests

Triggered by RW Rik Watson

Repository and version
python-build-test-deploy
main 55a03a7

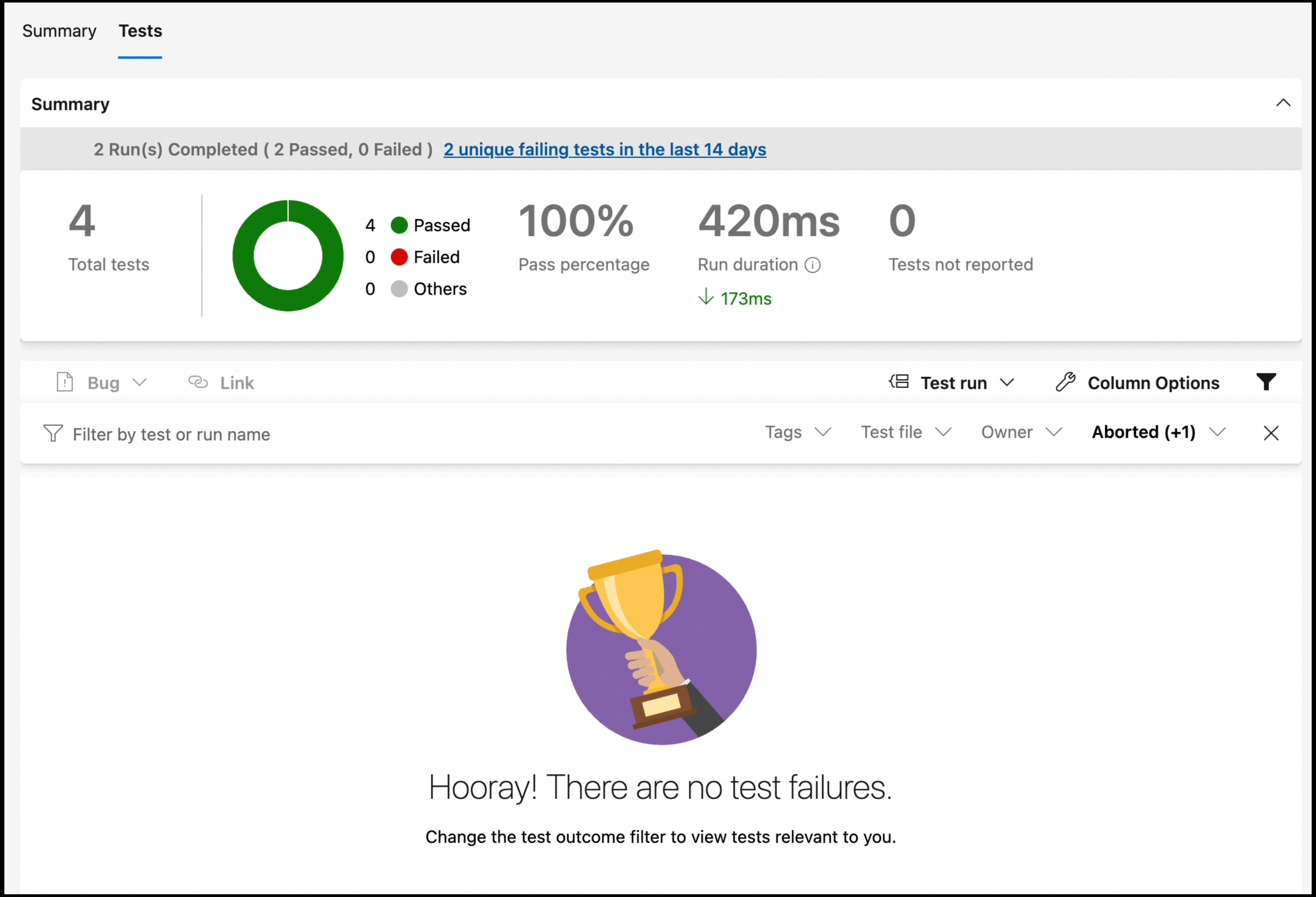
Stages Jobs

Build and Test
3 jobs completed 1m 43s
100% tests passed
2 artifacts

Publish
0/2 completed 13s
Python37
Python38 13s
Cancel

Azure Pipeline

Test results


Summary Tests

Summary

2 Run(s) Completed (2 Passed, 0 Failed) 2 unique failing tests in the last 14 days

4 Total tests

4	● Passed
0	● Failed
0	● Others

100% Pass percentage

420ms Run duration ↓ 173ms

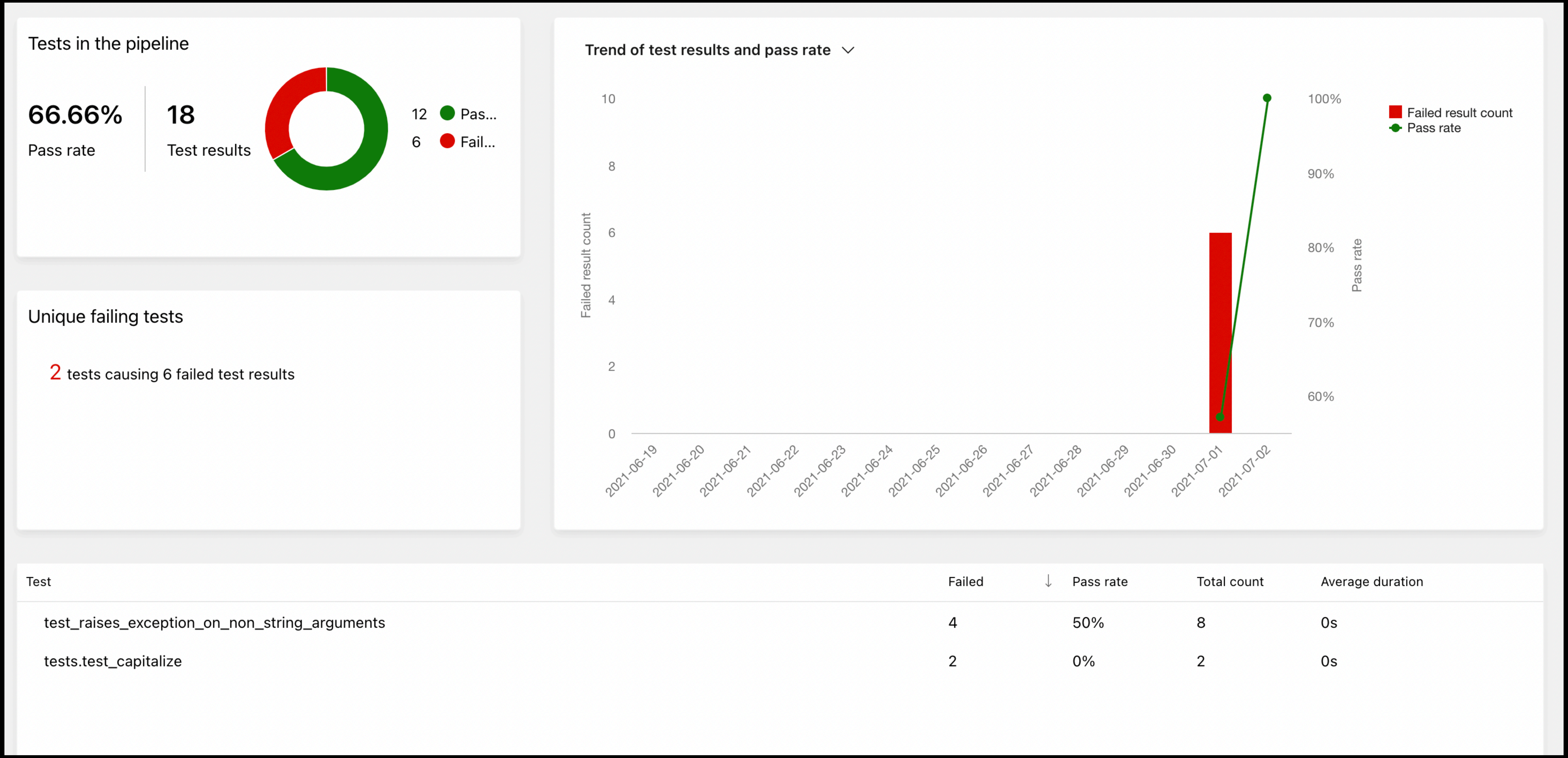
0 Tests not reported

Hooray! There are no test failures.

Change the test outcome filter to view tests relevant to you.

Azure Pipeline

Test results - details of previous tests



Tests in the pipeline

66.66%
Pass rate

18
Test results

12 ● Pas...
6 ● Fail...

Unique failing tests

2 tests causing 6 failed test results

Trend of test results and pass rate

Test	Failed ↓	Pass rate	Total count	Average duration
test_raises_exception_on_non_string_arguments	4	50%	8	0s
tests.test_capitalize	2	0%	2	0s

SemVer

Semantic Versioning - used in package management

- Given a version number MAJOR.MINOR.PATCH, increment the:
    - MAJOR version when you make incompatible API changes,
    - MINOR version when you add functionality in a backwards compatible manner, and
    - PATCH version when you make backwards compatible bug fixes.

    - https://rikwatson.github.io/semver

Extending Azure Pipelines

Tasks & Templates

- Azure provides two distinct methods for extending Azure Pipelines
- Templates: Just YAML text files. Simple parameterised text substitution
- Tasks: Written in TypeScript, have the full power of the Azure API

Azure Pipeline Templates

Basic Example

```
# File: simple-param.yml
parameters:
- name: yesNo # name of the parameter; required
  type: boolean # data type of the parameter; required
  default: false

steps:
- script: echo ${{ parameters.yesNo }}
```

Then within your azure-pipelines.yml

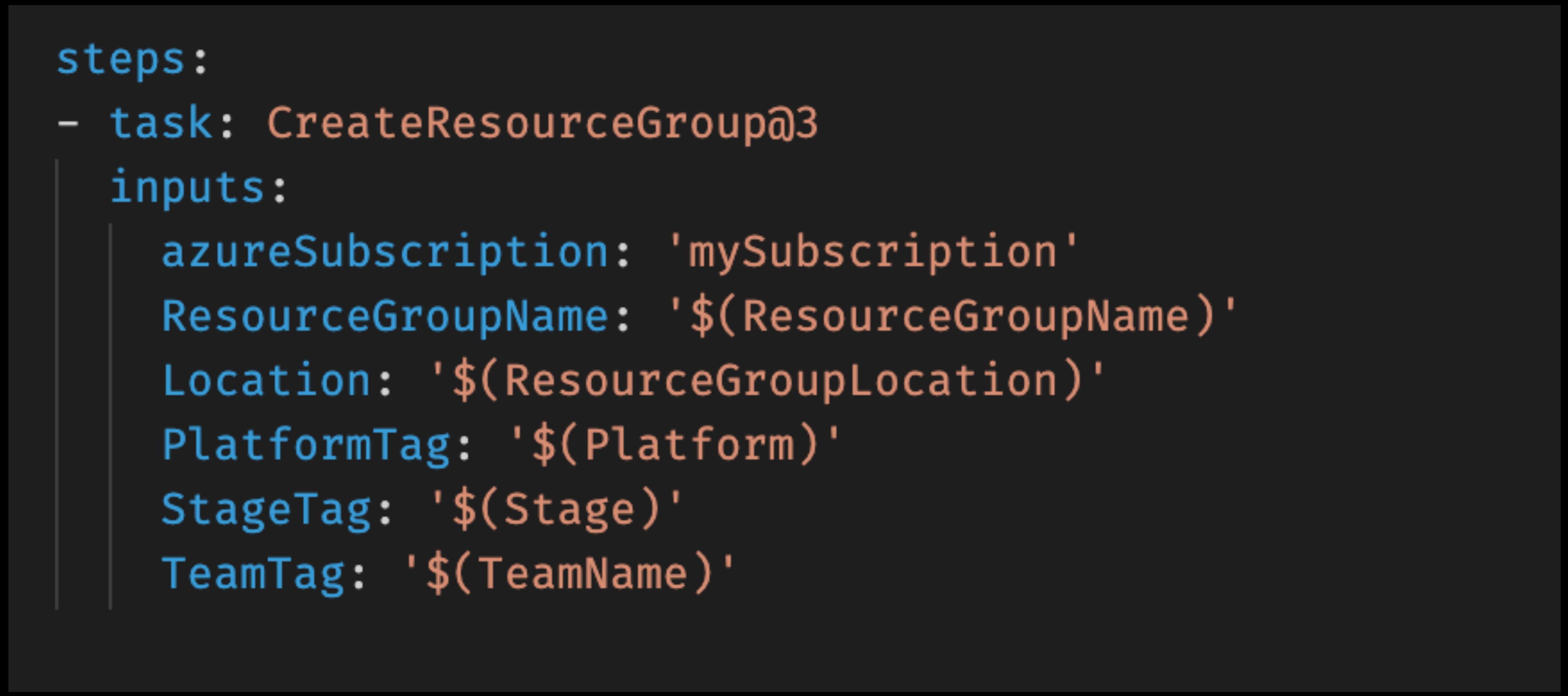
```
extends:
  template: simple-param.yml
  parameters:
      yesNo: false # set to a non-boolean value to have the build fail
```

https://docs.microsoft.com/en-us/azure/devops/pipelines/process/templates

Azure Pipeline Tasks

Basic Usage

- Written in TypeScript
- An example would be outside the scope of this presentation
- However they ‘look’ just like built-in tasks:



```yaml
steps:
- task: CreateResourceGroup@3
  inputs:
    azureSubscription: 'mySubscription'
    ResourceGroupName: '$(ResourceGroupName)'
    Location: '$(ResourceGroupLocation)'
    PlatformTag: '$(Platform)'
    StageTag: '$(Stage)'
    TeamTag: '$(TeamName)'
```

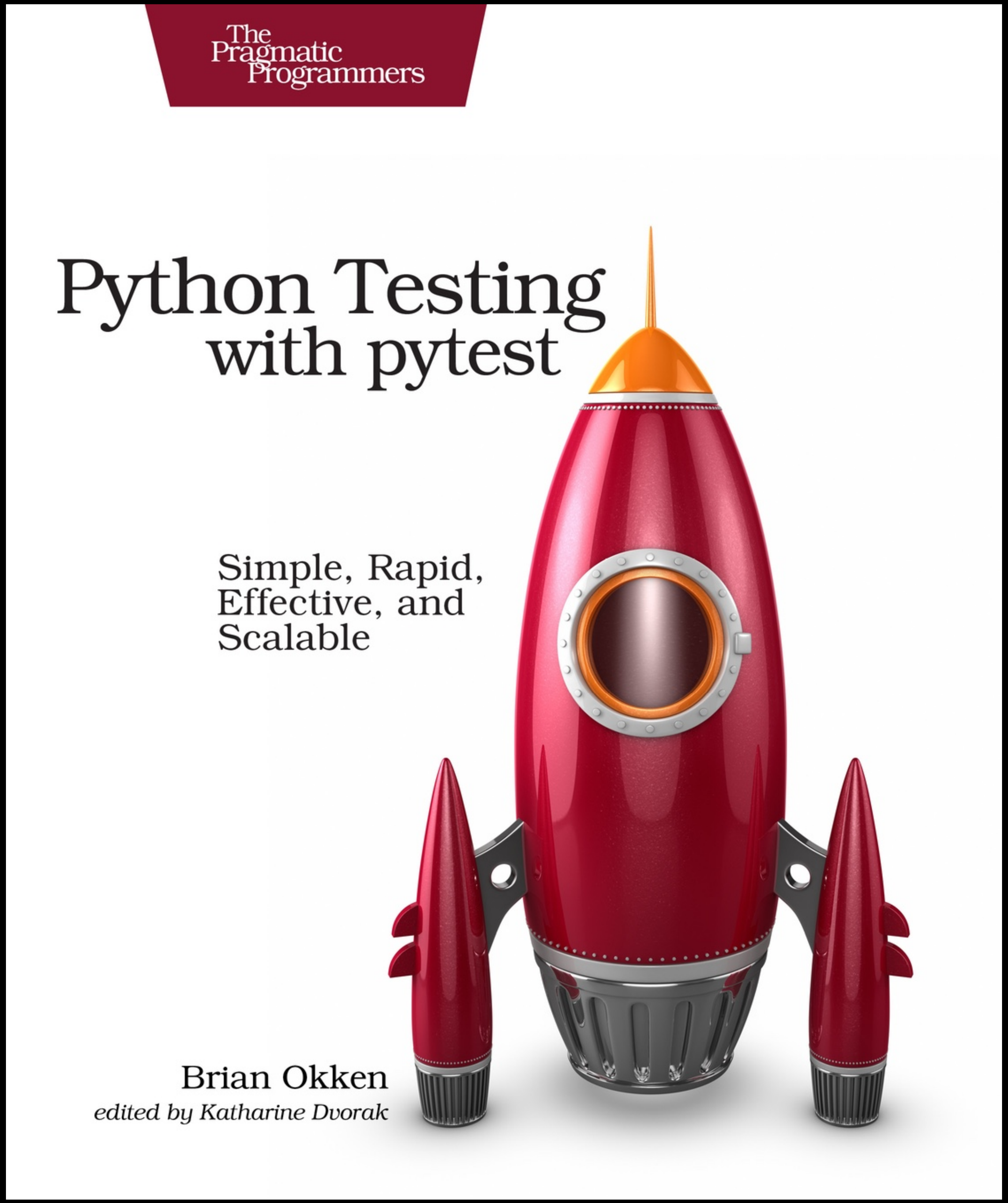
Further Reading

Pipelines & Tool chain

- Black - https://rikwatson.github.io/python_black
- YAML - https://rikwatson.github.io/yaml
- pylint - https://rikwatson.github.io/python_pylint
- Linting - https://rikwatson.github.io/python_lint
- Pipelines - https://rikwatson.github.io/azure_devops_pipelines

Further Reading

Python Testing with pytest - Brian Okken


The Pragmatic Programmers
Python Testing with pytest
Simple, Rapid, Effective, and Scalable
Brian Okken
edited by Katharine Dvorak

Follow on exercises

Should only take 30 min’s or so each

- Modify the azure-pipelines.yml to publish to a PyPI repository
- Enable the git hooks and understand how they can be used to slowly improve code quality
  Remember the hooks only run on files that have been modified
  - if you change your rules for Black they will only be reflected on changed files
  - how do you fix this
- BONUS: Add a docker compose step to show how tests could interact with a SQL db (for example)

Where next ?

- The project & repository we’ve discussed are world readable
  https://dev.azure.com/RikWatson0604/python-build-test-deploy
- The most important files are README.md and azure-pipelines.yml
  If you do nothing else then Grok these

Black

An opinionated code formatter

- Tabs or spaces - yes it matters
- Having a common code format has been show to greatly reduce cognitive load when coming into a new project
- If you *REALLY* want you could use IDE extensions (or git hooks) to format code to your way of working when they are in your working area