GitHub Actions:

Isaac Davies

Housekeeping

- Make sure you have a GitHub account
- Open the github.com/LungaFermata/Actions repository
- If you feel lost at any point let me know

What are Actions

- Designed as a continuous integration and continuous delivery (CI/CD) platform
- Allow for tasks to be automatically started and run on cloud hardware
- Workflow files define these tasks

Workflows

- A workflow is a YAML file defined in .github/workflows
- The file defines:
    - The name of the workflow
    - What events trigger it
    - What steps will be taken when triggered
- A workflow can be broken up into multiple jobs
- Workflow files only function when they are part of the default branch

Runs/Runners

- A run is started when the workflow is triggered
- Each job in a workflow is run on its own VM
- The operating system and architecture is defined on a per job basis
- Standard images have Ubuntu, Windows and macOS
- We’re going to stick to ubuntu-latest

YAML Ain’t Markup Language

- Yet Another Markup Language
- Human readable data serialisation language like XML or JSON
- YAML supports two distinct syntax styles
    - Block style is indentation based (tab isn’t allowed)
    - Flow style uses {braces} and [brackets]
- 3 Basic Data types – we don’t need the rest
- We’re only covering the basics

Scalars

- Stores Atomic data like strings or null
  - hello, 83, true, null
- YAML doesn’t process data and so doesn’t need the exact type
- They are the only non-structural data type
- formatting.yaml expands on scalar formats

Mappings

- Mappings store a key: value pair
- Key is identified with a colon, must be followed by a space
- They can be nested through indentation
- Nesting can be arbitrarily deep
- Multiple maps can be nested on the same level

```
name: CI
on:
  push:
    branches: main
```

Sequence

- A list or indexed array of values
- Can also be arbitrarily deep
- Mappings and sequences can be intermixed as necessary
- Indentation is still mandatory, but each dash is treated as an indent

```
- 1
- - 2
  - 3
- 4

Names:
  - Monica
  - Robert
```

Flow Style Collections

- Block style collections require every entry to be on their own line
- Flow style allows entries to be compressed while maintaining structure
  - {braces} are used to indicate mappings
  - [brackets] indicate sequences
  - Values on the same level are separated by commas
- Flow style can be nested inside Block style but not the inverse

```
Names:
 - Monica
 - Robert
```

```
{ Names: [ Monica, Robert ] }

Names: [ Monica, Robert ]
```

A Basic Workflow

```
name: CI
on: [push]
jobs:
  Hello:
    runs-on: ubuntu-latest
    steps:
      - name: Print Hello
        run: echo hello
```

Workflow structure

name: The workflow will be named “CI” in the action tab of the repo

on: When a commit is pushed, the workflow will trigger

jobs: maps to each job to be performed

```
name: CI

on: [push]

jobs:
```

Nested jobs: mapping

A single job named “Hello”

runs-on: requests the specified VM image

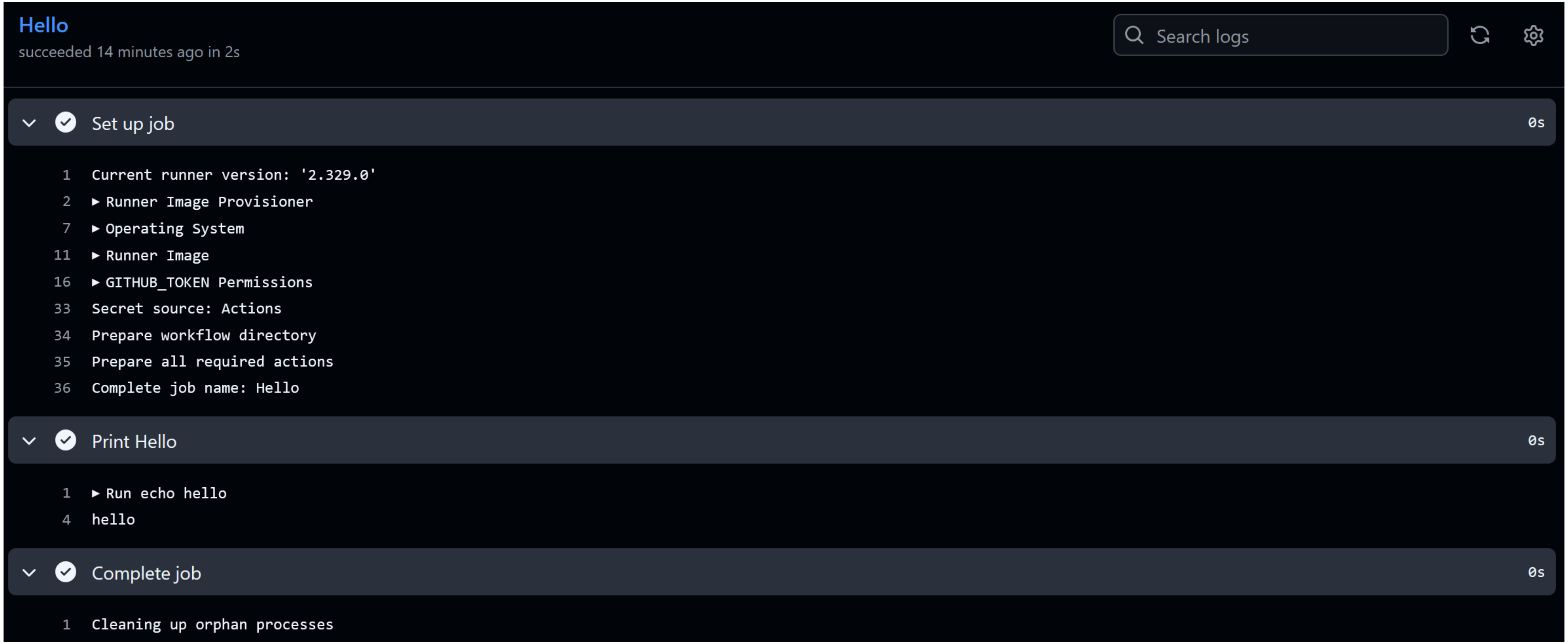
steps: maps to each step

- name: the name of that step

run: the code to be run on the terminal in this step

```
jobs:
  Hello:
    runs-on: ubuntu-latest
    steps:
      - name: Print Hello
        run: echo hello
```

Example Output Log



Hello

succeeded 14 minutes ago in 2s

Set up job 0s

```
Current runner version: '2.329.0'
▶ Runner Image Provisioner
▶ Operating System
▶ Runner Image
▶ GITHUB_TOKEN Permissions
Secret source: Actions
Prepare workflow directory
Prepare all required actions
Complete job name: Hello
```

Print Hello 0s

```
▶ Run echo hello
hello
```

Complete job 0s

```
Cleaning up orphan processes
```

Actions within Actions

- GitHub allows actions to be published and called by other actions
- These serve as publicly maintained functions
- They are called as part of your steps
- Actions can pose an upstream security risk but some are necessary

```
name: step_name

uses: actions/checkout@v5
```

Contexts and substitution

- Contexts are variables which are substituted when the workflow is called
- GitHub uses the syntax ${{ }} to identify a substitution
- These context allow the workflow to access information such as who triggered it, ${{ github.actor }}, and what event it was, ${{ github.event_name }}.
- This substitution is done when the workflow is called and is vulnerable to scrip injection

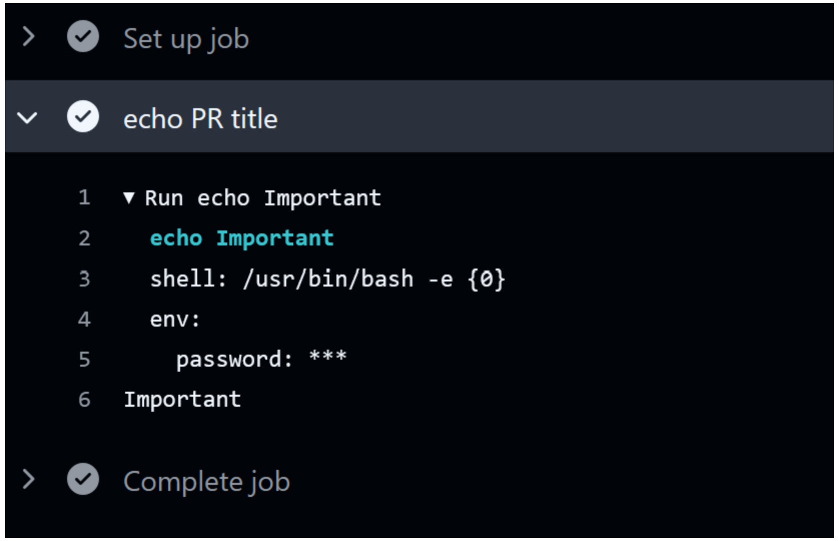
if Statements

- Both jobs and steps support if statements
- This is done by nesting if: under them, and if the content of the mapping evaluates to true the job or step is performed.

```
jobs:
  die:
    if: ${{ github.actor != LungaFermata }}
```

Secrets

- GitHub secrets are user created contexts which can be used in workflows.
- These secrets are automatically redacted in the logs
- This redaction can be easily defeated if used carelessly



```
▼ Run echo Important
  echo Important
  shell: /usr/bin/bash -e {0}
  env:
    password: ***
Important
```