

# Assignment - 1

## Github Actions

# CI/CD PIPELINE

A CI/CD pipeline is a set of automated processes that facilitate continuous integration and continuous development/delivery

Continuous integration(CI) involves automatically building, testing and integrating code changes into a shared repository whenever developers make changes. This ensures that code changes are regularly validated and integrated into the main codebase.

Continuous Delivery/Deployment(CD) involves automating the deployment of validated code changes to production environments. Continuous Delivery focuses on automatically deploying code changes to staging or preproduction environments for further testing, while continuous deployment goes a step further by automatically deploying changes directly to production environments after passing all the tests.

Overall, CI/CD pipelines streamline the software development and deployment process, enabling teams to deliver high quality software more quickly and efficiently.

## Git and Github

- Git:
  - Git is a distributed version control system (DVCS) used for tracking changes in source code during software development.
  - It allows multiple developers to collaborate on a project simultaneously.
  - Git enables developers to create branches, merge changes, revert to previous versions, and track the history of changes.
  - Git operates locally on the developer's machine and doesn't require a constant connection to a central server.
- GitHub:
  - GitHub is a web-based platform that provides hosting for Git repositories.
  - It adds a layer of collaboration and social networking features on top of Git, allowing developers to share code, collaborate on projects, and contribute to open-source projects.
  - GitHub offers features such as issue tracking, project management tools, wikis, and pull requests for code review and collaboration.
  - It serves as a central hub for developers to store, manage, and share their Git repositories.

## Key Differences:

Feature	Git	GitHub
Type	Version control system (software)	Web-based platform (service)
Functionality	Tracks changes in source code	Provides hosting, collaboration, and social features

Operation	Operates locally on developer's machine	Accessed via web browser and Git client
Collaboration	Limited collaboration features	Extensive collaboration tools, pull requests, etc.
Hosting	Repositories stored locally or on servers	Repositories hosted on GitHub's servers
Pricing	Open-source and free to use	Offers free and paid plans with additional features
Examples	Git command line, Git GUI clients	GitHub website, GitHub Desktop, GitHub CLI

In summary, Git is the underlying version control system used for tracking changes in source code, while GitHub is a web-based platform that provides hosting for Git repositories along with collaboration features and social networking aspects for developers.

## Git commands

1. **git init**: Initializes a new Git repository in the current directory. This command creates a hidden **.git** directory where Git stores its internal data and configuration files.
2. **git clone <repository URL>**: Clones an existing Git repository from a remote location (such as GitHub, GitLab, or Bitbucket) to your local machine. This command creates a copy of the repository, including all of its files, commit history, and branches.
3. **git add <file>**: Adds a file or changes in a file to the staging area, preparing them to be included in the next commit. You can use **git add .** to add all changes in the current directory.
4. **git commit -m "commit message"**: Records changes to the repository, creating a new commit with a descriptive commit message. The commit message should briefly describe the changes made in the commit.

5. **git status**: Displays the current status of the repository, including any untracked files, changes not staged for commit, and changes ready to be committed (in the staging area).
6. **git diff**: Shows the differences between the changes in the working directory and the changes in the staging area (index). This command helps you review the changes before committing them.
7. **git diff <commit1> <commit2>**: Shows the differences between two commits. You can specify commit hashes, branch names, or other references to compare different versions of the repository.
8. **git pull**: Fetches changes from the remote repository and merges them into the current branch. This command is used to update your local repository with changes made by others.
9. **git push**: Pushes commits from your local repository to the remote repository. This command is used to share your changes with others and keep the remote repository up-to-date.
10. **git branch**: Lists all branches in the repository. By default, it shows only local branches. You can use **git branch -a** to list both local and remote branches.
11. **git checkout <branch>**: Switches to the specified branch. This command allows you to work on different branches within the repository.
12. **git merge <branch>**: Merges the specified branch into the current branch. This command combines changes from another branch into the current branch.

13. **git remote add <name> <url>**: Adds a new remote repository with the specified name and URL. This command allows you to manage connections to remote repositories.
14. **git remote -v**: Lists all remote repositories associated with the current repository, along with their URLs. This command is useful for verifying remote configurations.
15. **git log**: Displays a list of commits in reverse chronological order. By default, it shows the commit hash, author, date, and commit message for each commit.
16. **git remote show <remote>**: Shows information about the specified remote repository, such as the URL, tracking branches, and remote branches.
17. **git fetch <remote>**: Fetches changes from the specified remote repository but does not merge them into the local branches. This command updates the remote tracking branches in your local repository.
18. **git checkout -b <branch>**: Creates a new branch with the specified name and switches to it. This command is a shortcut for creating a new branch and then checking it out.
19. **git branch -d <branch>**: Deletes the specified branch from the repository. The branch must be fully merged into other branches before it can be deleted.
20. **git branch -m <old-name> <new-name>**: Renames the specified branch from **<old-name>** to **<new-name>**. This command can be used to rename the current branch or any other branch in the repository.

21. **git reset <commit>**: Resets the current branch to the specified commit, discarding all changes after that commit. This command can be used to undo commits or reset the repository to a previous state.
22. **git revert <commit>**: Creates a new commit that undoes the changes introduced by the specified commit. This command is used to revert specific commits while preserving the commit history.
23. **git stash**: Temporarily stores changes in the working directory and staging area, allowing you to switch branches or perform other operations without committing the changes.
24. **git stash pop**: Applies the most recently stashed changes to the working directory and staging area, removing them from the stash.
25. **git cherry-pick <commit>**: Applies the changes introduced by the specified commit to the current branch. This command is useful for selectively applying commits from one branch to another.
26. **git rebase <branch>**: Reapplies commits from the current branch onto the specified branch, rewriting the commit history. This command is often used to integrate changes from one branch into another.
27. **git tag <tag-name> <commit>**: Creates a lightweight tag at the specified commit. Tags are used to mark specific points in the commit history, such as release versions or milestones.
28. **git push --tags**: Pushes all tags from the local repository to the remote repository. This command is used to share tags with others when pushing changes.

29. **git submodule**: Manages Git submodules within a repository. Submodules allow you to include other Git repositories as subdirectories within your main repository.
30. **git clean**: Removes untracked files from the working directory. This command is used to clean up the working directory by removing files that are not under version control.

## Github Actions

- GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform.
- Allows you to automate your build, test, and deployment pipeline.
- Allows you to create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.
- lets you run workflows when other events happen in your repository.
- GitHub provides Linux, Windows, and macOS virtual machines to run your workflows, or you can host your own self-hosted runners in your own data center or cloud infrastructure.

## Components

You can configure a GitHub Actions workflow to be triggered when an event occurs in your repository, such as a pull request being opened or an issue being created. Your workflow contains one or more jobs which can run in sequential order or in parallel. Each job will run inside its own virtual machine runner, or inside a container, and has one or more steps that either run a script that you define or run an action, which is a reusable extension that can simplify your workflow.

### Workflows -

- A configurable automated process that will run one or more jobs.
- Workflows are defined by a YAML file checked in to your repository
- Workflow runs when triggered by an event in your repository, or they can be triggered manually, or at a defined schedule.
- Workflows are defined in the `.github/workflows` directory in a repository, and a repository can have multiple workflows

### Events -

- An event is a specific activity in a repository that triggers a workflow run.
- activity can originate from GitHub when someone creates a pull request, opens an issue, or pushes a commit to a repository.
- You can also trigger a workflow to run on a [schedule](#), by [posting to a REST API](#), or manually.

### Jobs -

- A job is a set of steps in a workflow that is executed on the same runner.

- Each step is either a shell script that will be executed, or an action that will be run.
- Steps are executed in order and are dependent on each other.
- Since each step is executed on the same runner, you can share data from one step to another.
- You can configure a job's dependencies with other jobs; by default, jobs have no dependencies and run in parallel with each other.
- When a job takes a dependency on another job, it will wait for the dependent job to complete before it can run.

## Actions -

- An action is a custom application for the GitHub Actions platform that performs a complex but frequently repeated task.
- Use an action to help reduce the amount of repetitive code that you write in your workflow files.
- An action can pull your git repository from GitHub, set up the correct toolchain for your build environment, or set up the authentication to your cloud provider.
- You can write your own actions, or you can find actions to use in your workflows in the GitHub Marketplace.

## Runners -

- A runner is a server that runs your workflows when they're triggered.
- Each runner can run a single job at a time.
- GitHub provides Ubuntu Linux, Microsoft Windows, and macOS runners to run your workflows
- each workflow run executes in a fresh, newly-provisioned virtual machine.
- GitHub also offers larger runners, which are available in larger configurations.
- If you need a different operating system or require a specific hardware configuration, you can host your own runners.
-



# Workflow Implementation

- Creating directories and required files

```
MINGW64:/c/Users/ShreyaSharma/desktop/GithubAct/.github/workflows

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~
$ cd desktop

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop
$ mkdir GithubAct

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop
$ cd GithubAct

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop/GithubAct
$ git init
Initialized empty Git repository in C:/Users/ShreyaSharma/Desktop/GithubAct/.git/

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop/GithubAct (master)
$ git remote add origin https://github.com/xshreya/GithubAct.git

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop/GithubAct (master)
$ mkdir .github

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop/GithubAct (master)
$ cd .github

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop/GithubAct/.github (master)
$ mkdir workflows

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop/GithubAct/.github (master)
$ cd workflows

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop/GithubAct/.github/workflows (master)
$ echo myworkflow.yml
myWorkflow.yml

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/desktop/GithubAct/.github/workflows (master)
$ notepad myWorkflow.yml
```

```
MINGW64:/c/Users/ShreyaSharma/Desktop/GithubAct

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/Desktop/GithubAct (main)
$ git pull origin main --allow-unrelated-histories
From https://github.com/xshreya/GithubAct
 * branch          main          -> FETCH_HEAD
Already up to date.

AzureAD+ShreyaSharma@CEQ-ICT-DESKTOP-003 MINGW64 ~/Desktop/GithubAct (main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 1.15 KiB | 1.15 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/xshreya/GithubAct.git
 ab6887f..21d52f2  main -> main
```

- Writing code to the myWorkflow.yml file

```
1  name: Python CI
2
3  on:
4    push:
5      branches:
6        - main
7    pull_request:
8      branches:
9        - main
10
11  jobs:
12    build:
13      runs-on: ubuntu-latest
14
15      steps:
16        - name: Checkout Repository
17          uses: actions/checkout@v2
18
19        - name: Set up Python
20          uses: actions/setup-python@v2
21          with:
22            python-version: '3.x' # Specify the Python version you want to use
23
24        - name: Run tests
25          run: |
26            python try.py
27
```

- Multiline.yml

```
1  name: Run Multiline Script
2
3  on:
4    push:
5      branches:
6        - main
7
8  jobs:
9    run-script:
10     runs-on: ubuntu-latest
11     steps:
12       - name: Checkout repository
13         uses: actions/checkout@v2
14
15       - name: Run Multiline Script
16         run: |
17           echo "This is a multiline script"
18           echo "It can contain multiple commands"
19           echo "For example, you can run any bash commands here"
20           echo "You can also include variables or conditionals"
21           # Add your multiline script here
22
```

- javascript.yml

```
1
2 name: Run JavaScript Script
3
4 on:
5   push:
6     branches:
7       - main
8
9 jobs:
10  run-script:
11    runs-on: ubuntu-latest
12    steps:
13      - name: Checkout repository
14        uses: actions/checkout@v2
15
16      - name: Install Node.js
17        uses: actions/setup-node@v2
18        with:
19          node-version: '14'
20
21      - name: Run JavaScript Script
22        run: |
23          node -e '
24            console.log("This is a multiline JavaScript script");
25            console.log("You can write multiple lines of JavaScript code here");
26            console.log("For example, you can perform computations, manipulate data, etc.");
27            // Add your multiline JavaScript script here
28          '
```

- Matrix and env variables demo

```
1  name: Environment Variables and Matrix Demo
2
3  on:
4    push:
5      branches:
6        - main
7
8  jobs:
9    test:
10     runs-on: ubuntu-latest
11     strategy:
12       matrix:
13         node-version: [12.x, 14.x, 16.x]
14     env:
15       ENV_VAR_1: "Value 1"
16       ENV_VAR_2: "Value 2"
17     steps:
18       - name: Checkout repository
19         uses: actions/checkout@v2
20
21       - name: Set up Node.js ${{ matrix.node-version }}
22         uses: actions/setup-node@v2
23         with:
24           node-version: ${{ matrix.node-version }}
25
26       - name: Display Environment Variables
27         run: |
28           echo "Environment variable 1: $ENV_VAR_1"
29           echo "Environment variable 2: $ENV_VAR_2"
30           echo "Matrix node version: ${{ matrix.node-version }}"
31
```

## Running the code

! myWorkflow.yaml

! multiline.yaml

! javascript.yaml

! yetanother.yaml X

try.py

```
.github > workflows > ! yetanother.yaml
 8 jobs:
 9 test:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG AZURE

PS C:\Users\ShreyaSharma\Desktop\GithubAct> git commit -m "updated"
create mode 100644 try.py
PS C:\Users\ShreyaSharma\Desktop\GithubAct> git push origin main
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 698 bytes | 698.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/xshreya/GithubAct.git
21d52f2..106502a main -> main
PS C:\Users\ShreyaSharma\Desktop\GithubAct> git add .
PS C:\Users\ShreyaSharma\Desktop\GithubAct> git commit -m "updated"
[main 62a7b03] updated
1 file changed, 5 deletions(-)
PS C:\Users\ShreyaSharma\Desktop\GithubAct> git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 432 bytes | 432.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
106502a..62a7b03 main -> main
PS C:\Users\ShreyaSharma\Desktop\GithubAct> git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    javascript.yaml
    multiline.yaml
    yetanother.yaml
```

...

! myWorkflow.yaml

! multiline.yaml

! javascript.yaml

! yetanother.yaml X

try.py

.github &gt; workflows &gt; ! yetanother.yaml

8 jobs:

9 test:

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

CODE REFERENCE LOG

AZURE

nothing added to commit but untracked files present (use "git add" to track)

PS C:\Users\ShreyaSharma\Desktop\GithubAct> git add .

PS C:\Users\ShreyaSharma\Desktop\GithubAct> git status

On branch main

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: javascript.yaml

new file: multiline.yaml

new file: yetanother.yaml

Changes not staged for commit:

deleted: javascript.yaml

deleted: multiline.yaml

deleted: yetanother.yaml

Untracked files:

(use "git add <file>..." to include in what will be committed)

.github/workflows/javascript.yaml

.github/workflows/yetanother.yaml

PS C:\Users\ShreyaSharma\Desktop\GithubAct> git add .

PS C:\Users\ShreyaSharma\Desktop\GithubAct> git status

On branch main

(use "git restore --staged <file>..." to unstage)

new file: .github/workflows/javascript.yaml

new file: .github/workflows/multiline.yaml

new file: .github/workflows/yetanother.yaml

PS C:\Users\ShreyaSharma\Desktop\GithubAct> git commit -m "added other workflows"

[main 09dc093] added other workflows

3 files changed, 79 insertions(+)

create mode 100644 .github/workflows/javascript.yaml

create mode 100644 .github/workflows/multiline.yaml

create mode 100644 .github/workflows/yetanother.yaml

PS C:\Users\ShreyaSharma\Desktop\GithubAct> git push origin main

Enumerating objects: 10, done.

Counting objects: 100% (10/10), done.

Delta compression using up to 12 threads

Compressing objects: 100% (6/6), done.

Writing objects: 100% (7/7), 1.38 KiB | 1.38 MiB/s, done.

Total 7 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/xshreya/GithubAct.git

62a7b03..09dc093 main -> main

PS C:\Users\ShreyaSharma\Desktop\GithubAct> █

# Outputs

The screenshot shows the GitHub Actions interface for the repository `xshreya / GithubAct`. The `Actions` tab is selected, displaying a list of workflow runs under the heading "All workflows". A sidebar on the left lists various workflow files: `Environment Variables and Matrix Demo`, `Python CI`, `Run JavaScript Script`, `Run Multiline Script`, `Caches`, and `Runners`. The main area shows a table of 7 workflow runs. The first four runs are green, indicating success, and are labeled "added other workflows". The fifth and sixth runs are red, indicating failure, and are labeled "updated". The seventh run is also red and labeled "Merge branch 'main' of https://github.com/xshreya/GithubAct". Each run entry includes the workflow name, the commit hash, the branch, the status, the time taken, and a link to view the run details.

Event	Status	Branch	Actor
added other workflows	Success	main	xshreya
added other workflows	Success	main	xshreya
added other workflows	Success	main	xshreya
added other workflows	Success	main	xshreya
updated	Failure	main	xshreya
updated	Failure	main	xshreya
Merge branch 'main' of https://github.com/xshreya/GithubAct	Failure	main	xshreya

# Output of python.yml

The screenshot shows the details of a specific workflow run for the `python.yml` file. The run is labeled "updated #3" and is in a "succeeded" state. The left sidebar shows the "Summary" tab selected, with a "build" job listed. The main area displays the "build" job details, including a list of steps and their durations. The steps are: "Set up job" (1s), "Checkout Repository" (8s), "Set up Python" (8s), "Run tests" (8s), "Post Set up Python" (8s), "Post Checkout Repository" (8s), and "Complete job" (8s). The "Run tests" step is expanded, showing the command `python try.py` and the output `SHREYA`.

Step	Duration
Set up job	1s
Checkout Repository	8s
Set up Python	8s
Run tests	8s
Post Set up Python	8s
Post Checkout Repository	8s
Complete job	8s

# Output of javascript.yml

This screenshot shows the GitHub Actions interface for a workflow named 'Run JavaScript Script' in the repository 'xshreya / GithubAct'. The workflow is in a 'completed' state, indicated by a green checkmark and the text 'added other workflows #1'. The job 'run-script' is also completed, having succeeded 14 minutes ago. The workflow file is located at '.github/workflows/javascript.yml'. The job steps are listed as follows:

- Set up job (0s)
- Checkout repository (1s)
- Install Node.js (3s)
- Run JavaScript Script (0s)
- Post Install Node.js (0s)
- Post Checkout repository (0s)
- Complete job (0s)

The 'Run JavaScript Script' step contains the following multiline script:

```
1 ▶ Run node -e '  
9 This is a multiline JavaScript script  
10 You can write multiple lines of JavaScript code here  
11 For example, you can perform computations, manipulate data, etc.
```

# Output of multiline.yml

This screenshot shows the GitHub Actions interface for a workflow named 'Run Multiline Script' in the repository 'xshreya / GithubAct'. The workflow is in a 'completed' state, indicated by a green checkmark and the text 'added other workflows #1'. The job 'run-script' is also completed, having succeeded 16 minutes ago. The workflow file is located at '.github/workflows/multiline.yml'. The job steps are listed as follows:

- Set up job (0s)
- Checkout repository (1s)
- Run Multiline Script (0s)
- Post Checkout repository (0s)
- Complete job (1s)

The 'Run Multiline Script' step contains the following multiline script:

```
1 ▶ Run echo "This is a multiline script"  
8 This is a multiline script  
9 It can contain multiple commands  
10 For example, you can run any bash commands here  
11 You can also include variables or conditionals
```



# Output of yetanother.yml

This screenshot shows the GitHub Actions workflow run summary for the workflow named 'yetanother.yml'. The workflow was triggered by a push event and completed successfully. The summary includes a table of jobs and a list of annotations.

**Workflow Summary:**

Job	Status	Duration
test (12.x)	Success	9s
test (14.x)	Success	10s
test (16.x)	Success	6s

**Annotations:**

- test (16.x): Node.js 16 actions are deprecated. Please update the following actions to use Node.js 20: actions/checkout@v2, actions/setup-node@v2. For more information see: <https://github.blog/changelog/2023-09-22-github-actions-tr...>
- test (16.x): The following actions uses node12 which is deprecated and will be forced to run on node16: actions/checkout@v2, actions/setup-node@v2. For more info: <https://github.blog/changelog/2023-06-13-github-actions-all-actions-w...>

This screenshot shows the detailed view of the 'test (12.x)' job from the previous workflow run. The job succeeded and the logs show the steps executed.

**Job Details:**

- Set up job**: 1s
- Checkout repository**: 0s
- Set up Node.js 12.x**: 3s
- Display Environment Variables**: 0s
  - Run echo "Environment variable 1: \$ENV\_VAR\_1"
  - Environment variable 1: Value 1
  - Environment variable 2: Value 2
  - Matrix node version: 12.x
- Post Set up Node.js 12.x**: 0s
  - Post job cleanup.
- Post Checkout repository**: 0s
- Complete job**: 0s

Untitled document - Google D... x ChatGPT x Understanding GitHub Actions x added other workflows - xshreya x +

github.com/xshreya/GithubAct/actions/runs/8293358190/job/22696291370

github.com/xshreya / GithubAct

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Environment Variables and Matrix Demo

added other workflows #1

Summary

Jobs

test (12.x)

test (14.x)

test (16.x)

Run details

Usage

Workflow file

test (12.x)

succeeded 17 minutes ago in 9s

Beta Give feedback Search logs

Set up job 1s

Checkout repository 8s

Set up Node.js 12.x 3s

Display Environment Variables 8s

1 Run echo "Environment variable 1: \$ENV\_VAR\_1"

9 Environment variable 1: Value 1

10 Environment variable 2: Value 2

11 Matrix node version: 12.x

Post Set up Node.js 12.x 8s

1 Post job cleanup.

Post Checkout repository 8s

Complete job 8s

Type here to search 29°C Sunny 14:17 15-03-2024

Untitled document - Google D... x ChatGPT x Understanding GitHub Actions x added other workflows - xshreya x +

github.com/xshreya/GithubAct/actions/runs/8293358190/job/22696291370

github.com/xshreya / GithubAct

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Environment Variables and Matrix Demo

added other workflows #1

Summary

Jobs

test (12.x)

test (14.x)

test (16.x)

Run details

Usage

Workflow file

test (12.x)

succeeded 17 minutes ago in 9s

Beta Give feedback Search logs

Set up job 1s

Checkout repository 8s

Set up Node.js 12.x 3s

Display Environment Variables 8s

1 Run echo "Environment variable 1: \$ENV\_VAR\_1"

9 Environment variable 1: Value 1

10 Environment variable 2: Value 2

11 Matrix node version: 12.x

Post Set up Node.js 12.x 8s

1 Post job cleanup.

Post Checkout repository 8s

Complete job 8s

Type here to search 29°C Sunny 14:17 15-03-2024