CMSC388T

Working With DevOps

Contents

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

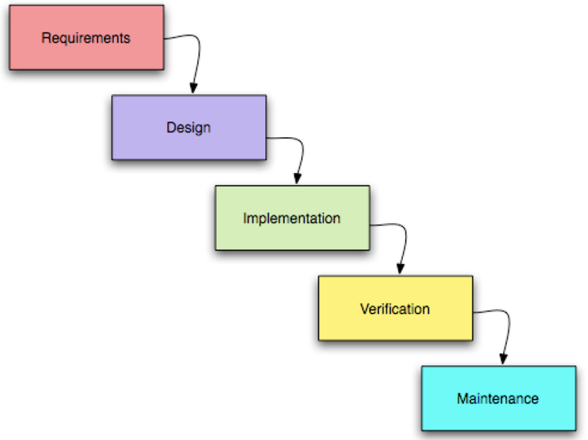


Image Source: umsl.edu

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

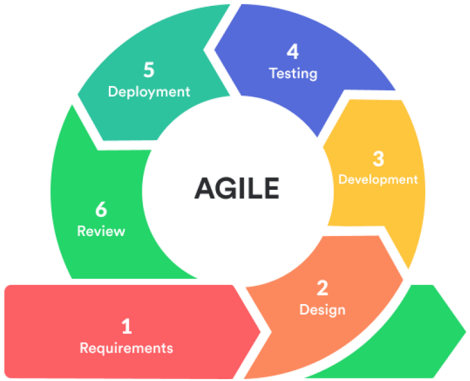


Image Source: mlsdev.com

Waterfall vs Agile

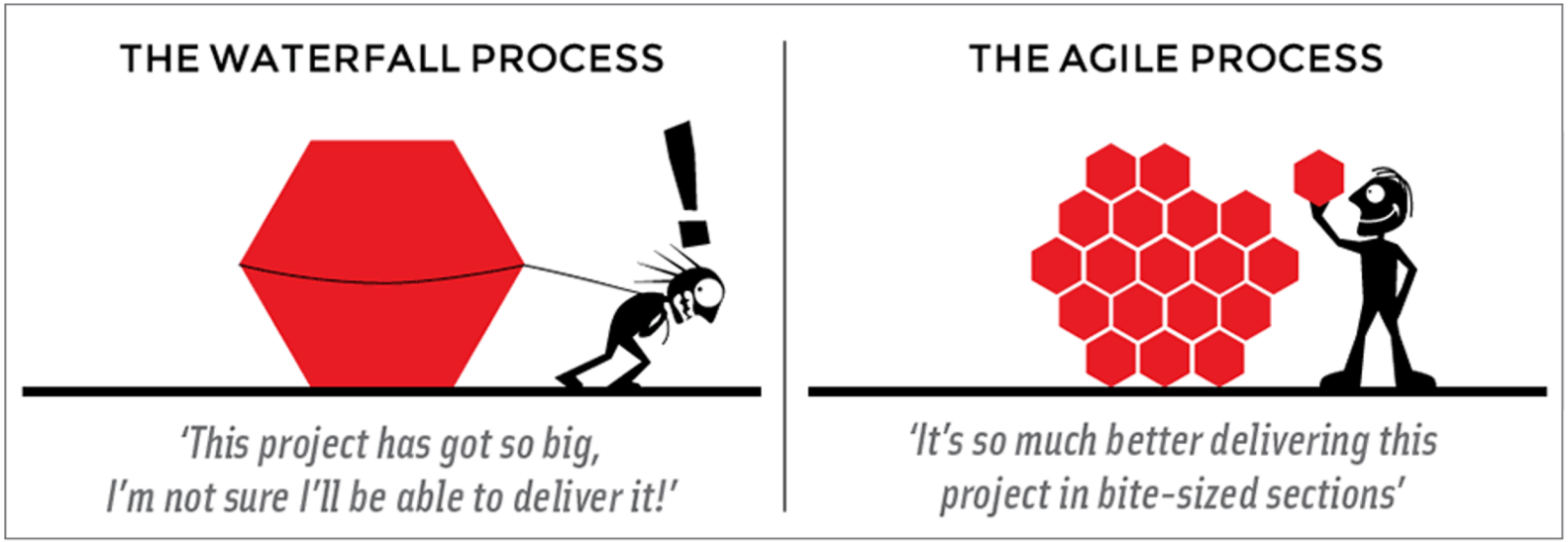


Image Source: meddigital.com

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

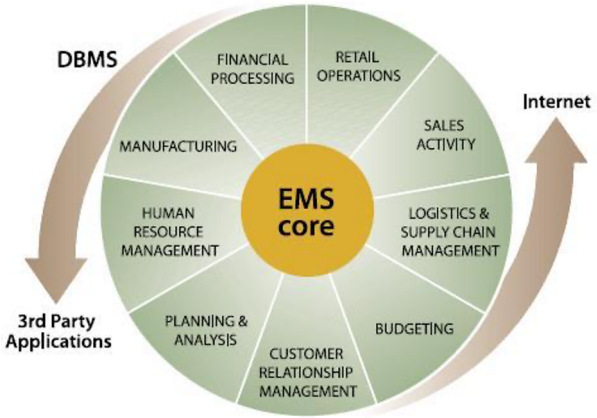


Image Source: thegeek.com

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

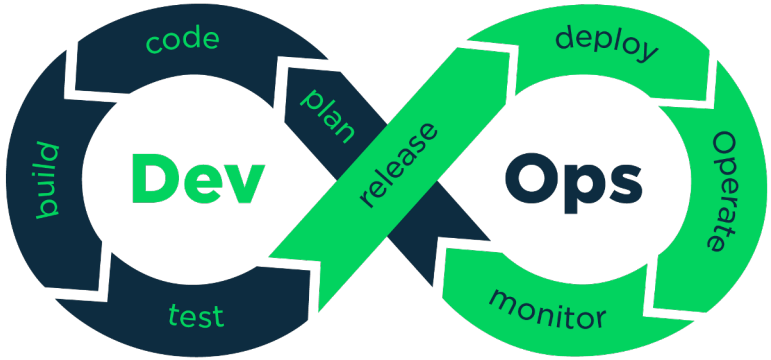


Image Source: medium.com

Industry Tools

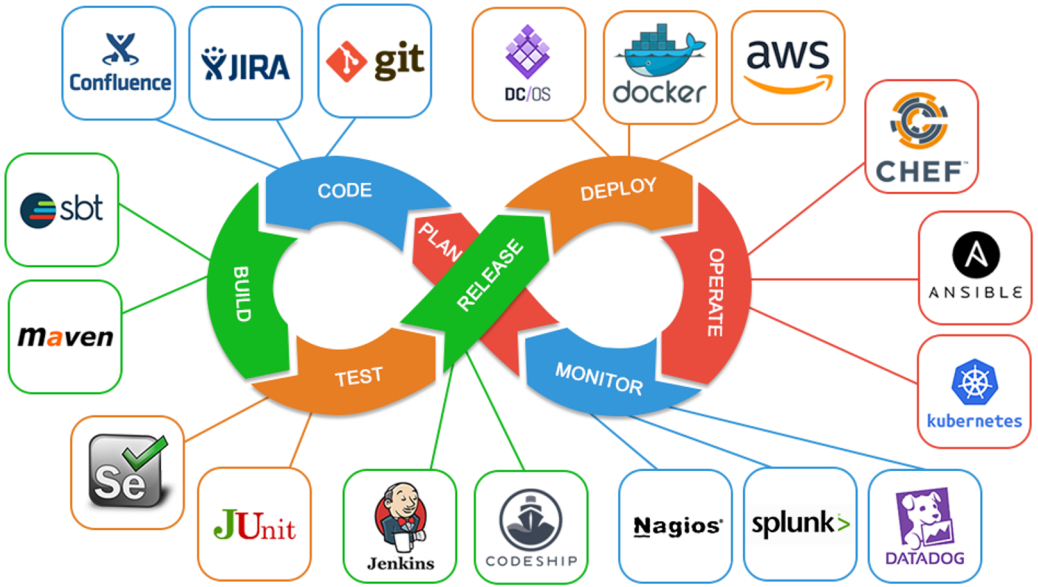


Image Source: medium.com

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

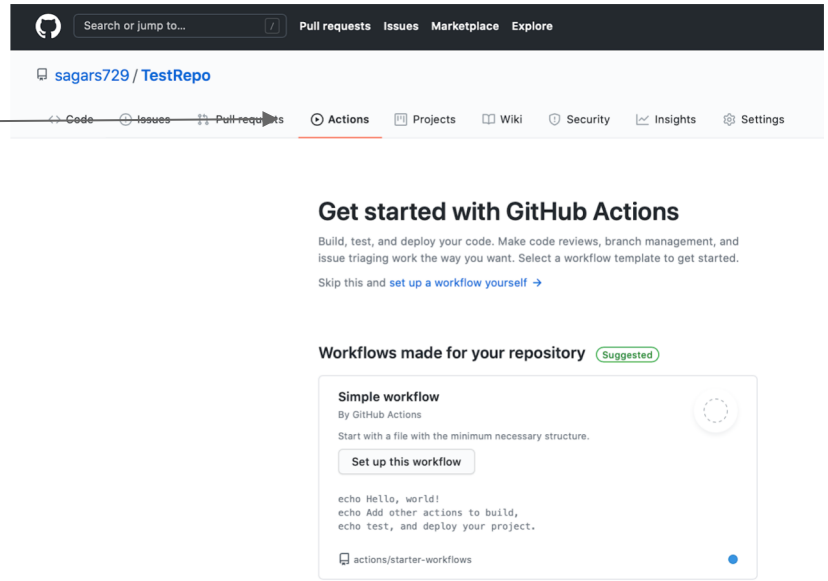
Configuring GitHub Actions with existing repos

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

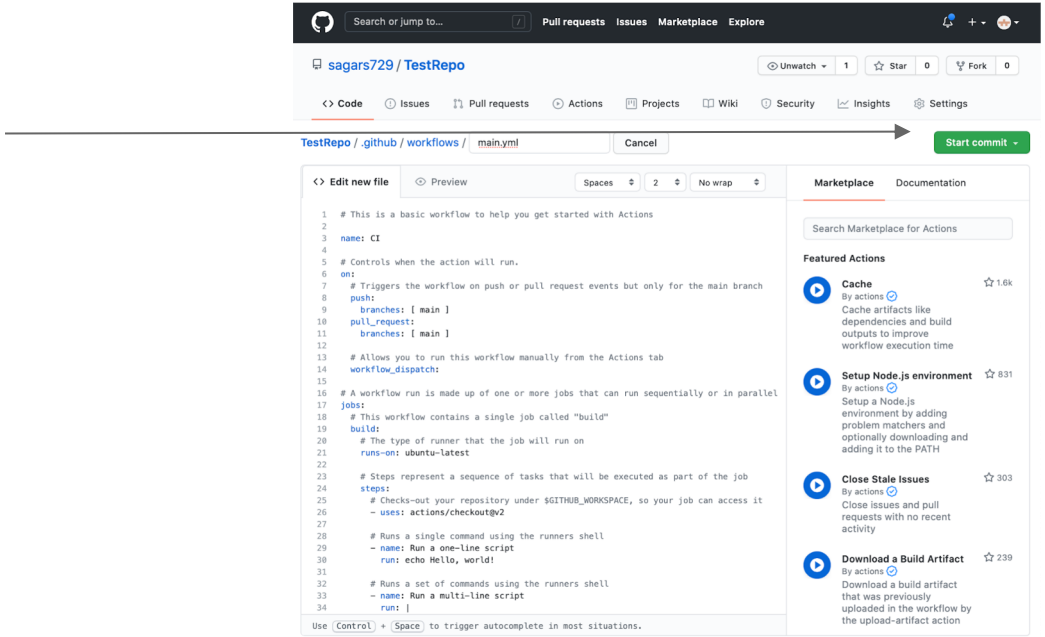
Using Github Actions

Enable Actions For Your Repository



Add Default Action

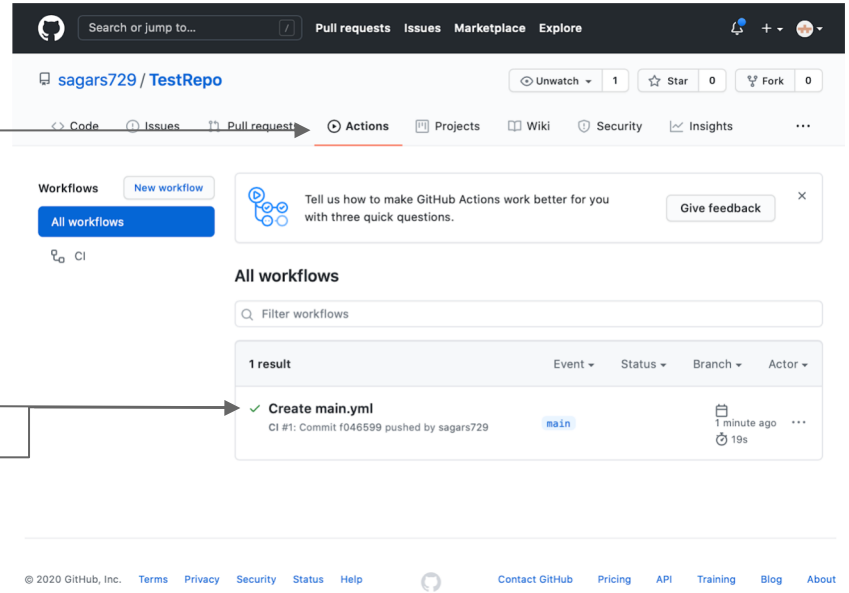
Commit The main.yaml file



Viewing The Workflow Status

Actions Tab

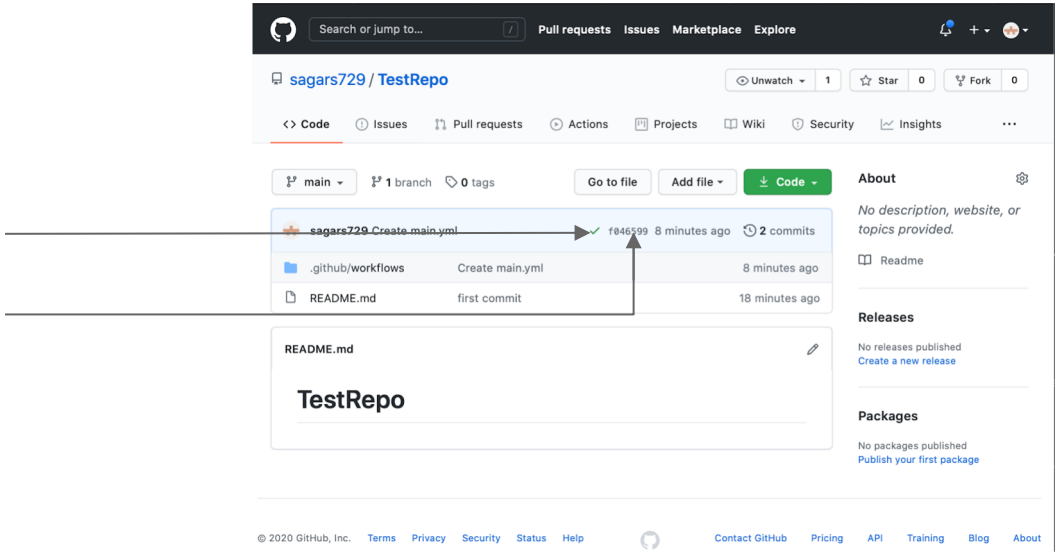
Green Check Mark - Success
Commit Message



Making Changes

New Commit Status

New Commit Hash



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.

```yaml
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

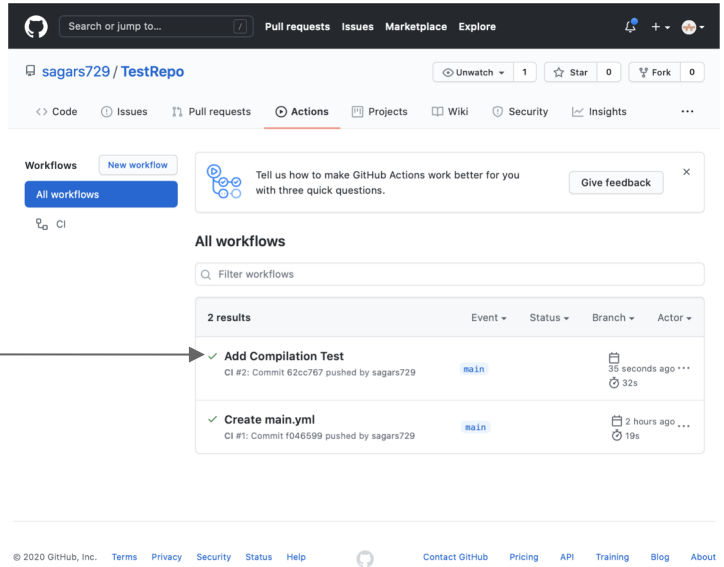
- **on** specifies when the jobs are run. It is set to run jobs any time changes are pushed, a pull_request is merged, or it is run manually
- **jobs** details the jobs to run
  - **build** is the name of the job
  - **runs-on** specifies the OS
  - **container** details the docker image that will be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

```yaml
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



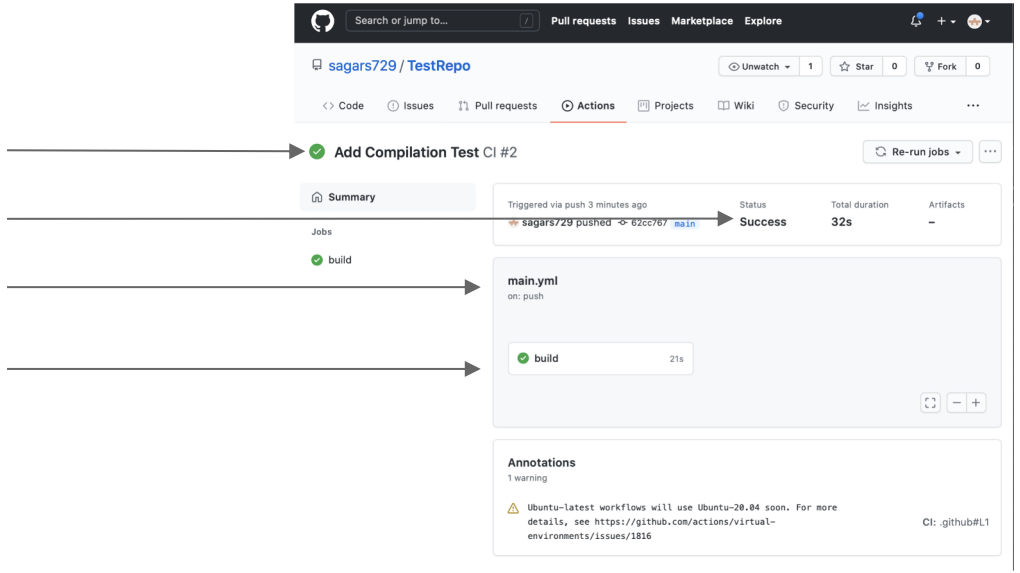
Viewing More Details

Commit Message

Status

Workflow File

Jobs



Viewing Job Logs

Job Name

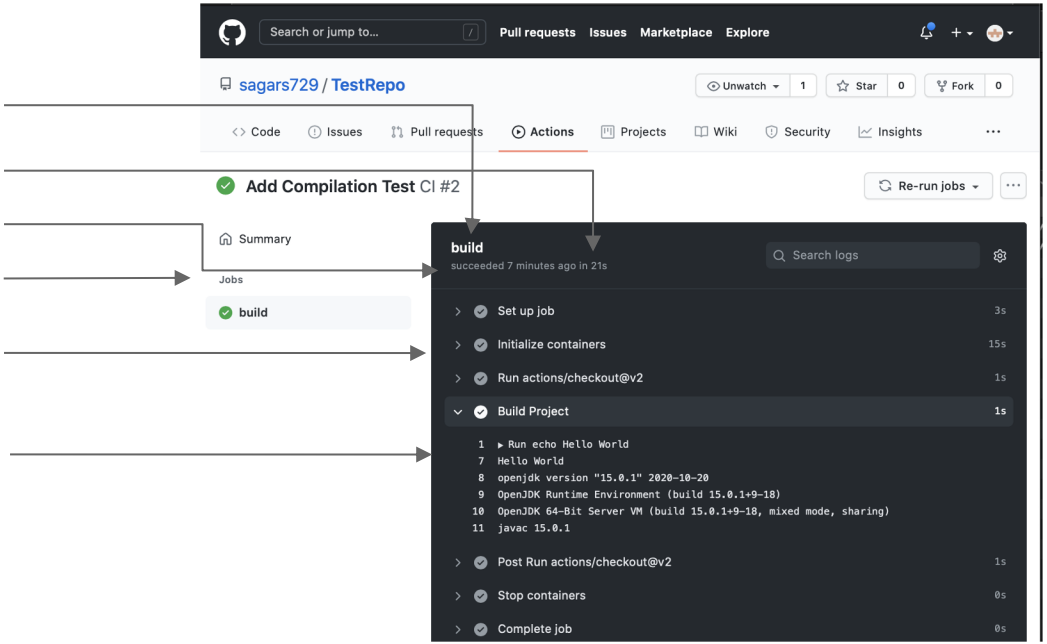
Jobs Duration

Jobs Status

Jobs Tab

Steps

Step Logs



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.

```java
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.

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

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

TestAdd.java

```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.yaml.

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

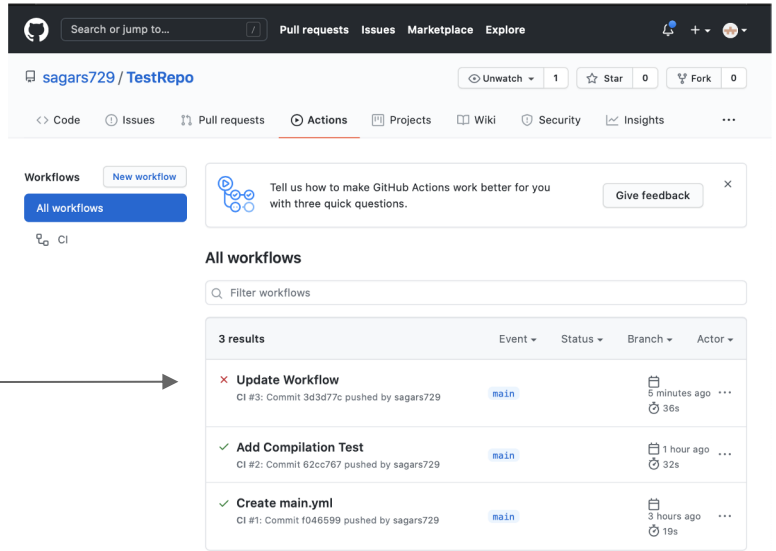
The junit-4.10.jar file has also been added to our repository to allow us to run JUnit tests.

```
...
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

  calcsub:
    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

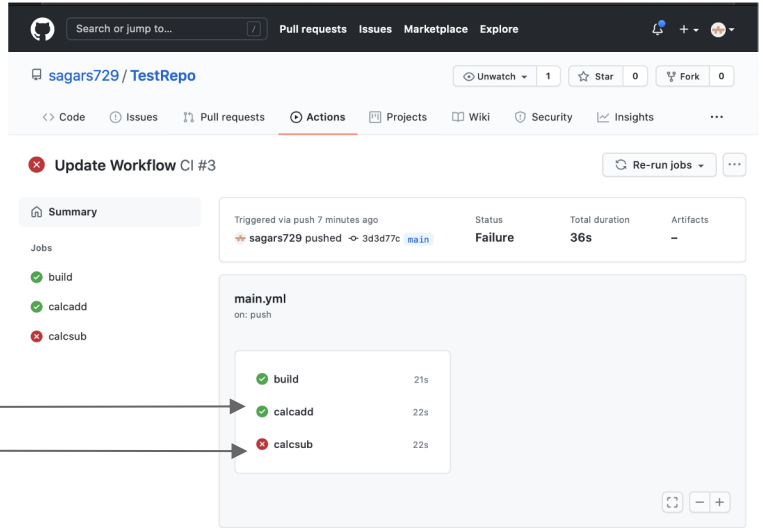
As Expected Our Build Has Failed



Viewing More Details

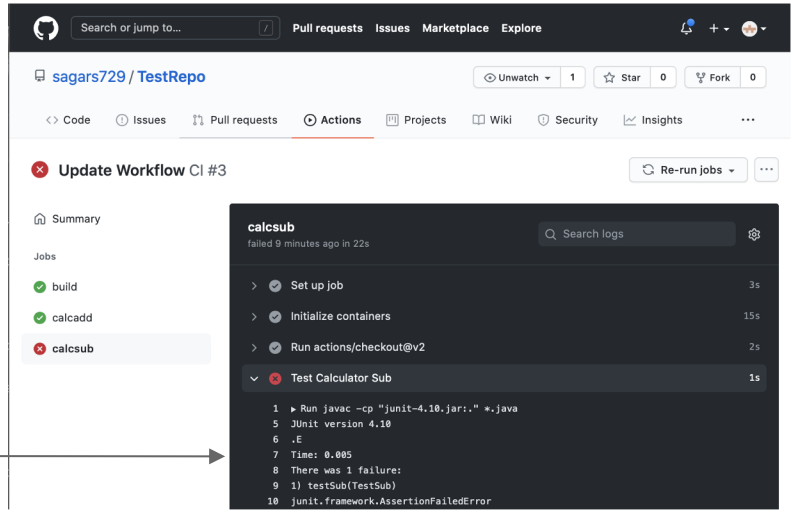
The calcadd job has passed

The calcsub job has failed



Viewing Job Logs

The calcsub job failed due to an assertion error



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