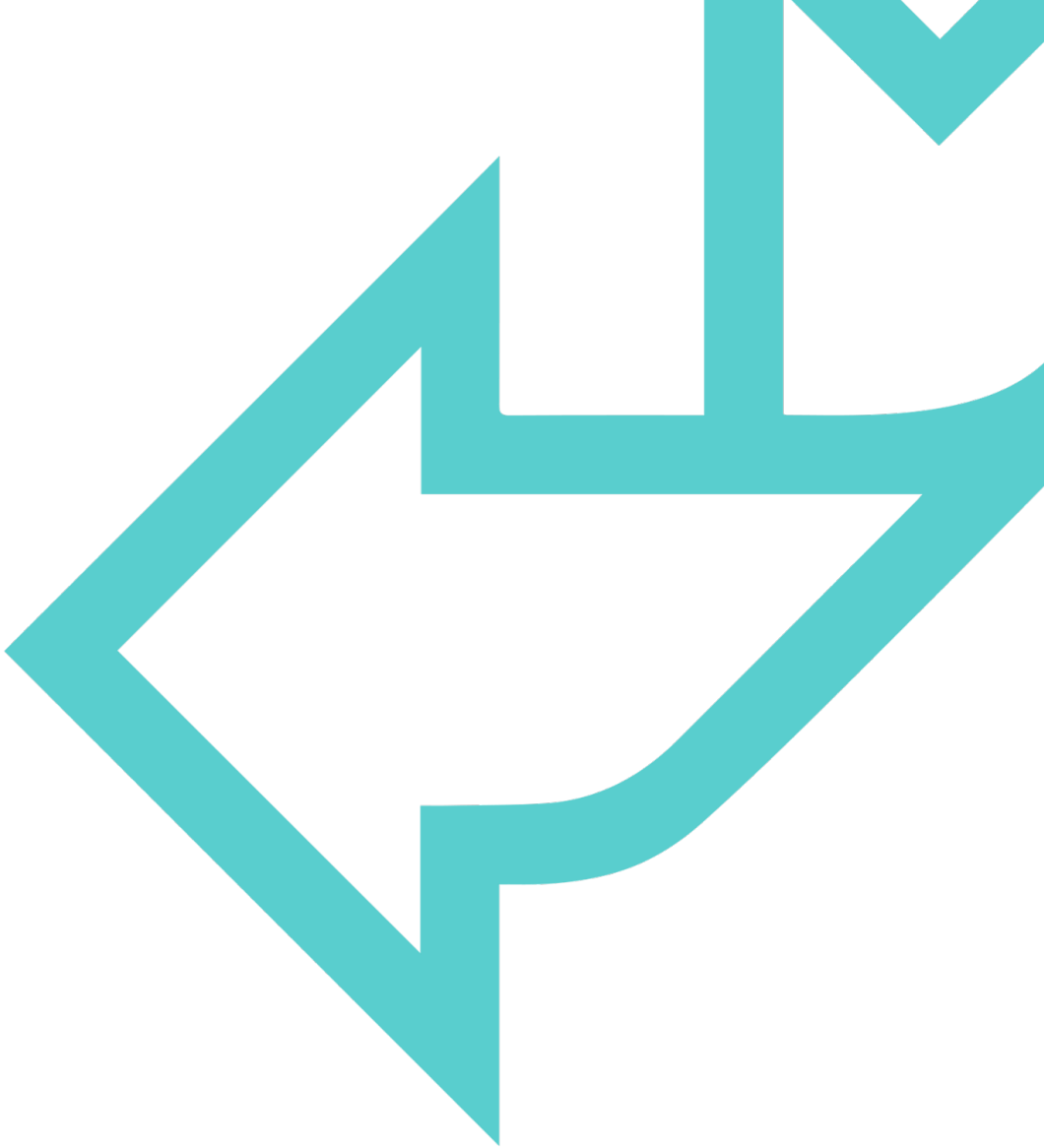
POWERING POTENTIAL



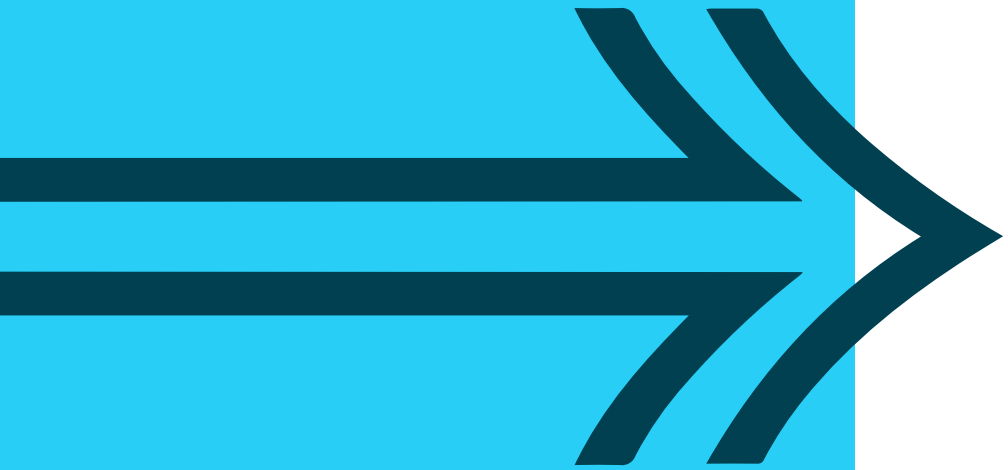
Introduction to GitHub Actions

Module 5 – Deploying to the Cloud

RECAP – CI/CD

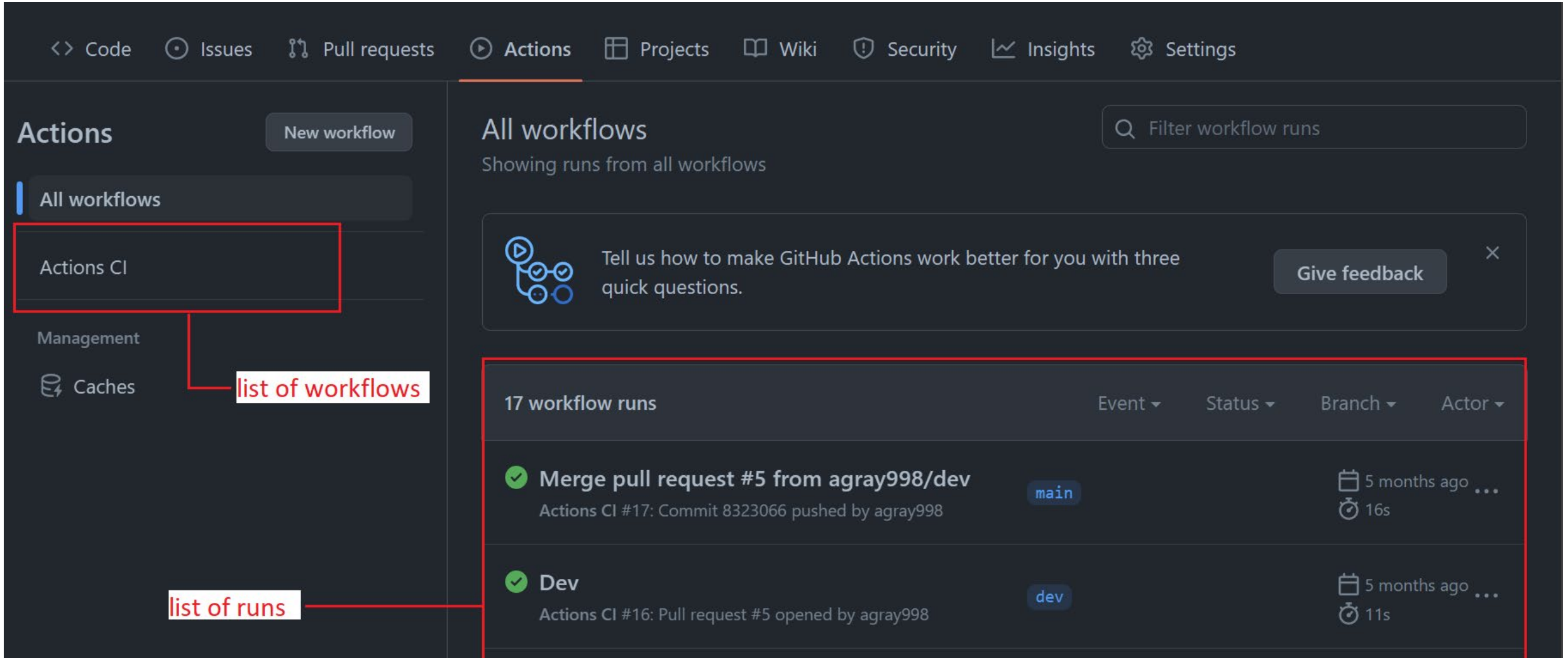
- Continuous integration (CI) and continuous delivery /deployment (CD) are methods for automating the processes involved in testing and deploying new code
- Performing these tasks automatically as opposed to manually saves a lot of time and allows for quicker deployments and more stable builds
- CI/CD has become so important that a wide range of tools exist to facilitate these automated workflows
- Many repository-hosting services and cloud providers offer their own suites of CI/CD tools
- One such tool offered by GitHub is **GitHub Actions**

ESSENTIAL CONCEPTS

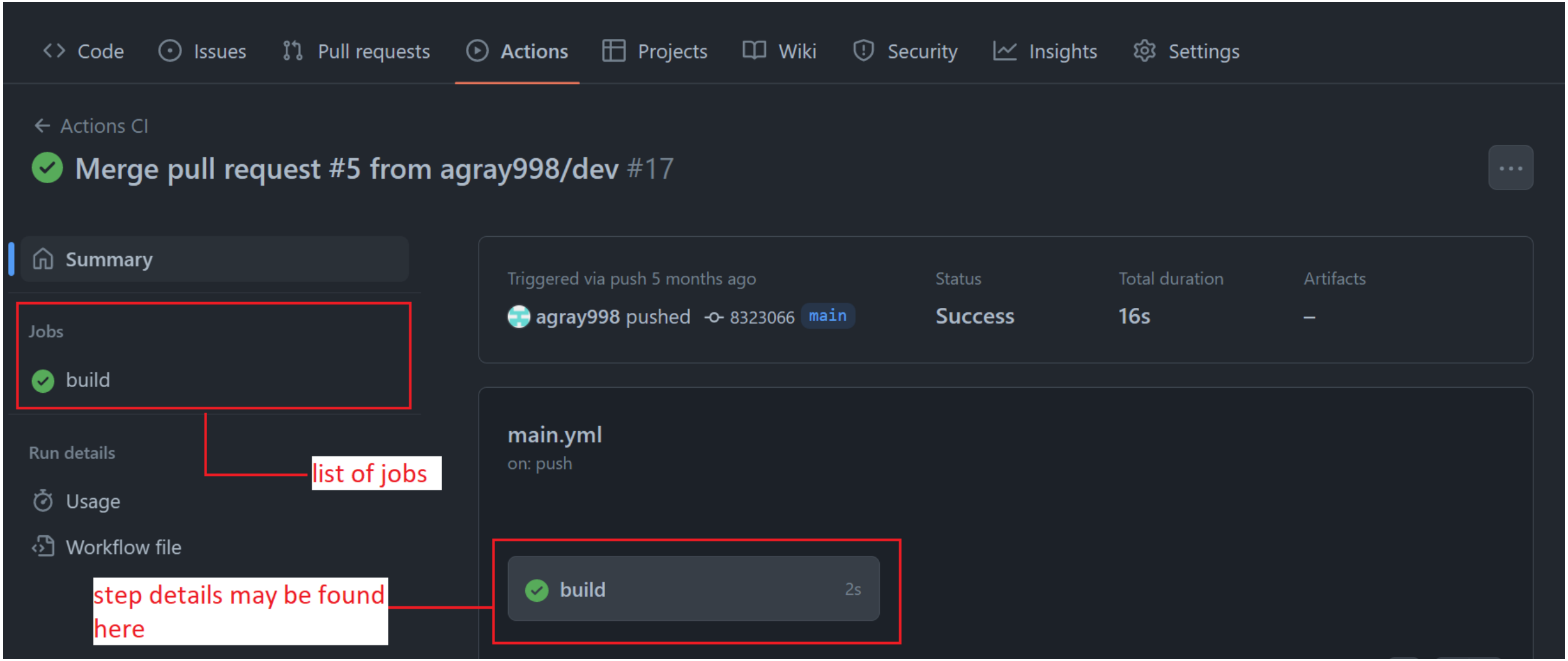


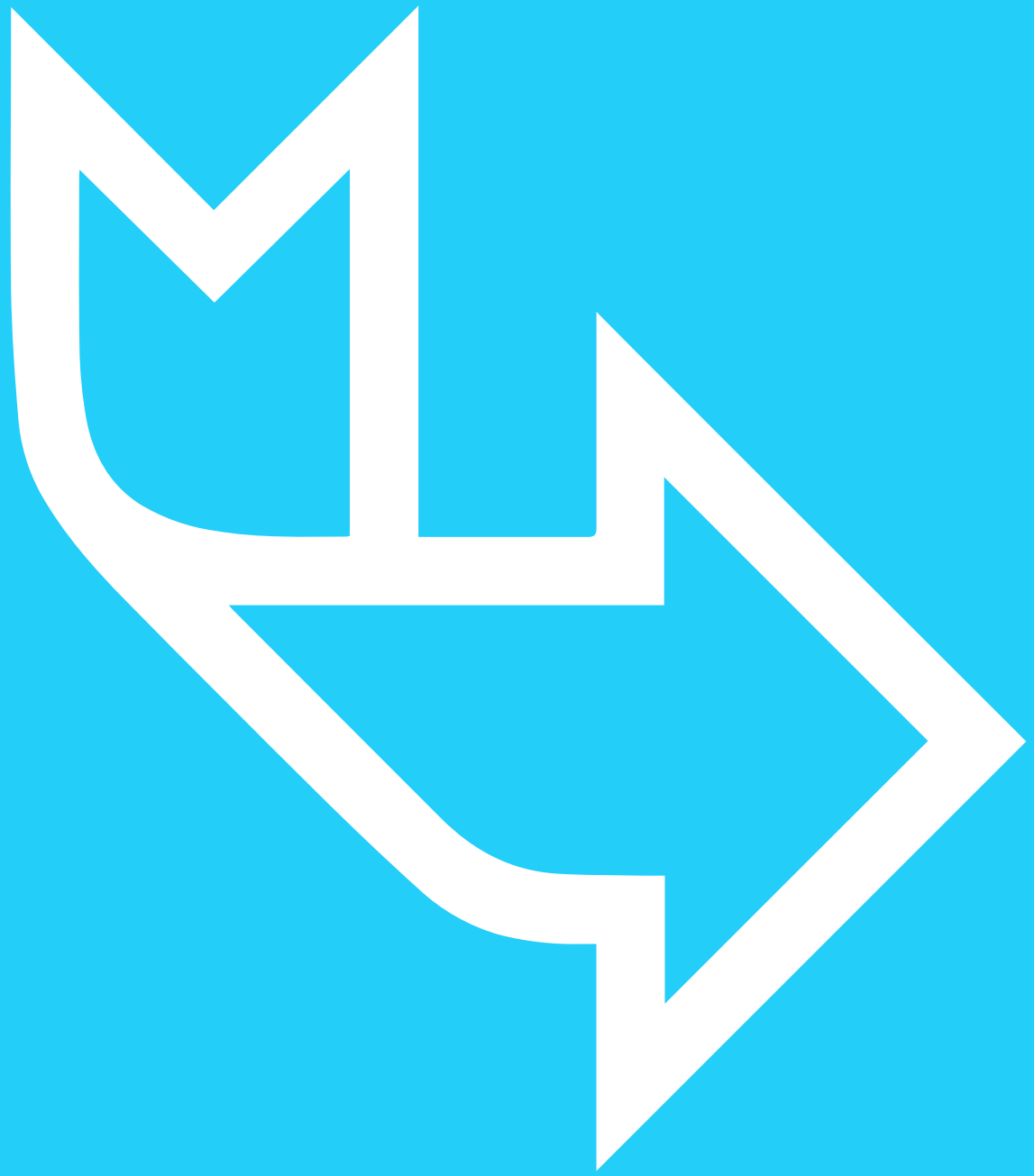
- GitHub Actions may be enabled on any GitHub repository in order to implement CI/CD for that project
- When Actions is enabled on a repository it will, on pre-defined triggers, execute the steps laid out in one or more **workflows**
- these are defined in a directory within the repository called .github/workflows
- A workflow can be broken down further into **jobs** and then **steps**
- Each time a workflow is triggered, the resulting execution is called a **run**
- Actions executes the tasks making up the workflow in **containers** – each job gets its' own container which defines the **build environment**

UI example – workflows



UI example – jobs



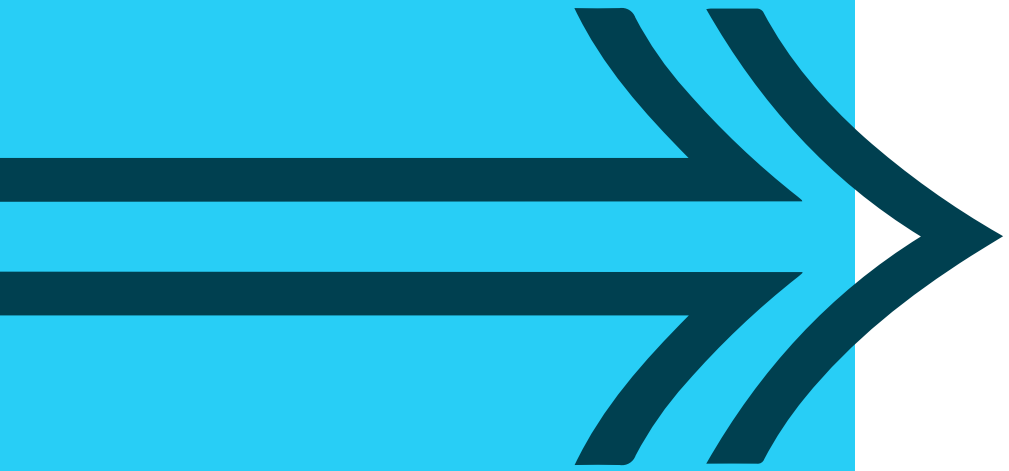


Discussion: why GitHub actions?

In breakout groups, think about how GitHub Actions compares to other common CI/CD tools – e.g., Jenkins, TeamCity, CircleCI – and consider the following questions:

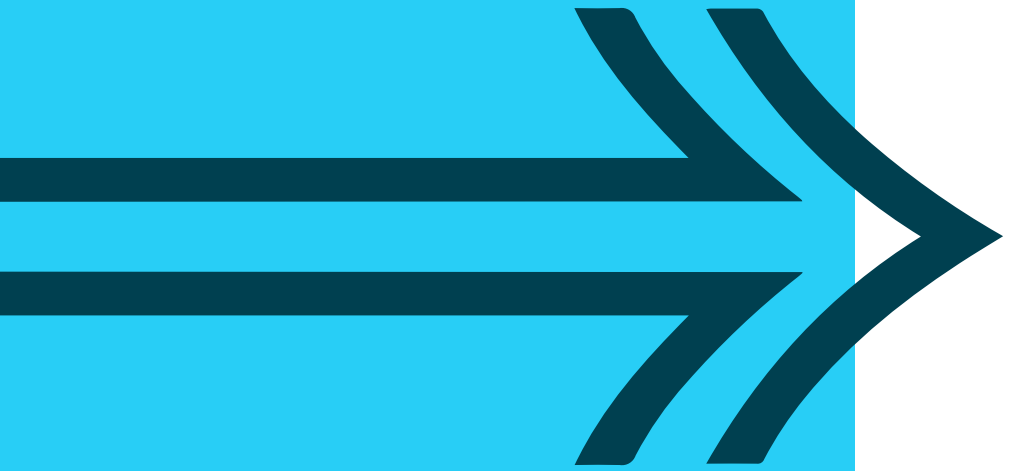
- Do you notice any common patterns in how these tools are used?
- What might be the pros and cons of GitHub actions, in comparison with these other tools?

WHY GITHUB ACTIONS?



- Broadly speaking, CI/CD tools may be self-hosted (e.g., Jenkins server, TeamCity server) or managed (e.g., GitHub Actions, CircleCI)
- Managed CI/CD tools offer a cost reduction – you do not need to incur the costs of maintaining a server, and are typically only billed for the time that your jobs are running
- Managed CI/CD tools are also highly scalable – it is easy to spin up more containers to handle increased workloads
- Self-hosted servers can offer more customisability, useful if a somewhat niche configuration is required for your build environment
- Compared to other managed services, GitHub Actions offers extensive integration with other tools and services, including major cloud platforms
- GitHub automatically runs workflows defined in GitHub repositories – if your repository is not hosted on GitHub then you may want to use an alternative service

WORKFLOW STRUCTURE



- Workflows are specified using a syntax called YAML – if you're not familiar with YAML think of it as JSON without the curly braces
- The workflow begins with a name, which sets the name of the workflow as it will appear when viewing the runs for that workflow
- The on section sets the triggers for the workflow – these are grouped by event type, with a list of branches for which each event should trigger a run
- The jobs section contains one or more jobs, each of which should define the runner which will execute that job, and the steps of which the job is comprised
- Workflows may also include an env section, which sets environment variables for use during the run

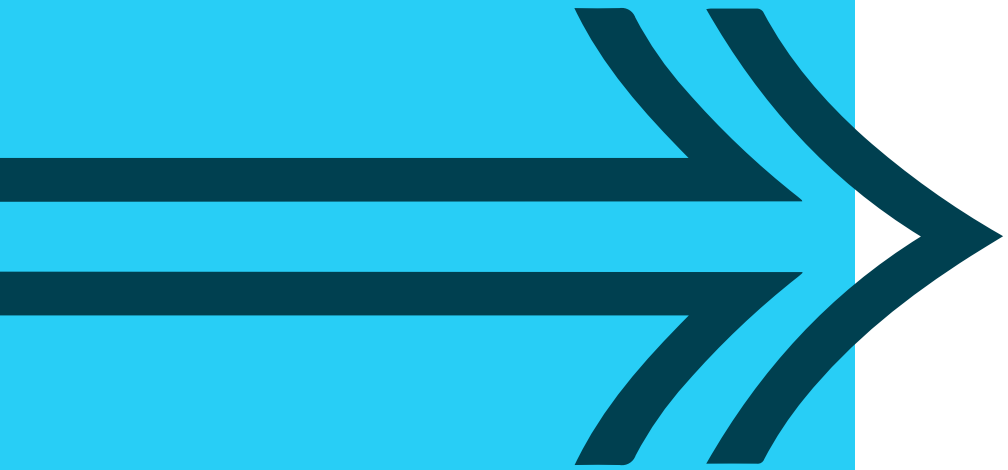
BASIC WORKFLOW

```
name: example workflow

on:
    push:
        branches: [“dev”]
    pull_request:
        branches: [“main”]
    workflow_dispatch:

jobs:
    example_job:
        runs-on: ubuntu-latest
        steps:
            - uses: actions/checkout@v3
            - name: example step
              run: echo “Hello World”
```

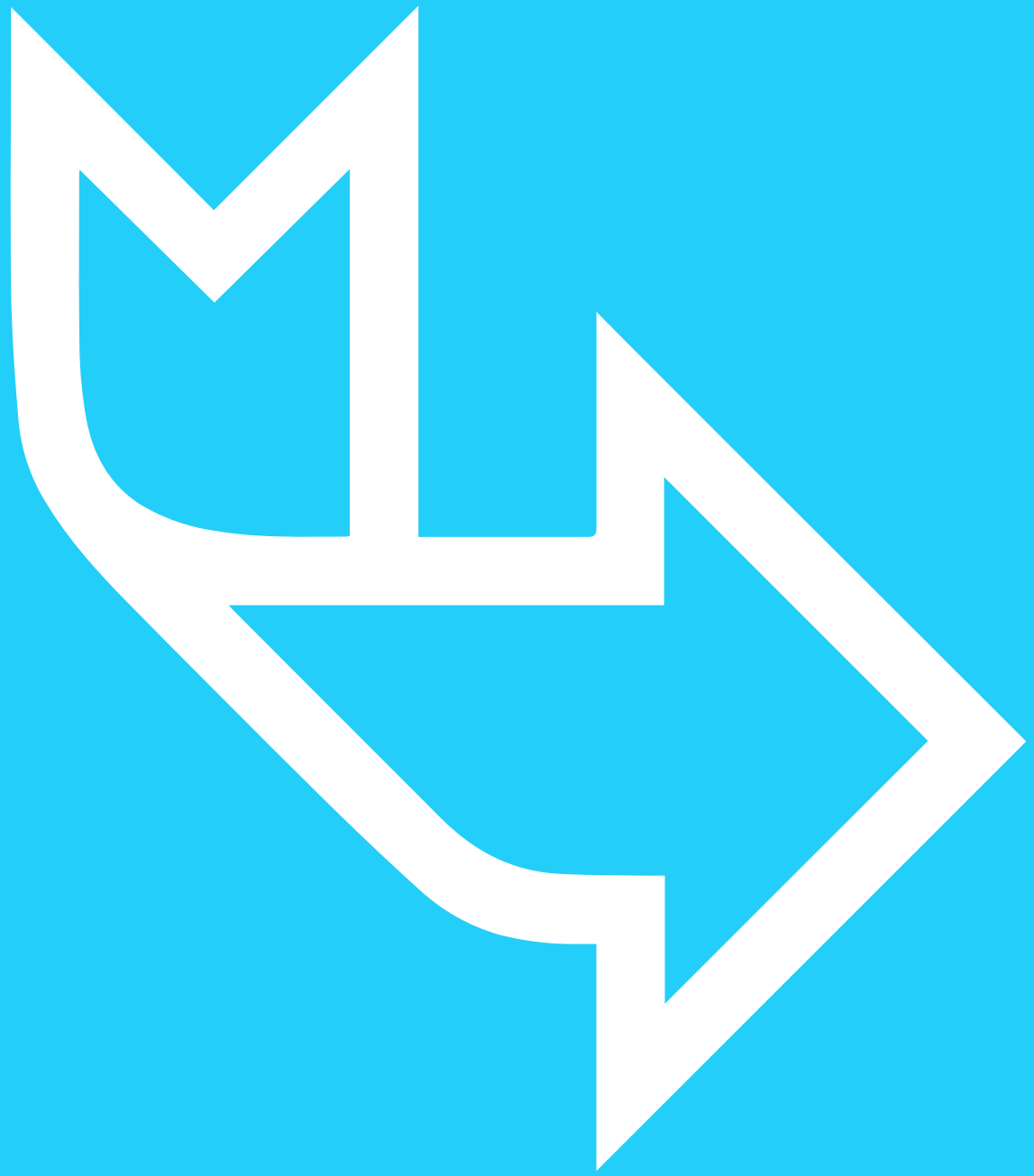
CONFIGURING ACTIONS



- From the main page of a GitHub repository, Actions can be enabled via the Actions tab
- If actions is not already enabled, the actions tab will provide templates for setting up workflows
- Among the provided templates are suggested templates based on the language(s) detected in the repository
- Various templates are also available for deploying to various cloud platforms
- If you want to start from scratch, a simple Hello World template is also available
- When you add a workflow, the .github/workflows directory will be created automatically

CUSTOM WORKFLOWS

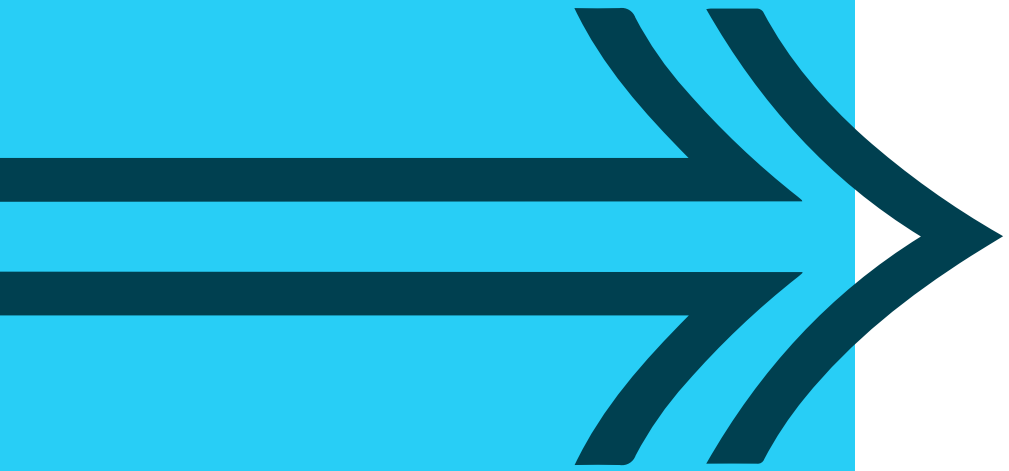
- The YAML file defining a workflow is stored in the repository, and may be modified as with any other file
- This allows you to modify templates to customise them to your needs, and update the CI as a project evolves
- To add functionality to a workflow, a variety of runners and modules are available to give access to specific functionality
- Alternately, for simple jobs, you can use a basic ubuntu runner and create a shell script which defines the necessary steps



Exercise 2: setting up actions

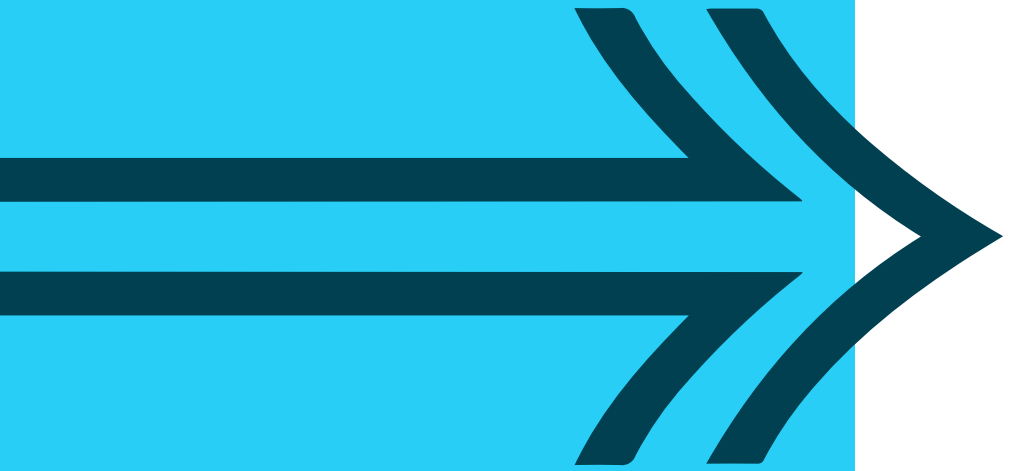
- Enable Actions on an existing GitHub repository
- Update the workflow to run automated tests for your code

RUNNERS



- Recall that each job within a workflow gets its' own **runner** – a container which is spun up to execute the steps of that job
- GitHub provides hosted runners for Windows, MacOS and Ubuntu – these have a range of preinstalled software that can be used during runs
- The documentation for Actions has a comprehensive list of the software installed on each of these runners
- If you need a very specific build environment, and want to save time configuring the default runners for each build, you can self-host runners
- Self-hosted runners require more effort to set up than GitHub-hosted, but allow you to use custom container images with the exact build environment you need

MODULES



- In addition to running shell scripts, GitHub Actions has a range of modules – these are the 'Actions' in 'GitHub Actions' - available for use as part of jobs
- There are a variety of actions available from GitHub, under the actions namespace, and many more third-party actions which offer a range of functionality
- Actions are invoked via the 'uses' key. The action is referenced as 'namespace/action@version'
- Many actions require inputs to be given, these are provided via a 'with' mapping following the invocation
- If needed, it is possible to create and publish your own custom actions by writing a node application with the desired functionality, and an 'action.yml' file which defines usage details

REVIEW

- GitHub Actions is a managed, container-based service for implementing scalable, cost-effective CI on GitHub repositories
- Actions executes *workflows*, themselves subdivided into *jobs* which are in turn comprised of *steps*
- Workflows are defined by YAML files which specify the details of the workflow via a set of key-value pairs
- A wide range of YAML templates are provided as starting points, which can then be customised to your own needs
- Modules, called **actions**, offered by GitHub and by third-parties provide extended functionality

End of section