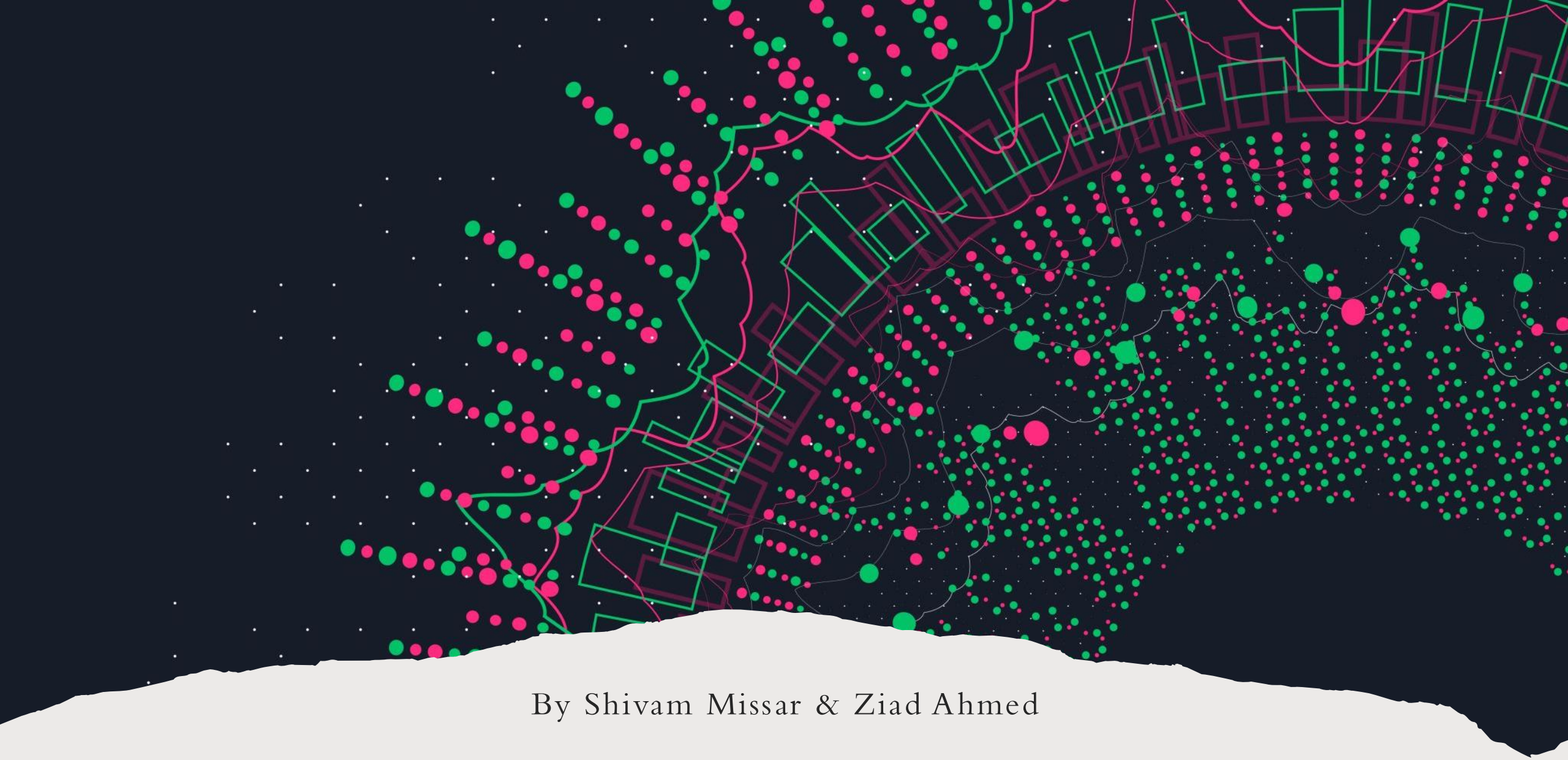


By Shivam Missar & Ziad Ahmed

GITHUB ACTIONS

TEAM INTRODUCTION

Team Memebers:

- Shivam Missar – (BSc) Software Engineer & (MSc) Software Development graduate
- Ziad Ahmed - (BSc) Software Engineer graduate

Placement Overview:

- A 6-week internship scheme provided by Nottingham Trent University (NTU)

Carrying out two projects:

- ED-Simulation (Completed)
- GitHub Actions

Objectives:

- To gain more knowledge of continuous integration and delivery via GitHub Actions

GITHUB ACTIONS

WHAT IS GITHUB ACTIONS?

- GitHub Actions is a tool that helps automate tasks in your project. Instead of doing everything manually, like checking your code, testing it, or deploying it, GitHub Actions can automatically do these things for you.
- It is a built-in tool within GitHub itself

What are the benefits of using GitHub Actions?

- Saves time
- Keeps projects organised
- Prevents mistakes

HOW GITHUB ACTIONS WORKS

- **Triggers:** triggered by specific events (e.g., push, pull request, issue creation).
- **Workflow: r**esponse to those triggers of when it should happen and should happen.
- **Actions:** steps that perform tasks, like running tests

HOW ARE THEY APPLIED?

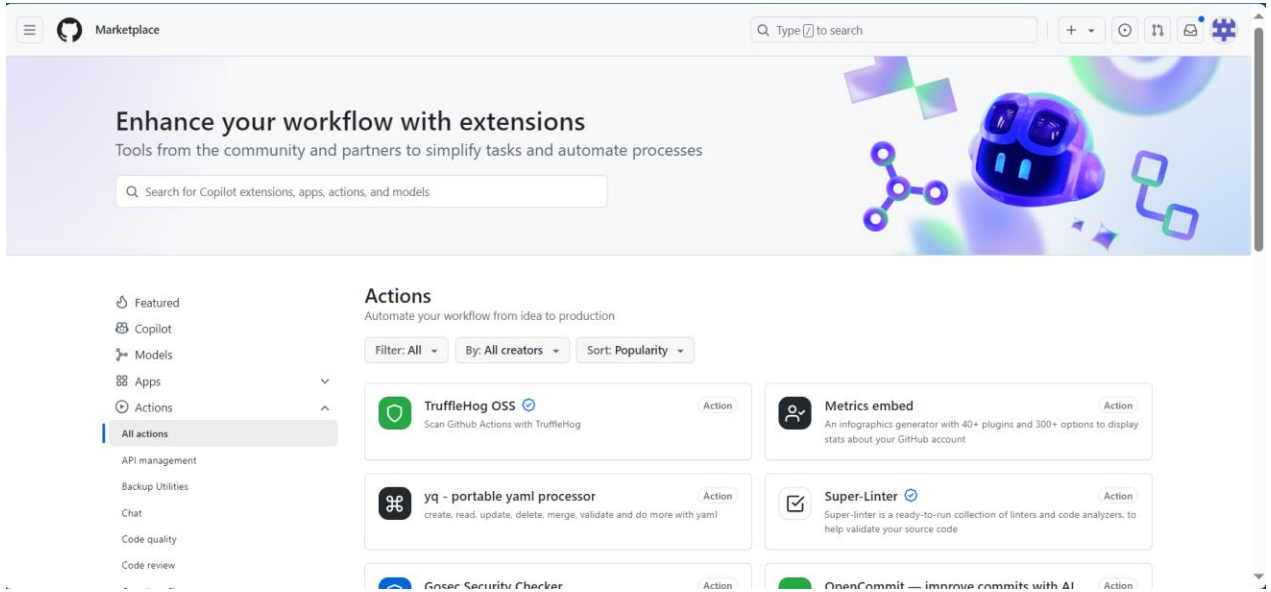
- To create a GitHub Action, it uses something called 'YAML' (Yet Another Markup Language) syntax and this used for defining workflows. These YAML files are used to setup the automated process.
- GitHub Actions can be used applied in a variety of ways through these YAML files such as:
  - Automated Testing
  - Continuous Integration
  - Deployment Automation
  - Code Quality Checks

GITHUB ACTIONS MARKETPLACE

- The [GitHub Actions Marketplace] is like a store where you can find pre-built actions created by others.
- These actions are created by the community, companies, or GitHub itself, and you can easily add them to your workflows with just a few lines of code.

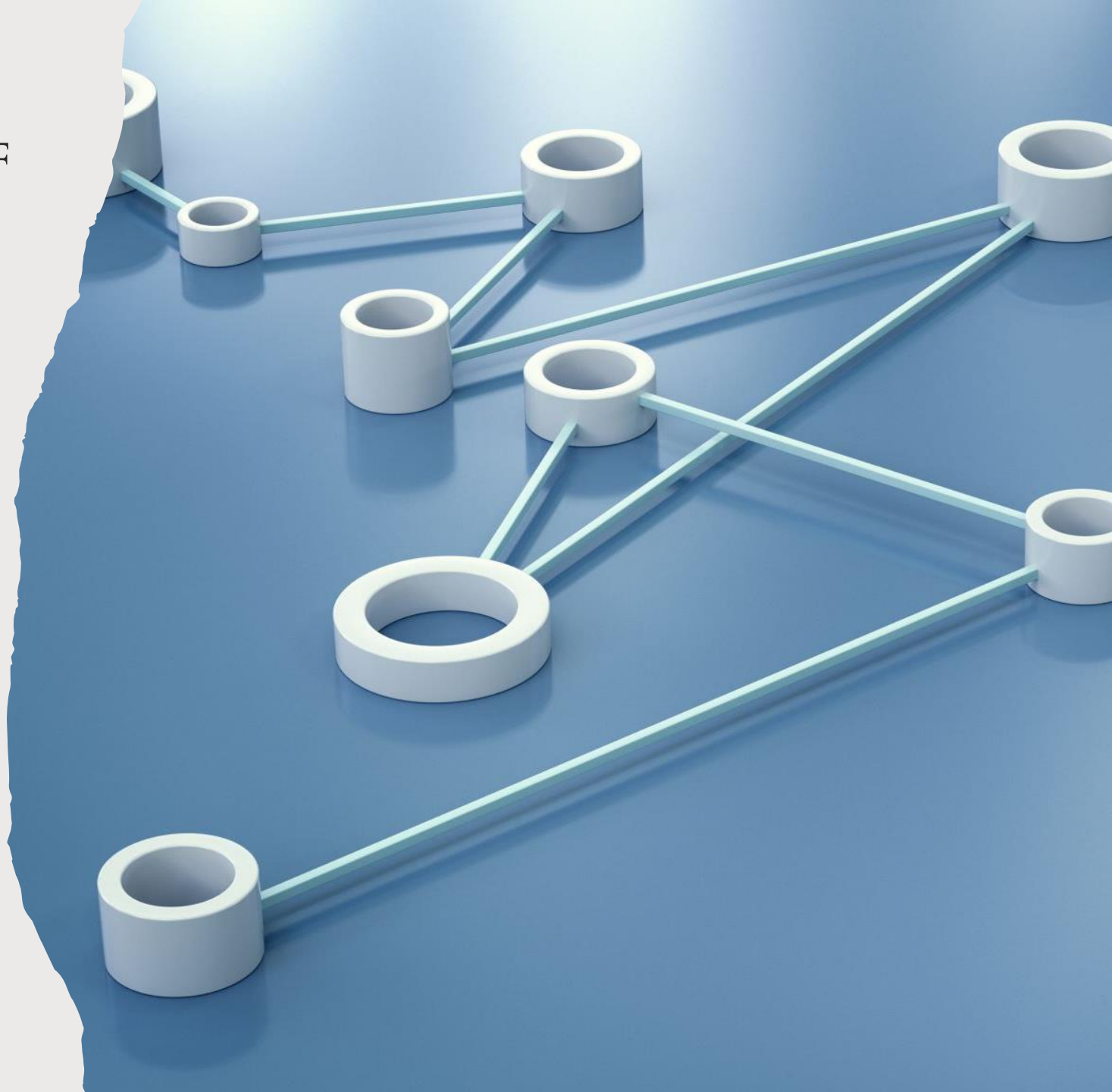
Why use the Marketplace?

- **Saves time:** No need to reinvent the wheel; just use what's already built.
- **Reliable:** Many of the actions are widely used and tested by other developers.
- **Easy to use:** You can add them quickly by copying a few lines from the marketplace page.



INTEGRATION OF OTHER TOOLS

- GitHub actions is a very versatile tool, working with hundreds of different programming languages and third-party applications
- Integration with cloud-based services like Azure, Amazon Web Services (AWS) and much more
- Notification system via your registered email



GITHUB ACTIONS COST

FREE PLAN OVERVIEW FOR ORGANISATIONS

- **Included Minutes**: 2,000 per month for Windows & Ubuntu; macOS requires pay for usage
- **Storage included:** Storage: 500 MB of storage.

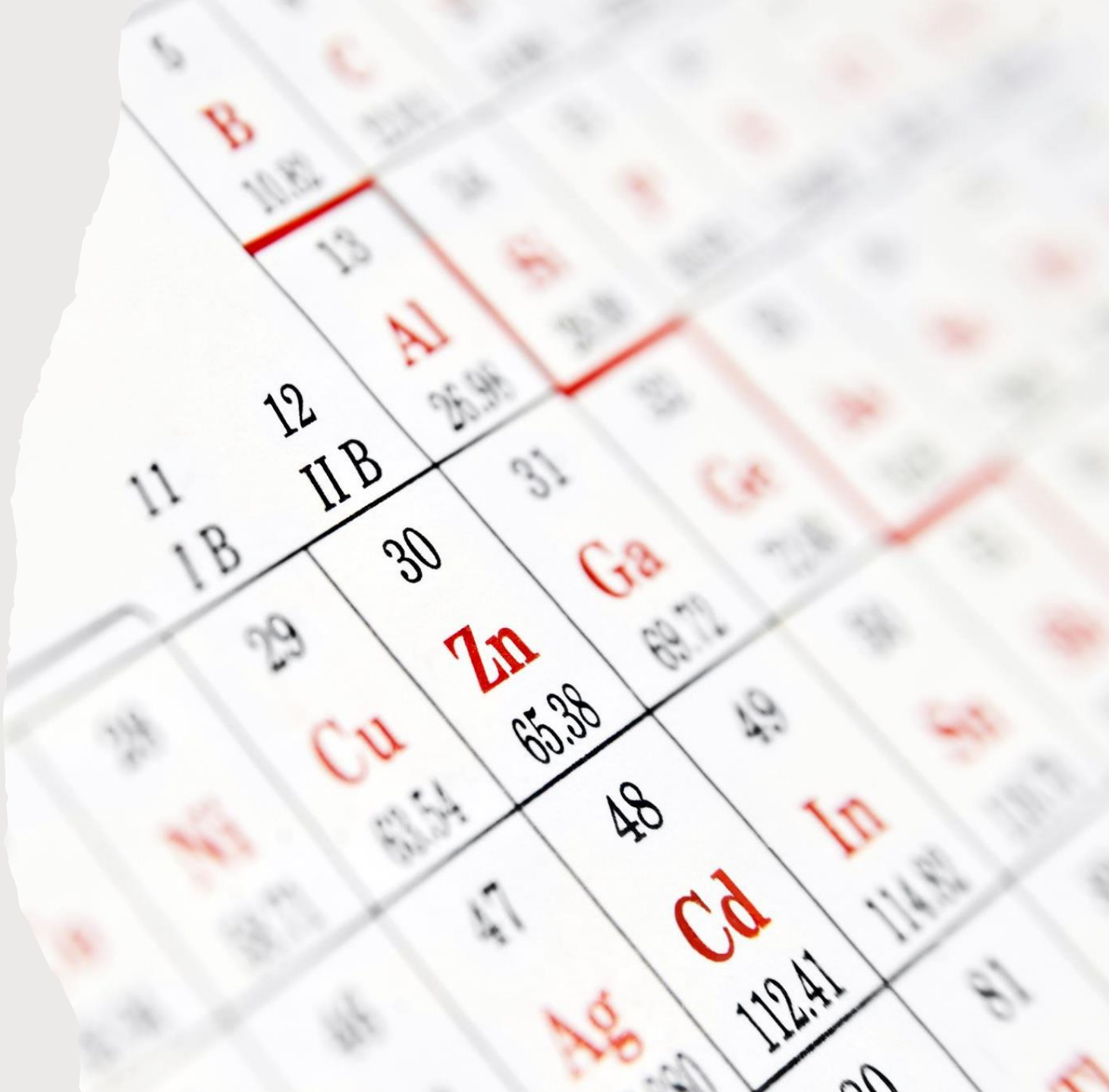
PAID PLAN OVERVIEW FOR ORGANIZATIONS

GitHub Team:

- 3,000 minutes/month.
- 2 GB storage.

GitHub Enterprise:

- 50,000 minutes/month.
- 50 GB storage.



PLAN COMPARISON FREE VS PAID PLANS

Free Tier:

- 2,000 minutes/month
- 500 MB storage.

GitHub Team (Paid Plan):

- 3,000 minutes/month
- 2 GB storage.

GitHub Enterprise (Paid Plan):

- 50,000 minutes/month
- 50 GB storage.

BILLING CONSIDERATIONS

- Minutes and storage are shared across all repositories in the organisation.
- Exceeding limits incurs extra charges at the end of each billing cycle.
- Usage monitoring is available under "Billing & Plans" in the settings.
- Total billable is not equal to run time, GitHub will always round up the minutes

GENERAL CLARIFICATIONS

The time a job takes depends on the type of task being performed:

- **Small tasks:** Only a few minutes.
- **Larger tasks**: May take significantly more time.

The storage capacity depends on the size of files

- **Small projects:** May take less storage (caches and artifacts)
- **Larger projects**: May take significantly more storage (caches and artifacts)

HOW MANY MINUTES DO JOBS USE?

Python Project:

- "Hello World" script: <1 minute.
- Small Python project with unit tests: 3-5 minutes.

R Project:

- "Hello World" script: <1 minute.
- Small R project with libraries like ggplot2: 3-6 minutes.

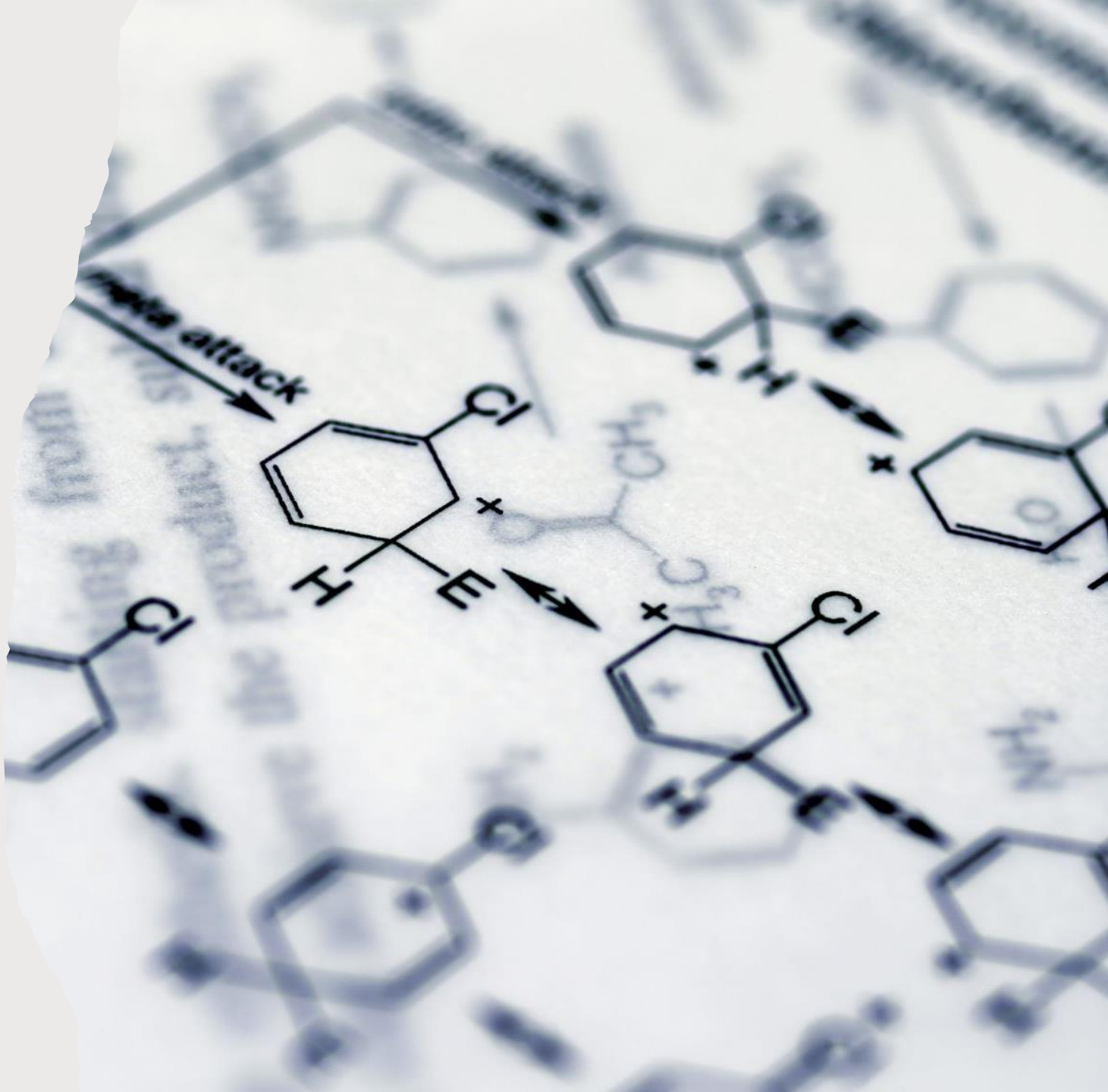
HOW MUCH STORAGE DO I NEED?

Python Project:

- Basic project with small scripts and testing: 50-100 KB.
- Cached dependencies (e.g., matplotlib, pandas): 50-100 KB.

R Project:

- Code and small plot artifacts: 50-150 KB.
- Cached libraries (ggplot2, dplyr): 50-200 KB.

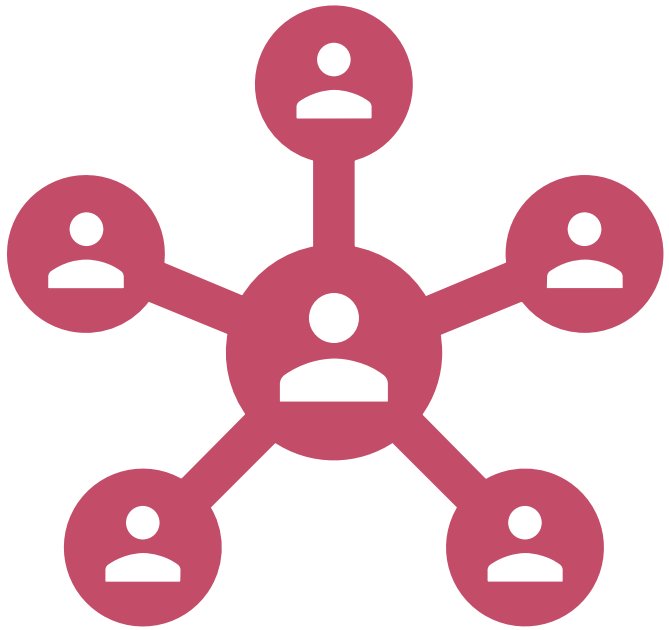


GENERAL LIMITATIONS OF WORKFLOWS PER PLAN

- **Max task/job runtime:** 6 hours.
- **Workflow duration limit:** 35 days.
- **Concurrent job limits:**
  - Free Tier: 20 jobs.
  - GitHub Team: 60 jobs.
  - GitHub Enterprise: 180 jobs.

BENEFITS OF A PAID PLAN IN USING GITHUB ACTIONS FOR ORGANISATIONS

- **Scalable**: Suitable for both small teams and large enterprises.
- **Cost-Effective**: paid plans for growing needs.
- **Security & Compliance:** Advanced security features,
- **Customization & Flexibility**: Paid plans allow more allowances
- **Priority Support:** priority customer support



SUMMARY OF COSTS

- GitHub Actions offers a range of plans suited to different organization sizes.
- Free and paid options cater to both small teams and large enterprises.
- Monitoring usage ensures cost management and efficient resource allocation.

GITHUB ACTIONS NEEDS TO KNOW

GENERAL INFORMATION

- GitHub Actions works the same regardless of the programming language used whether it is Python, JavaScript, R or any other language.
- The only difference is setting up the file
- You must define which operating system Actions will need to be designed for (windows, ubuntu or macOS).

KEY TERMS

- **name:** Labels the workflow or job.
- **Job**: A collection of steps that are ran.
- **YAML**: The syntax used for writing GitHub Actions, which stands for Yet Another Markup Language.
- **on**: Defines the events that trigger the workflow (push or pull...)
- **runs-on**: Defines which operating system the workflow should run on and what version
- **steps:** Details each action that will be performed
- **with**: Used to specify the version of the language
- **build**: Compiles the code
- **run**: Executes commands directly

GITHUB ACTION RUNNERS

A **runner** refers to a **server or machine** that runs the jobs specified in a GitHub Actions workflow

GitHub-Hosted Runners:

- **Pros**: No setup required
- **Cons**: Limited customization and compute resources.

Self-Hosted Runners:

- **Pros**: Full control over the environment
- **Cons**: Requires maintenance and management

CASE EXAMPLES

AUTOMATIC TESTING

```yaml
name: Test Recipe App

on: [push]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '16'
      - run: npm install
      - run: npm test
```

CONTINUOUS INTEGRATION (CI)

```yaml
name: CI for Messaging App

on: [push]

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '18'
      - run: npm install
      - run: npm run build
      - run: npm test
```

GITHUB ACTIONS LIVE DEMONSTRATION

- Total of 2 scripts showing different examples
- Mix of both Python and R languages
- The basics of GitHub Actions
- How they are triggered

LIVE DEMONSTRATION

SUMMARY OF DEMONSTRATION

- Actions Created:
- Python & R Actions:

  Python/R installed checker

  Code Formatter

  Assert function for testing addition of two numbers

  Assert function for testing string function output

  Checking for package dependencies and if are included in the files

  K-Number Checker

  Ward-Name checker using data read from a text file (wardnames.txt)

DATA CONSUMED PER DAY

- **05/09/2024** – 34 commits, 102 workflow runs and 214 minutes consumed with 6 scripts
- **06/09/2024** – 48 commits, 338 workflow runs and 166 minutes consumed with 7 scripts
- **11/09/2024** – 23 commits, 557 workflow runs 448 minutes consumed with 8 scripts
- **12/09/2024** – 7 commits, 639 workflow runs, 484 minutes consumed with 10 scripts

KEY ADVICE

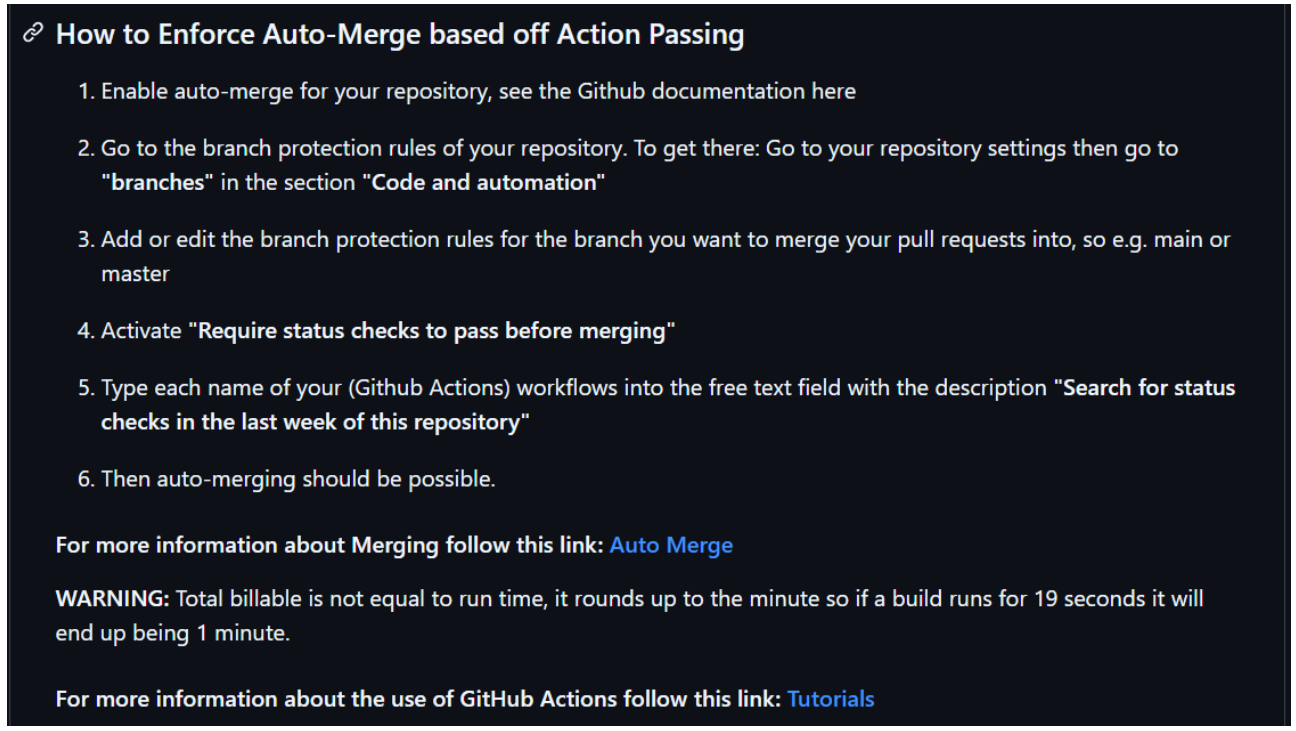
SECURITY INTEGRATION THROUGH AUTO MERGING

Prevent Unchecked Code from Being Merged:

- Ensures all status checks (e.g., security scans) must pass before merging, reducing the risk of introducing vulnerabilities or bugs into production.

Controlled Codebase:

- Only authorized individuals with verified code can merge changes, minimizing the risk of unauthorized or unverified code reaching sensitive branches.



How to Enforce Auto-Merge based off Action Passing

1. Enable auto-merge for your repository, see the Github documentation here
2. Go to the branch protection rules of your repository. To get there: Go to your repository settings then go to **"branches"** in the section **"Code and automation"**
3. Add or edit the branch protection rules for the branch you want to merge your pull requests into, so e.g. main or master
4. Activate **"Require status checks to pass before merging"**
5. Type each name of your (Github Actions) workflows into the free text field with the description **"Search for status checks in the last week of this repository"**
6. Then auto-merging should be possible.

For more information about Merging follow this link: Auto Merge

WARNING: Total billable is not equal to run time, it rounds up to the minute so if a build runs for 19 seconds it will end up being 1 minute.

For more information about the use of GitHub Actions follow this link: Tutorials

OUR UNDERSTANDING

- Run and Commit are two different concepts in GitHub
- Run refers to running a script
- Commit refers to updating the changes to file
- Commenting out an action helps avoid running the execution saving minutes and storage
- You do not need to place a text file within the same directory as the GitHub Actions when reading data from that text file
- If you want to actions to run without pushes from certain file types, you can specify those file types using '**paths-ignore: - **/FILENAME**' different from gitignore

CHALLENGES WE FACED

- Limited on minutes
- Based on the OS type specified we had to program using that language for example Windows requires PowerShell and GitHub Actions does not allow mix and match
- Could not test specific workflows before committing
- A lot of redundancy in Actions
- GitHub Actions is complex to use and takes time to understand

GITHUB ACTIONS LIMITATIONS

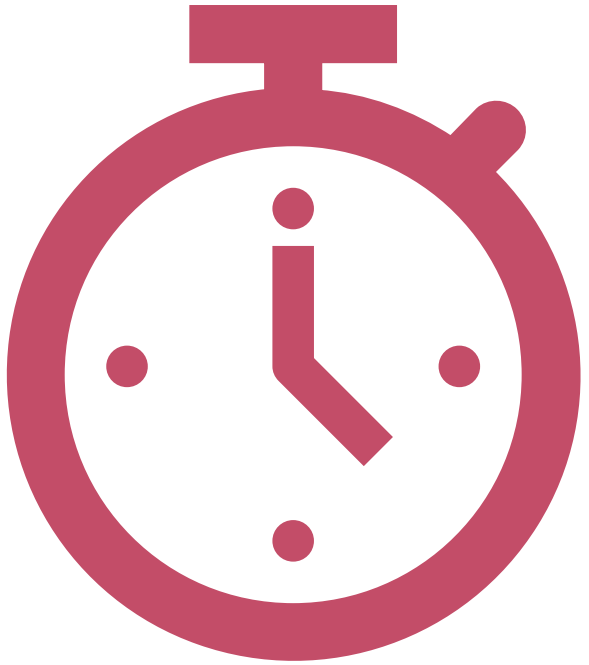
- There is no current automated way of checking for specified file types like .txt, .sql and vice versa, this can only be achieved via the .gitignore file
- GitHub Actions might not support the latest version of the programming language
- UI is not helpful
- Error messages are not helpful
- Job run times are unexpectedly long for a short tasks
- No viable way of testing a workflow – runs all workflows at once to test one workflow
- Cannot create actions to do tasks which require manual interactions (e.g., merge requests, pull requests)

CONCLUSION

OUR FINDINGS

- Limited minutes meant limited commits
- High costs to maintain a codebase
- Good for automating everyday tasks like testing solutions before merging with main codebase
- Has support with thousands of different templates and reusable code saving time and development

- GitHub Actions cannot completely block commits. However, we can add force checks before passing



SUMMARY

A ROUND UP

- GitHub Actions is a great way of automating everyday tasks
- Great tool for testing outputs of functionalities to ensure they work
- Versatile with the support for many different languages
- Open source meaning that developers can create reusable code to avoid redundancy
- Can be used for some many different tasks
- Limited minutes = limited commits
- Certain tasks like approval of merges and commits, cannot be converted into an automated task.
- May not support the latest update of the preferred programming language

THANK YOU

- Ziad.ahmed@nhs.net
- Shivam.missar@nuh.nhs.uk

QUESTIONS?