Agile: “Delivering value faster”

- ➔ Scrum (agile framework): produce product increments in the form of sprints (2 weeks)
- ➔ After each sprint, the software is in a fully deployable state for the customer to use
- ➔ Customer immediately receives value, company immediately receives value

This still leaves the question: how do we approach “delivering” this value (software)?

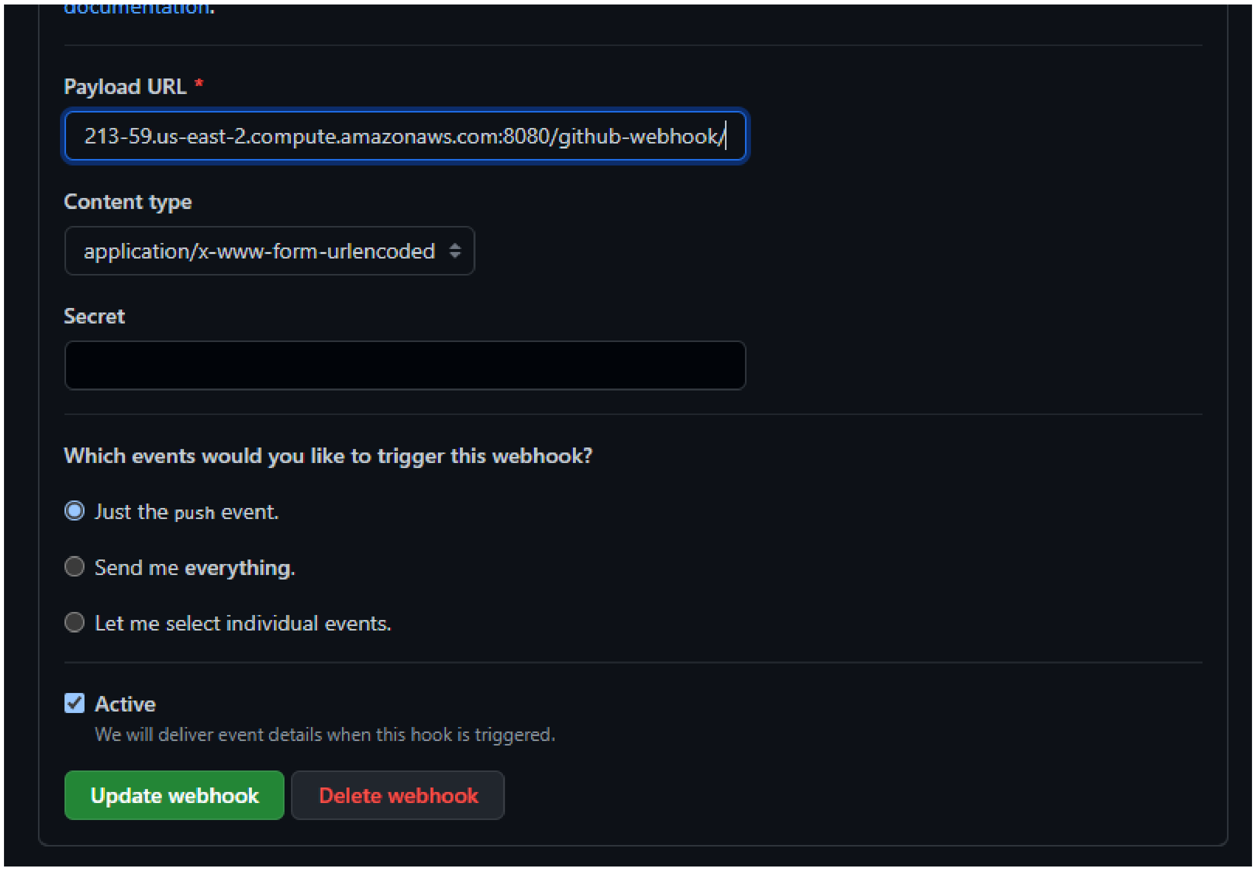
DevOps: The combination of development and operations into one team

- ➔ Nowadays we have tools known as CI/CD tools
- ➔ CI/CD = Continuous Integration / Continuous Delivery + Continuous Deployment

- **Continuous Integration**: merging code together frequently
- **Continuous Delivery**: ensuring that the code that has been merged together is in a “deliverable” state (so that we could deploy at a moments notice)
    - **Code in the “main” or “master” branch should always be in a deliverable state**
        - **The main branch is the “ground truth” of the entire project**
        - **Whenever you create a new feature branch, it should be branching off of main**
    - **The code that gets merged together should be passing all tests that were written (unit tests + integration tests, aka our whitebox tests)**
    - **build: compile the code**

        - **test: run the tests**
        - **Stable builds**
            - **Builds that pass all crucial tests**
        - **Unstable builds**
            - **Builds that fail many crucial tests**
    - Continuous Deployment: continuously deploy merged code **automatically**
        - **We can setup, on Github, webhook triggers**
        - **The purpose of a webhook is to have Github inform external services about any pushes that have been made to the Github repository**
        - **In this case, we set up a webhook to inform Jenkins about the changes to the main branch on**

Github, so that Jenkins can then initiate a build automatically



documentation.

Payload URL *

213-59.us-east-2.compute.amazonaws.com:8080/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

[✓ Just the push event.]

Send me everything.

Let me select individual events.

[✓ Active]

We will deliver event details when this hook is triggered.

GitHub Actions: A cloud based solution that GitHub provides for running various "workflows"

- ➔ An example of a workflow would be to compile the project using Maven (mvn) and also run all of the tests

- ➔ This gives us a visualization of whether the codebase as it currently is within our remote branch is stable or not

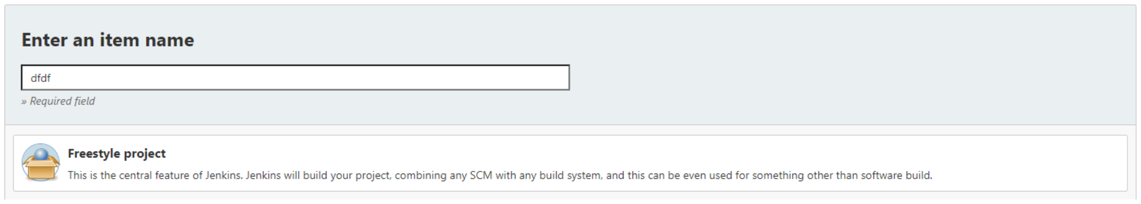
Pull Request: this is a feature on the GitHub website (it is not a terminology inherent to Git itself)

- ➔ We can create a pull request whereby changes between the main branch and the feature branch can be made. Any Github Actions workflows that are configured to run for pull requests into main will also execute.
- ➔ It allows us to review changes between our feature branches and the branch we want to merge into (likely main, for example) before we actually merge

➔ We can then actually merge the branch into the main branch assuming there are no merge conflicts

Jenkins: A CI/CD tool that is utilized for DevOps. It allows us to create "pipelines" whereby we have our source code as an input to the pipeline and the final deployment as an output

Create a new pipeline:


Enter an item name

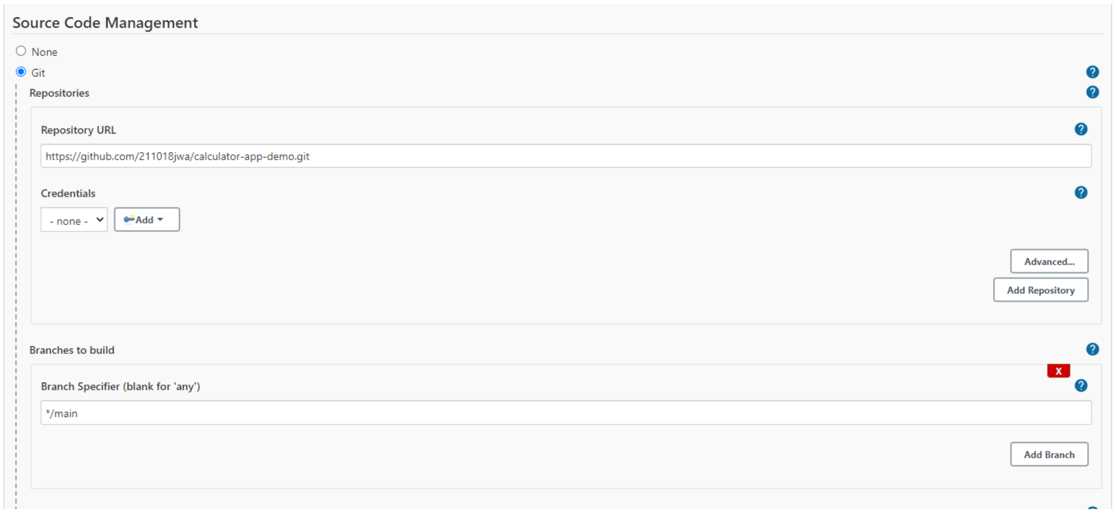
dfdf

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Configure pipeline:



Source Code Management

○ None
◉ Git

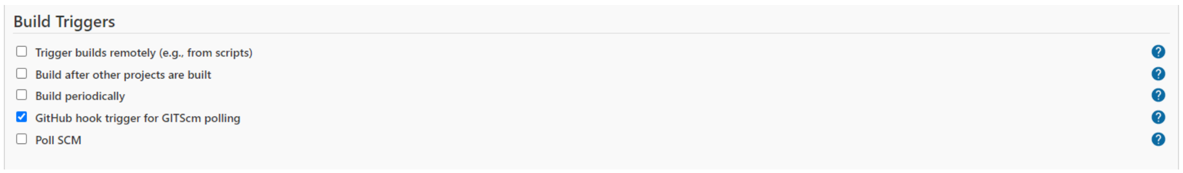
Repositories

Repository URL: https://github.com/211018jwa/calculator-app-demo.git

Credentials: - none -

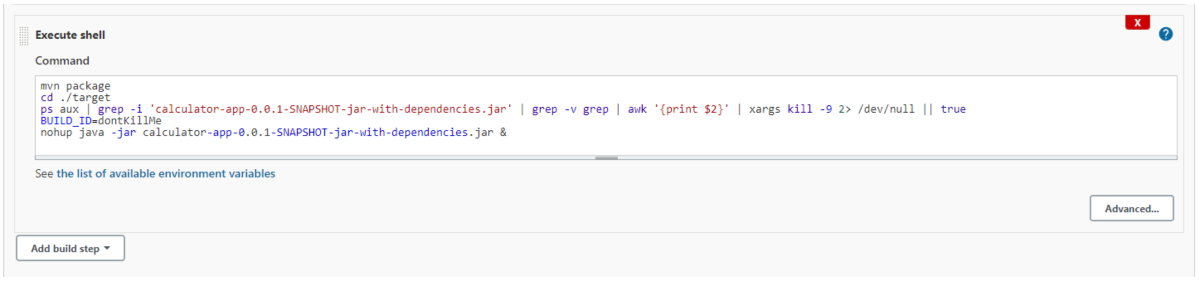
Branches to build

Branch Specifier (blank for 'any'): */main



Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☑ GitHub hook trigger for GITScm polling
- ☐ Poll SCM



Execute shell

Command

```
mvn package
cd ./target
ps aux | grep -i 'calculator-app-0.0.1-SNAPSHOT-jar-with-dependencies.jar' | grep -v grep | awk '{print $2}' | xargs kill -9 2> /dev/null || true
BUILD_ID=dontKillMe
nohup java -jar calculator-app-0.0.1-SNAPSHOT-jar-with-dependencies.jar &
```

See the list of available environment variables

<u>How does DevOps and the understanding of DevOps Tie in with Test Automation?</u>

- ➔ When we write software, we are first going to develop features in separate branches, and then merge continuously into the main branch (CI)
- ➔ We will then have a continuous delivery setup which will first trigger the **unit tests** and the **integration tests** (whitebox) tests so that we can ensure that our code modules are working as intended
- ➔ If those pass, then we can move onto the step where we deploy our application to a **TEST** environment (for example, an EC2 instance that is not open to the public/customer, only for testers to access)
- ➔ Once your application is deployed to the **TEST** environment, testers can then execute **SMOKE TESTS (E2E)**. Smoke tests

are tests that verify critical functionality (such as log in, anything that is absolutely crucial and very important) that we run before all of our other tests, so that we don’t waste any time on the other tests if these smoke tests fail.

- ➔ If all the smoke tests pass, we can then move onto more extensive E2E testing
- ➔ Once we know that our application is then good to go with the E2E testing, we can also further validate the application through User Acceptance Testing (Beta and Alpha tests). This will be in another environment known as the **PRE-PROD (pre-production)** environment
- ➔ Once we have passed our UATs, we can then deploy the application to the **prod (production) environment**