

CMSC388T

# Working With DevOps

# Contents

1

## DevOps

Understanding DevOps and other Software Development and Lifecycle practices.

2

## Using GitHub Actions

Configuring GitHub Actions with existing repos

3

## CI/CD With GitHub Actions

Adding CI/CD pipelines to a Github Repo



# 1. DevOps

---

Understanding DevOps and other  
Software Development and  
Lifecycle practices.

# The Heavy Definition

**“DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.”**

—Amazon Web Services

Let’s Break This Down

# A Brief History: The Waterfall Model

---

- A Plan-Driven and Linear Approach
- All steps must be planned and scheduled in advance
- Each phase in the software development lifecycle should not start until the previous stage has been completed

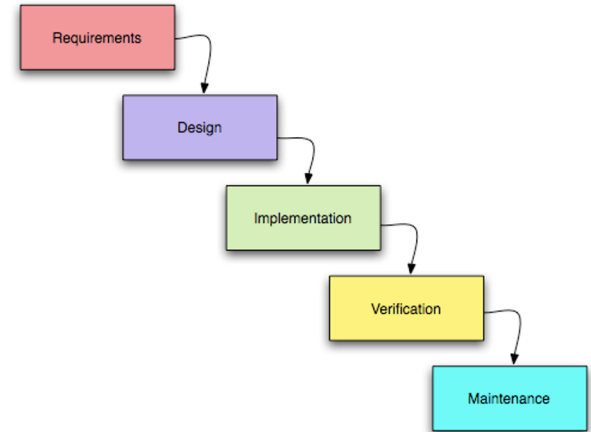


Image Source: [umsl.edu](https://umsl.edu)

# A Brief History: Agile Development

---

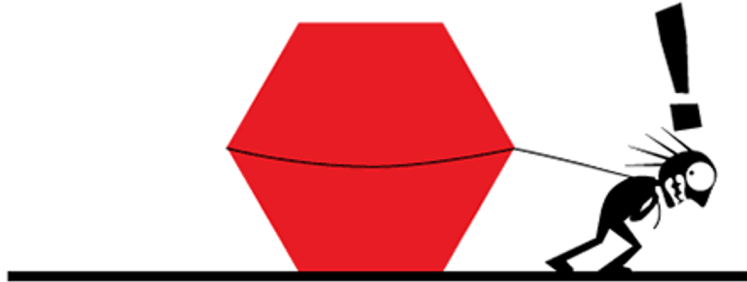
- A Feedback Driven Cyclic Approach
- Emphasizes continuous feedback from end users
- Focus on short development cycles that deliver incremental updates
- Capitalizes on Continuous Integration and Continuous Deployment



Image Source: [mlsdev.com](https://mlsdev.com)

# Waterfall vs Agile

## THE WATERFALL PROCESS



*'This project has got so big,  
I'm not sure I'll be able to deliver it!'*

## THE AGILE PROCESS



*'It's so much better delivering this  
project in bite-sized sections'*

Image Source: [meddigital.com](https://www.meddigital.com)

# A Brief History: Enterprise Management Systems

- Help teams manage IT infrastructure and applications
- Focus on optimizing the delivery of IT services
- Useful for managing and monitoring complex enterprise-scale applications

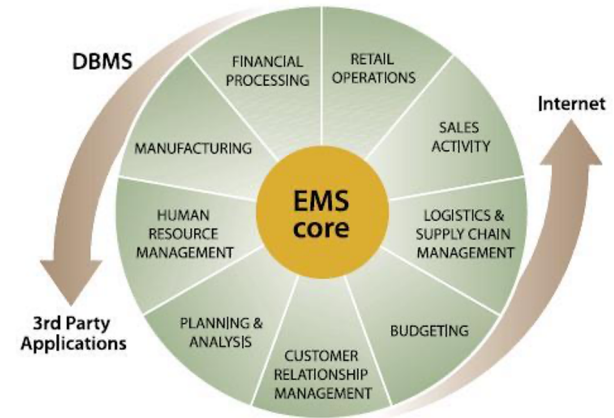


Image Source: [thegeek.com](http://thegeek.com)

# What is DevOps?

---

- Integrates all parties involved with software development and deployment into a single workflow
- Emphasizes that Developers and IT Operations work together
- Focuses on rapid delivery, high quality, and reliability
- Emphasizes the use of automation
- Extends Agile principles beyond code to the entire software development process

# Key Features of DevOps

---

- Collaboration
- Automation
- Continuous Integration
- Continuous Testing
- Continuous Deployment
- Rapid Remediation

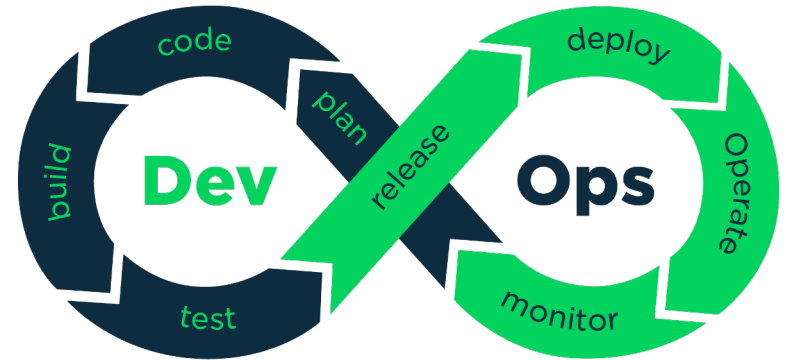


Image Source: [medium.com](https://medium.com)

# Industry Tools

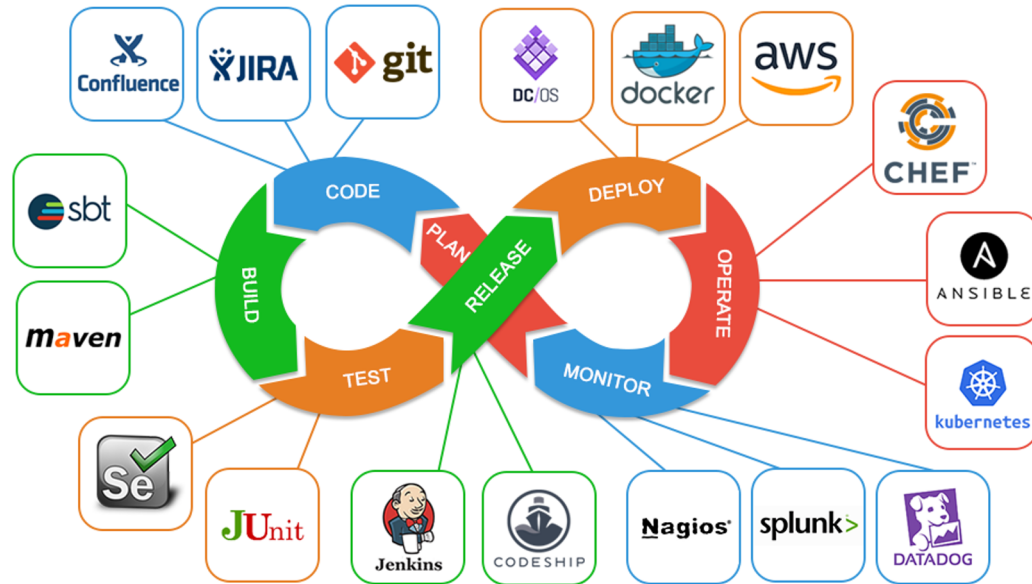


Image Source: [medium.com](https://medium.com)

# Clicker Quiz

Which of the following coding practices/methodologies places an emphasis on planning before starting coding ?

- a) Waterfall
- b) Agile
- c) Enterprise Management Systems
- d) DevOps
- e) all of the above

# Clicker Quiz

Which of the following coding practices/methodologies places an emphasis on planning before starting coding ?

- a) **Waterfall**
- b) Agile
- c) Enterprise Management Systems
- d) DevOps
- e) all of the above



## 2. Using Github Actions

---

Configuring GitHub Actions with  
existing repos

# What is GitHub Actions

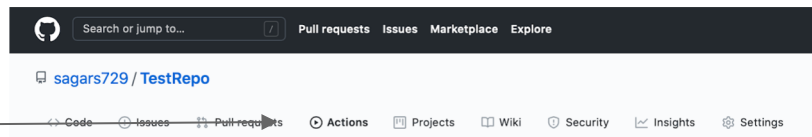
---

- Executes code when changes are made to a GitHub repository
- Used to integrate CI/CD pipelines (also known as Workflows)
- Fully Automated



# Using Github Actions

Enable Actions For Your Repository



## Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow template to get started.

Skip this and [set up a workflow yourself](#) →

### Workflows made for your repository Suggested

#### Simple workflow

By GitHub Actions

Start with a file with the minimum necessary structure.

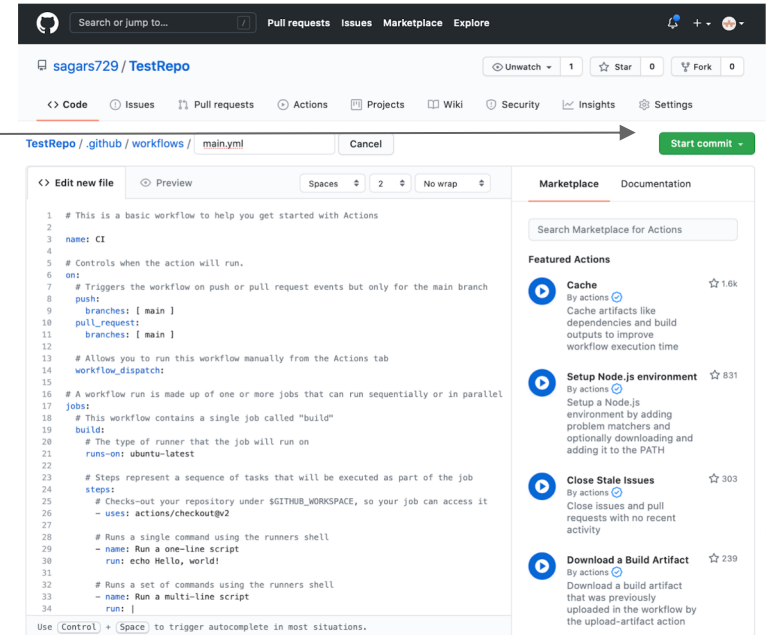
[Set up this workflow](#)

```
echo Hello, world!  
echo Add other actions to build,  
echo test, and deploy your project.
```

[actions/starter-workflows](#)

# Add Default Action

Commit The main.yaml file



The screenshot shows the GitHub Actions workflow editor for a repository named 'TestRepo'. The main editor displays a workflow file with the following content:

```
1 # This is a basic workflow to help you get started with Actions
2
3 name: CI
4
5 # Controls when the action will run.
6 on:
7   # Triggers the workflow on push or pull request events but only for the main branch
8   push:
9     branches: [ main ]
10  pull_request:
11    branches: [ main ]
12
13 # Allows you to run this workflow manually from the Actions tab
14 workflow_dispatch:
15
16 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
17 jobs:
18   # This workflow contains a single job called "build"
19   build:
20     # The type of runner that the job will run on
21     runs-on: ubuntu-latest
22
23     # Steps represent a sequence of tasks that will be executed as part of the job
24     steps:
25       # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
26       - uses: actions/checkout@v2
27
28       # Runs a single command using the runners shell
29       - name: Run a one-line script
30         run: echo Hello, world!
31
32       # Runs a set of commands using the runners shell
33       - name: Run a multi-line script
34         run: |
35
36 Use Control + Space to trigger autocomplete in most situations.
```

The right sidebar shows the 'Marketplace' tab with a search bar and a list of featured actions:

- Cache** (1.6k stars): Cache artifacts like dependencies and build outputs to improve workflow execution time.
- Setup Node.js environment** (831 stars): Setup a Node.js environment by adding problem matchers and optionally downloading and adding it to the PATH.
- Close Stale Issues** (303 stars): Close issues and pull requests with no recent activity.
- Download a Build Artifact** (239 stars): Download a build artifact that was previously uploaded in the workflow by the upload-artifact action.

# Viewing The Workflow Status

Actions Tab

Green Check Mark - Success  
Commit Message

The screenshot shows the GitHub Actions interface for the repository 'sagars729 / TestRepo'. The 'Actions' tab is selected, displaying a list of workflows. A green checkmark indicates a successful workflow run named 'Create main.yml'.

**Workflows**

- [New workflow](#)
- [All workflows](#)

**All workflows**

Filter workflows

1 result	Event	Status	Branch	Actor
<b>Create main.yml</b> CI #1: Commit f046599 pushed by sagars729			<a href="#">main</a>	1 minute ago 19s

© 2020 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

# Making Changes

New Commit Status

New Commit Hash

The screenshot displays the GitHub interface for the repository 'sagars729 / TestRepo'. The 'Code' tab is selected, showing a file list with the following details:

File	Commit	Time
sagars729: Create main.yml	f046599	8 minutes ago
.github/workflows	Create main.yml	8 minutes ago
README.md	first commit	18 minutes ago

Annotations on the left side of the image point to the commit status and hash:

- A horizontal line points from the text 'New Commit Status' to the green checkmark icon next to the commit 'sagars729: Create main.yml'.
- A horizontal line points from the text 'New Commit Hash' to the commit hash 'f046599'.

The commit details for 'sagars729: Create main.yml' show a green checkmark, the hash 'f046599', the time '8 minutes ago', and '2 commits'.

The README.md content is visible below the file list, showing the title 'TestRepo'.

# Clicker Quiz

Where can you check the status of a workflow?

- a) The Workflows Tab On GitHub
- b) Next to the commit hash on GitHub
- c) The git workflow status command
- d) All of the above

# Clicker Quiz

Where can you check the status of a workflow?

- a) The Workflows Tab On GitHub
- b) Next to the commit hash on GitHub**
- c) The git workflow status command
- d) All of the above



## 2. CI/CD With GitHub

---

Adding CI/CD pipelines to a GitHub Repo

# Adding A Program

---

Before we get started with CI/CD Pipelines, let's first add a few files to our git TestRepo.

The first file we are adding is a simple Calculator class with only one method, add, which adds two integers

```
import java.util.*;
import java.io.*;

class Calculator {

    public static int add(int a, int b) {
        return a + b;
    }

}
```

Calculator.java

# Add a main.yaml File

The next file we need to add is the main.yml file.

The main.yml file is a YAML file that configures our CI/CD pipeline and is located in the “.github/workflows” directory.

In this main.yaml file we specify a job “build” that compiles all of the java files.

```
name: CI

on: [push, pull_request, workflow_dispatch]

jobs:
  build:
    runs-on: ubuntu-latest
    container: openjdk
    steps:
      - uses: actions/checkout@v2
      - name: Build Project
        run: |
          echo Hello World
          java -version
          javac -version
          javac *.java
```

main.yaml

# Add a main.yaml File

- **on** specifies when the jobs are run. It is set to run jobs any time changes are pushed, a pull\_request is merged, or it is run manually
- **jobs** details the jobs to run
  - **build** is the name of the job
  - **runs-on** specifies the OS
  - **container** details the docker image that will be used to run the code
  - **steps** are the actions taken in the job
  - **run** lists the commands to run on the CLI

```
name: CI

on: [push, pull_request, workflow_dispatch]

jobs:
  build:
    runs-on: ubuntu-latest
    container: openjdk
    steps:
      - uses: actions/checkout@v2
      - name: Build Project
        run: |
          echo Hello World
          java -version
          javac -version
          javac *.java
```

main.yaml

# Checking Our Pipelines

Status Of Our Pipeline  
Click To View More Details

The screenshot shows the GitHub Actions interface for the repository `sagars729/TestRepo`. The `Actions` tab is selected, displaying the `Workflows` section. A `New workflow` button and an `All workflows` button are visible. A search bar for `Filter workflows` is present. Below, a table lists `2 results` of workflows:

	Event	Status	Branch	Actor
✓ <b>Add Compilation Test</b> CI #2: Commit 62cc767 pushed by sagars729	main	Completed	main	35 seconds ago ... 32s
✓ <b>Create main.yml</b> CI #1: Commit f046599 pushed by sagars729	main	Completed	main	2 hours ago ... 19s

At the bottom of the page, the footer contains copyright information and links: © 2020 GitHub, Inc., Terms, Privacy, Security, Status, Help, Contact GitHub, Pricing, API, Training, Blog, About.

# Viewing More Details

Commit Message

Status

Workflow File

Jobs

The screenshot displays the GitHub Actions interface for the repository `sagars729 / TestRepo`. The `Actions` tab is selected, showing a workflow named `Add Compilation Test CI #2`. The workflow status is `Success`, triggered by a push to the `main` branch. The summary shows a total duration of `32s`. The workflow file `main.yml` is shown with a job named `build` that has a duration of `21s`. The annotations section shows a warning about Ubuntu-latest workflows using Ubuntu-20.04 soon.

Commit Message

Status

Workflow File

Jobs

# Viewing Job Logs

Job Name  
Jobs Duration  
Jobs Status  
Jobs Tab  
Steps  
Step Logs

The screenshot displays the GitHub Actions interface for a repository named 'sagars729 / TestRepo'. The 'Actions' tab is selected, showing a workflow named 'Add Compilation Test CI #2' with a green checkmark indicating success. The 'Jobs' section shows a single job named 'build' with a green checkmark. The 'Steps' section for the 'build' job is expanded, showing a list of steps with their durations. The 'Build Project' step is selected, and its logs are displayed in a dark-themed panel on the right. The logs show the execution of various commands, including setting up the environment, running tests, and completing the job.

Jobs

- build

Steps

- Set up job 3s
- Initialize containers 15s
- Run actions/checkout@v2 1s
- Build Project 1s
  - Run echo Hello World
  - Hello World
  - openjdk version "15.0.1" 2020-10-20
  - OpenJDK Runtime Environment (build 15.0.1+9-18)
  - OpenJDK 64-Bit Server VM (build 15.0.1+9-18, mixed mode, sharing)
  - javac 15.0.1
- Post Run actions/checkout@v2 1s
- Stop containers 0s
- Complete job 0s

Step Logs

```
1  ▶ Run echo Hello World
7  Hello World
8  openjdk version "15.0.1" 2020-10-20
9  OpenJDK Runtime Environment (build 15.0.1+9-18)
10 OpenJDK 64-Bit Server VM (build 15.0.1+9-18, mixed mode, sharing)
11 javac 15.0.1
```

# Modifying Our Program

---

Let's modify our program to include a subtraction method. Instead of returning the difference however, let's "make a mistake" and return the sum.

```
import java.util.*;
import java.io.*;

class Calculator {

    public static int add(int a, int b) {
        return a + b;
    }

    public static int sub(int a, int b) {
        return a + b;
    }

}
```

Calculator.java

# Add Test Files

---

Let's also include two test files  
TestAdd.java and TestSub.java that test our  
Calculator Class.

```
import junit.framework.*;
public class TestAdd extends TestCase {

    public void testAdd(){
        int sum = Calculator.add(2,4);
        assertTrue(sum == 6);
    }
}
```

TestAdd.java

```
import junit.framework.*;
public class TestSub extends TestCase {

    public void testSub(){
        int sub = Calculator.sub(2,4);
        assertTrue(sub == -2);
    }
}
```

TestSub.java

# Adding Tests To main.yml File

Now that we have created our tests, we can add them to the main.yml.

We add two jobs, calcadd and calcsb that run each test.

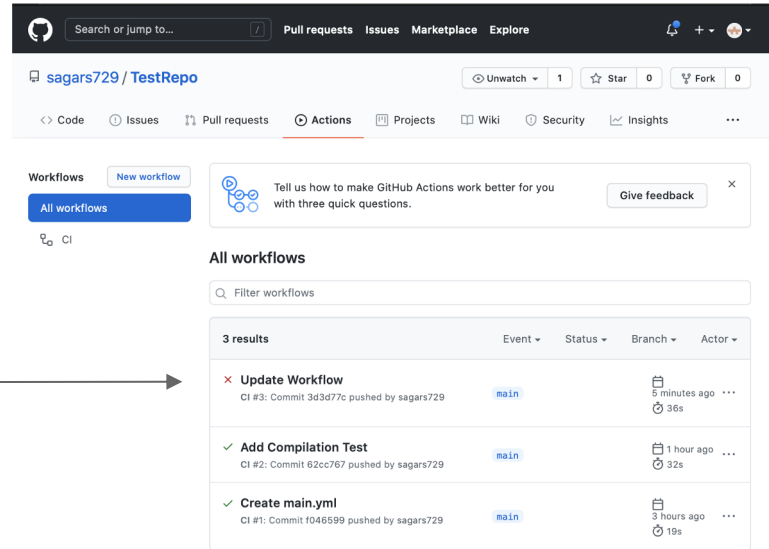
The junit-4.10.jar file has also been added to our repository to allow us to run JUnit tests.

```
...
jobs:
  build:
    ...
  calcadd:
    runs-on: ubuntu-latest
    container: openjdk
    steps:
      - uses: actions/checkout@v2
      - name: Test Calculator Add
        run: |
          javac -cp "junit-4.10.jar:." *.java
          java -cp "junit-4.10.jar:."
org.junit.runner.JUnitCore TestAdd

  calcsb:
    runs-on: ubuntu-latest
    container: openjdk
    steps:
      - uses: actions/checkout@v2
      - name: Test Calculator Sub
        run: |
          javac -cp "junit-4.10.jar:." *.java
          java -cp "junit-4.10.jar:."
org.junit.runner.JUnitCore TestSub
```

main.yml

# Viewing Pipeline Status



The screenshot shows the GitHub Actions interface for the repository `sagars729 / TestRepo`. The `Actions` tab is selected, displaying a list of workflows. The workflow `Update Workflow` is highlighted, showing a failed status (red X) for the commit `3d3d77c` pushed by `sagars729` on the `main` branch. The failure occurred 5 minutes ago and took 36 seconds. Below it, two other workflows, `Add Compilation Test` and `Create main.yml`, are shown with successful statuses (green checkmarks).

Workflow	Event	Status	Branch	Actor	Time
Update Workflow	CI #3: Commit 3d3d77c pushed by sagars729	Failed	main	sagars729	5 minutes ago, 36s
Add Compilation Test	CI #2: Commit 62cc767 pushed by sagars729	Success	main	sagars729	1 hour ago, 32s
Create main.yml	CI #1: Commit f046599 pushed by sagars729	Success	main	sagars729	3 hours ago, 19s

As Expected Our Build Has Failed

# Viewing More Details

The screenshot shows the GitHub Actions interface for a workflow named 'Update Workflow CI #3'. The workflow was triggered by a push from 'sagars729' to the 'main' branch. The overall status is 'Failure' with a total duration of 36 seconds. The workflow consists of three jobs: 'build' (passed, 21s), 'calcadd' (passed, 22s), and 'calcsb' (failed, 22s). The 'main.yml' file content is shown, indicating the workflow is triggered on push. The 'Jobs' section on the left lists the jobs with their status icons: a green checkmark for 'build', a green checkmark for 'calcadd', and a red X for 'calcsb'. Arrows from the text on the left point to the 'calcadd' and 'calcsb' entries in the job list.

Jobs	Status	Duration
build	Passed	21s
calcadd	Passed	22s
calcsb	Failed	22s

The calcadd job has passed  
The calcsb job has failed

# Viewing Job Logs

The calsub job failed due to an  
assertion error

The screenshot displays the GitHub Actions interface for a workflow named 'Update Workflow CI #3'. On the left, a sidebar lists the jobs: 'build', 'calcadd', and 'calsub'. The 'calsub' job is highlighted with a red status icon. An arrow points from the text 'The calsub job failed due to an assertion error' to the 'calsub' job in the sidebar. The main panel shows the logs for the 'calsub' job, which failed 9 minutes ago. The logs include the following content:

```
1 ▶ Run javac -cp "junit-4.10.jar;" *.java
5 JUnit version 4.10
6 .E
7 Time: 0.005
8 There was 1 failure:
9 1) testSub(TestSub)
10 junit.framework.AssertionFailedError
```

# Clicker Quiz

Which of the following are **required** for creating a job in the main.yml file?

- a) A test file to run
- b) A container for the underlying software
- c) The branches that trigger jobs
- d) None of the above are required

# Clicker Quiz

Which of the following are **required** for creating a job in the main.yml file?

- a) A test file to run
- b) A container for the underlying software
- c) The branches that trigger jobs
- d) None of the above are required**