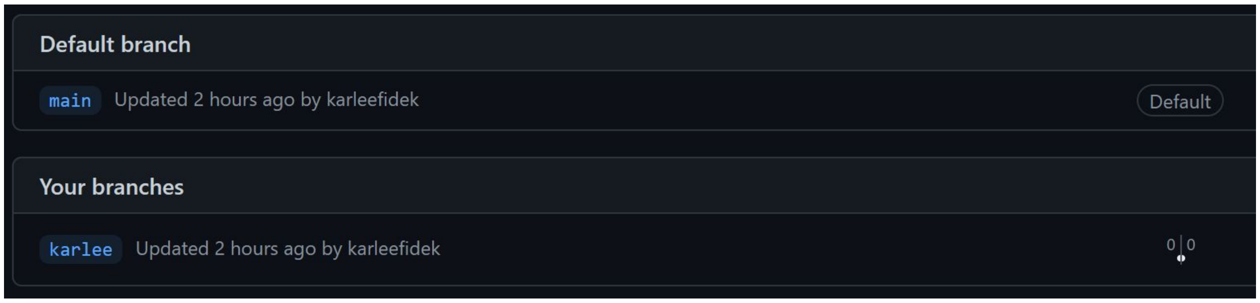
Group B

How we set up Jenkins using a Maven project.

Firstly, we create a new GitHub branch in which we will push our untested code.



Default branch

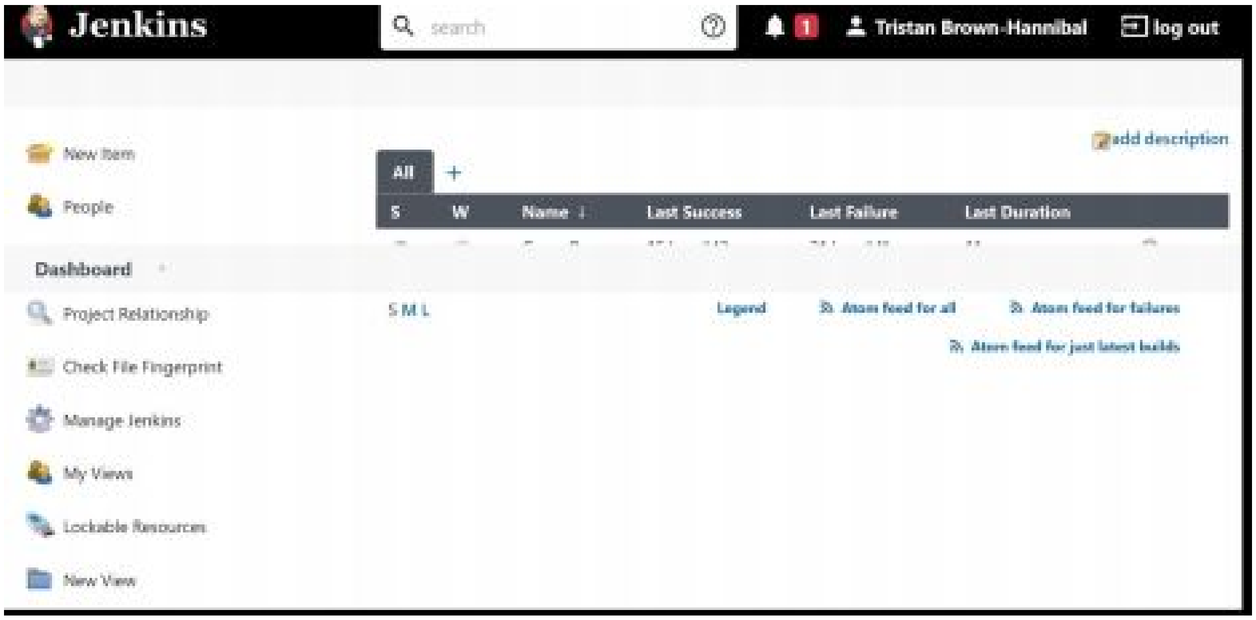
main Updated 2 hours ago by karleefidek Default

Your branches

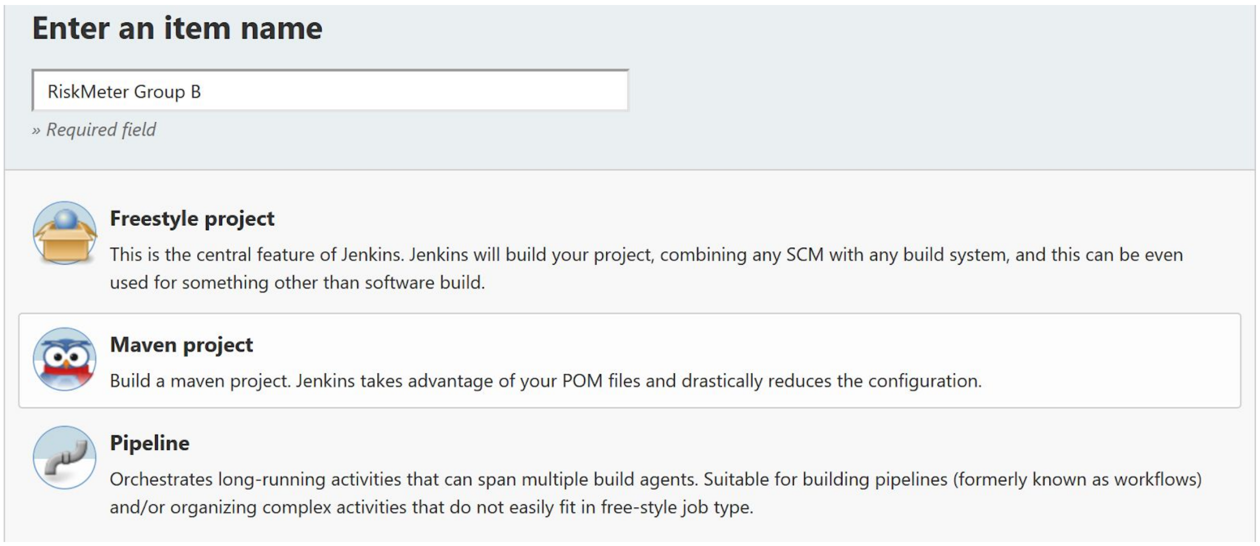
karlee Updated 2 hours ago by karleefidek 0|0

This branch is where the individual's work is placed, before being merged back with the main.

Then using docker we will run Jenkins, create an account and login.



From here, we created a new item, a Maven project.


Enter an item name

RiskMeter Group B

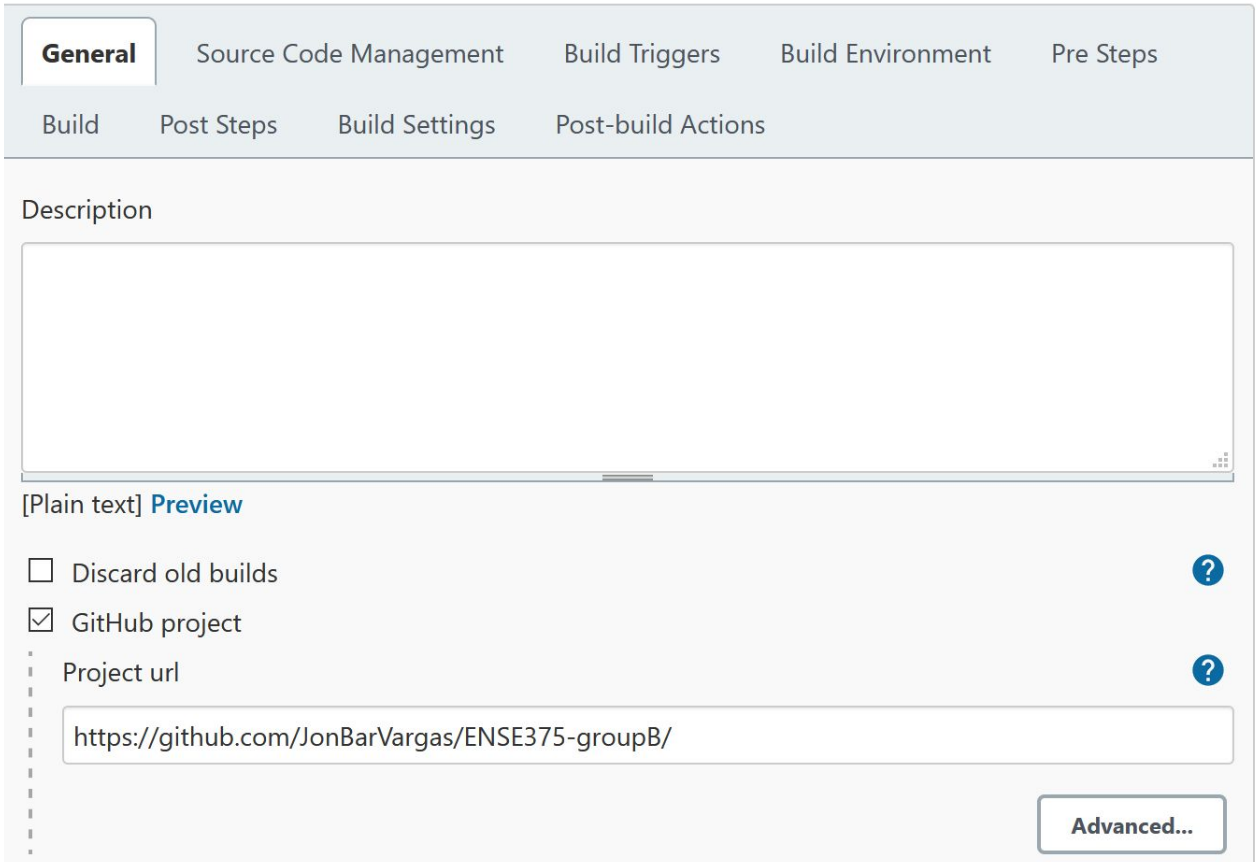
» *Required field*

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.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Then we configured it as such:
We link the project to our GitHub



Description

[Plain text] Preview

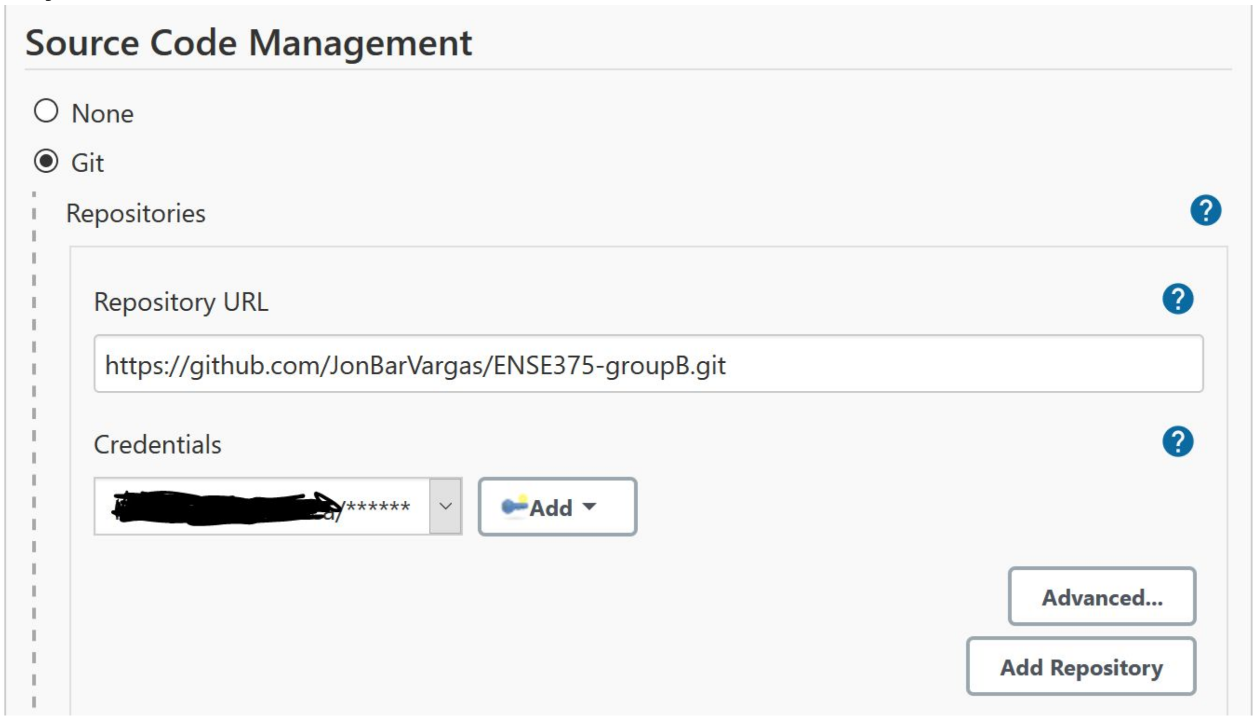
☐ Discard old builds

☑ GitHub project

Project url

https://github.com/JonBarVargas/ENSE375-groupB/

And Indicate we want to use Git, while providing valid credentials for the project to use.



Source Code Management

○ None

● Git

Repositories

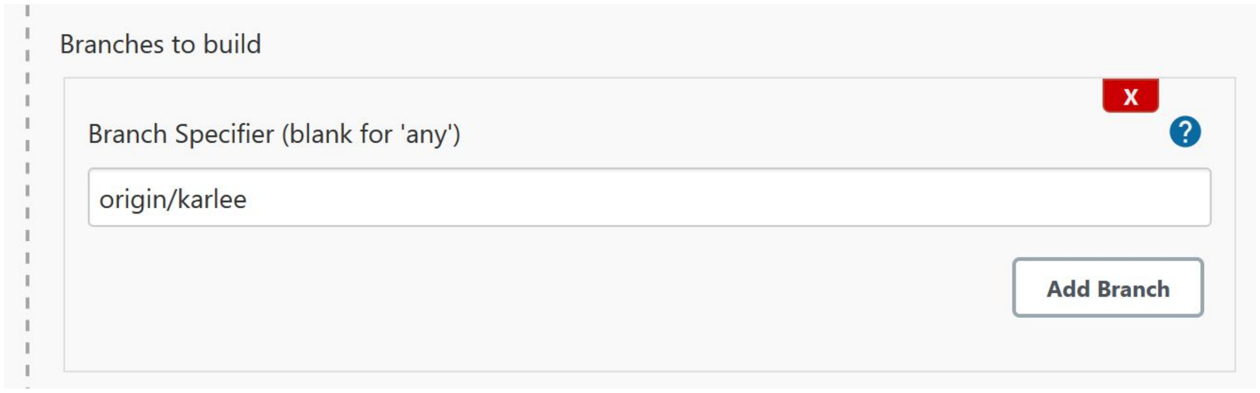
Repository URL

https://github.com/JonBarVargas/ENSE375-groupB.git

Credentials

[redacted]/******

Next, the individual chooses which branch they want to build and test (The one where they uploaded their own work)

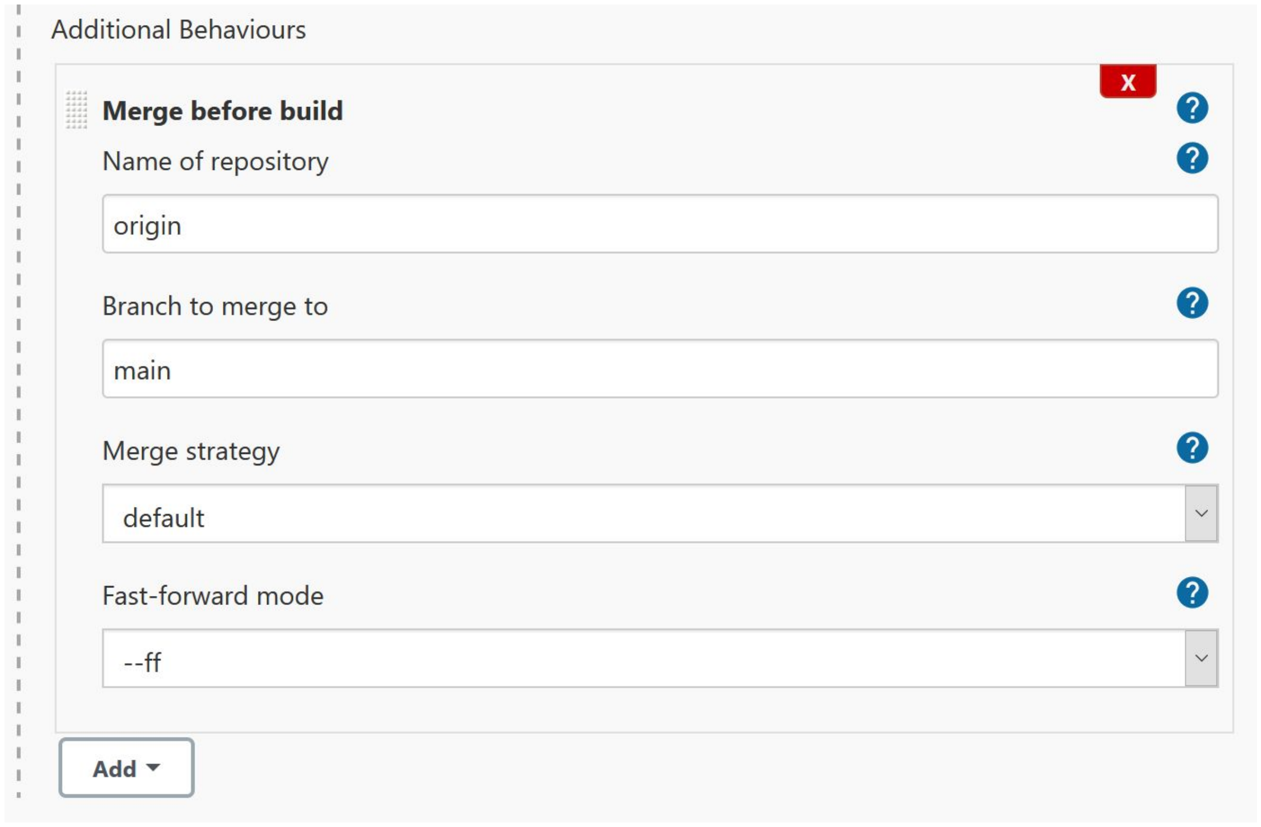


Branches to build

Branch Specifier (blank for 'any')

origin/karlee

Additional behaviour is required to merge, we indicate the process we want


Additional Behaviours

Merge before build

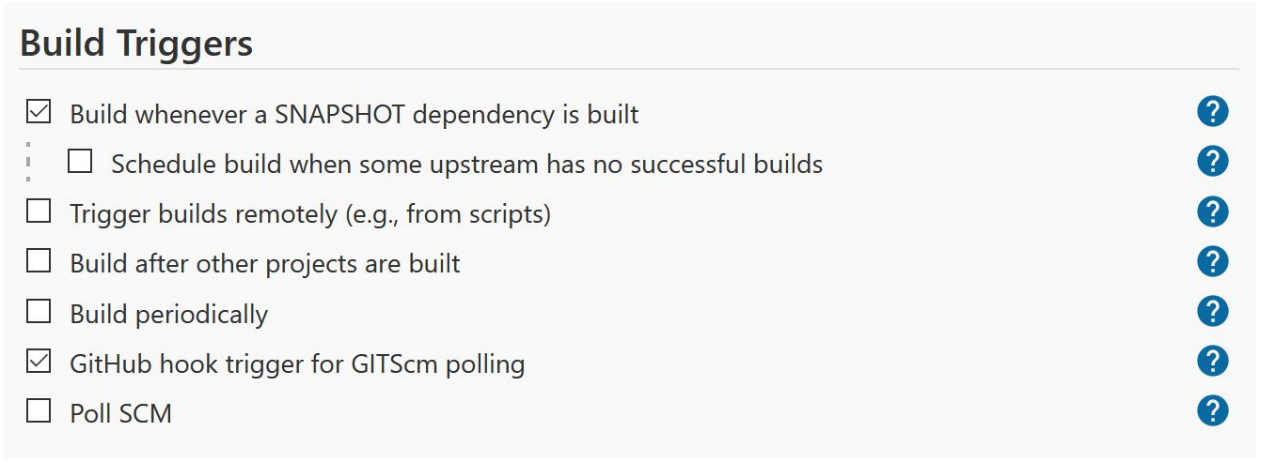
Name of repository: origin

Branch to merge to: main

Merge strategy: default

Fast-forward mode: --ff

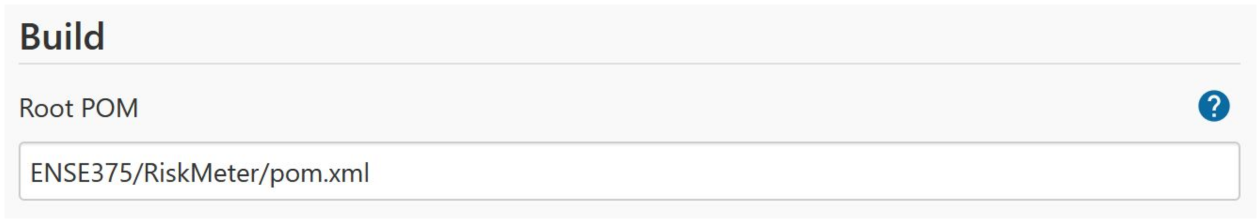
We tried to get webhooks working, but since we are on localhost, we couldn't figure out how to set up localhost and GitHub webhooks properly. But if this was a real server we could set up webhooks to allow an automatic build and test after committing to a GitHub repo


Build Triggers

- ☑ Build whenever a SNAPSHOT dependency is built
  - ☐ Schedule build when some upstream has no successful builds
- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☑ GitHub hook trigger for GITScm polling
- ☐ Poll SCM

Then we have to configure the tools Jenkins uses to build and test our project.

We specified the path to our pom.xml file as it was within sub-directories of our main repository


Build

Root POM

ENSE375/RiskMeter/pom.xml

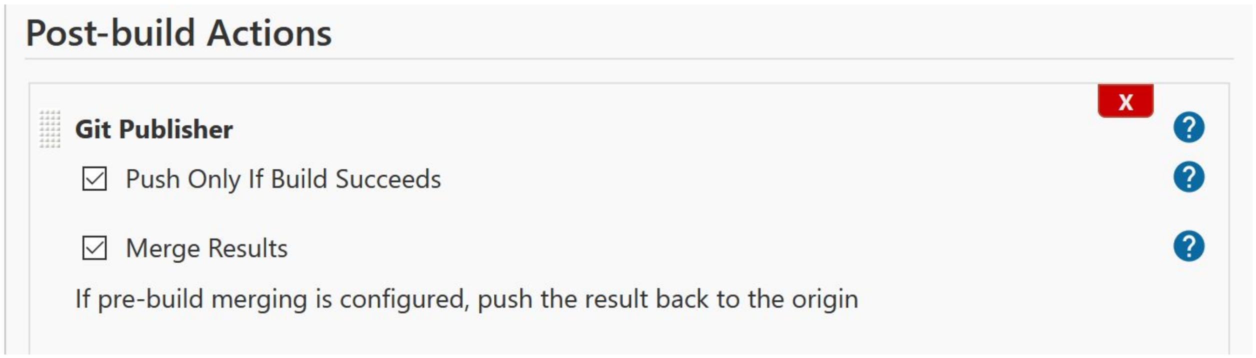
Our pom.xml file includes:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>org.ENSE375-groupB</groupId>
        <artifactId>ENSE375-groupB</artifactId>
        <packaging>jar</packaging>
        <version>0.1.0-SNAPSHOT</version>

        <name>ENSE375-groupB</name>
        <description>Project Step 3 - Continuous Integration - RiskMeter</description>

        <properties>
    <java.version>15</java.version>
         <maven.compiler.source>1.8</maven.compiler.source>
         <maven.compiler.target>1.8</maven.compiler.target>
        </properties>

        <dependencies>
         <dependency>
                        <groupId>org.junit.platform</groupId>
                        <artifactId>junit-platform-console-standalone</artifactId>
                        <version>1.7.0</version>
                        <scope>test</scope>
         </dependency>
         <dependency>
                        <groupId>org.junit.platform</groupId>
                        <artifactId>junit-platform-surefire-provider</artifactId>
                        <version>1.3.2</version>
```

```
                <scope>test</scope>
 </dependency>
</dependencies>

<build>
 <pluginManagement>
                <plugins>
        <plugin>
                <artifactId>maven-clean-plugin</artifactId>
                <version>3.1.0</version>
         </plugin>
         <plugin>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.0.2</version>
         </plugin>
         <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
         </plugin>
         <plugin>
                <artifactId>maven-jar-plugin</artifactId>
                <version>3.0.2</version>
         </plugin>
 <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.1</version>
         </plugin>
         <plugin>
                <artifactId>maven-install-plugin</artifactId>
                <version>2.5.2</version>
         </plugin>
         <plugin>
                <artifactId>maven-deploy-plugin</artifactId>
                        <version>2.8.2</version>
         </plugin>
         <plugin>
                <artifactId>maven-site-plugin</artifactId>
                        <version>3.7.1</version>
         </plugin>
         <plugin>
                <artifactId>maven-project-info-reports-plugin</artifactId>
                <version>3.0.0</version>
         </plugin>
         <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.0</version>
         </plugin>
```

```
                </plugins>
            </pluginManagement>
        </build>
</project>
```

We only want the branch to be merged with main if all the tests pass.



Post-build Actions

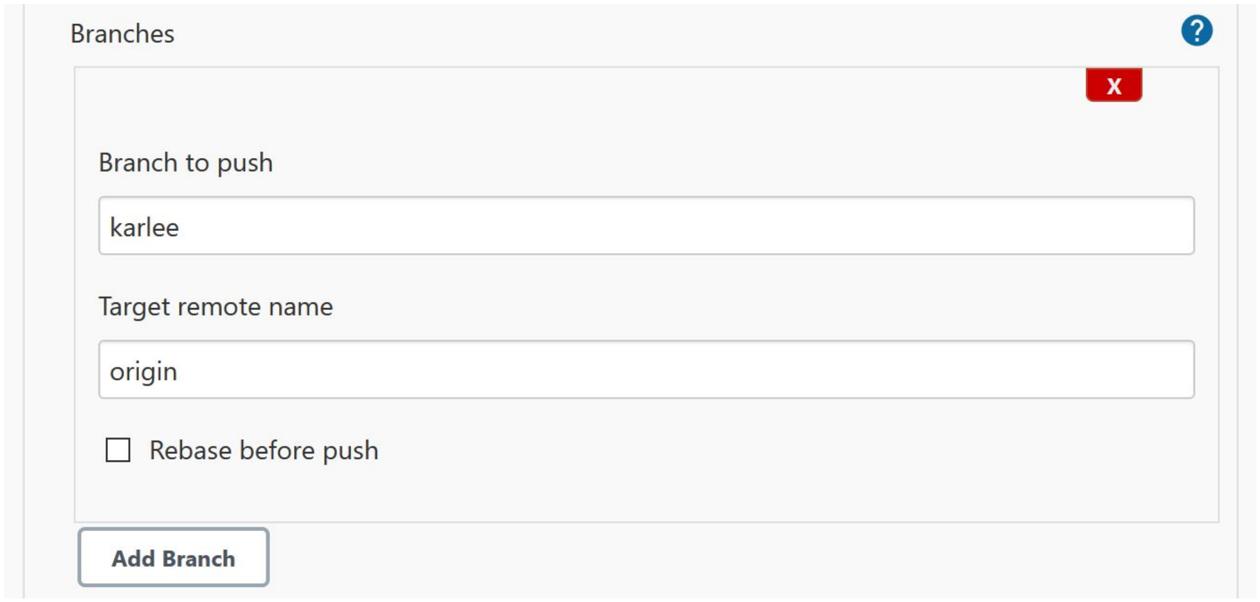
Git Publisher

☑ Push Only If Build Succeeds

☑ Merge Results

If pre-build merging is configured, push the result back to the origin

Here we indicate the branch and push target



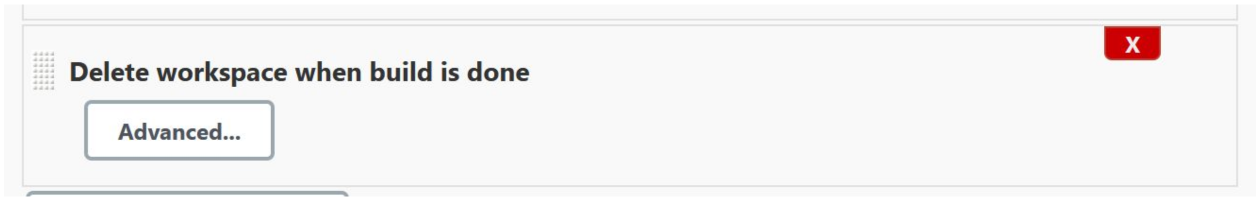
Branches

Branch to push: karlee

Target remote name: origin

☐ Rebase before push

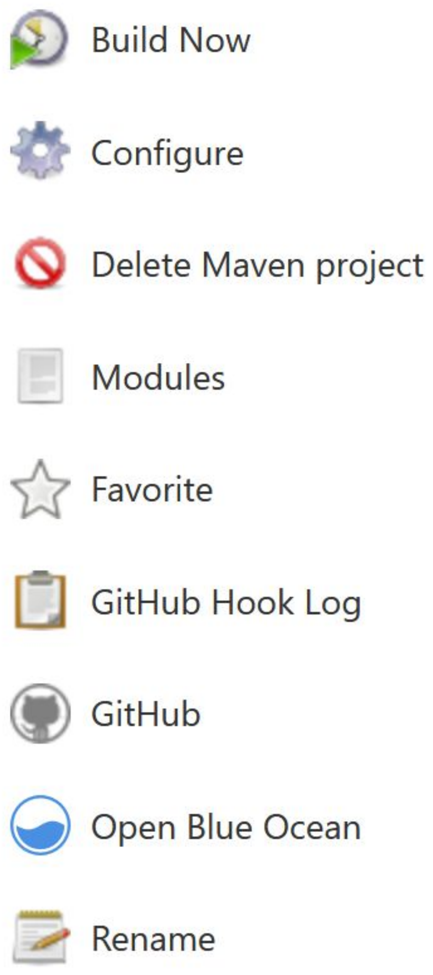
Once we are done, we delete the workspace so it is clean.

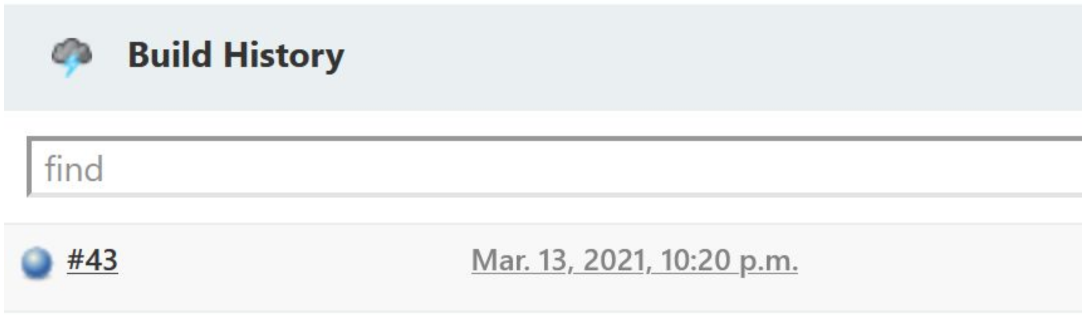


Delete workspace when build is done

This setup allows for easy building, testing and merging if successful.

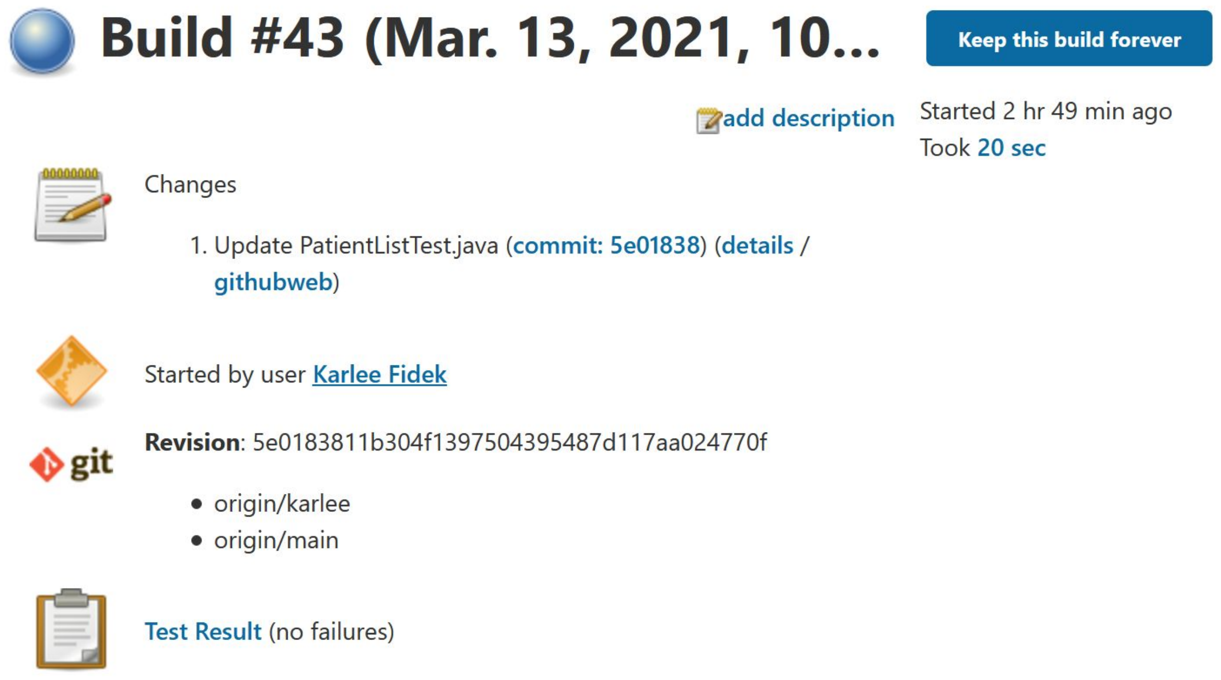
After making a commit, we can go to Jenkins, and manually build. This is an unneeded step if we had GitHub webhooks. Pressing ‘Build Now’ we can see Jenkins begin the process.





#43 Mar. 13, 2021, 10:20 p.m.

We can then view the details of the Jenkins build. Such as commit history and the test results.



Build #43 (Mar. 13, 2021, 10...

Keep this build forever

add description

Started 2 hr 49 min ago
Took 20 sec

Changes

1. Update PatientListTest.java (commit: 5e01838) (details / githubweb)

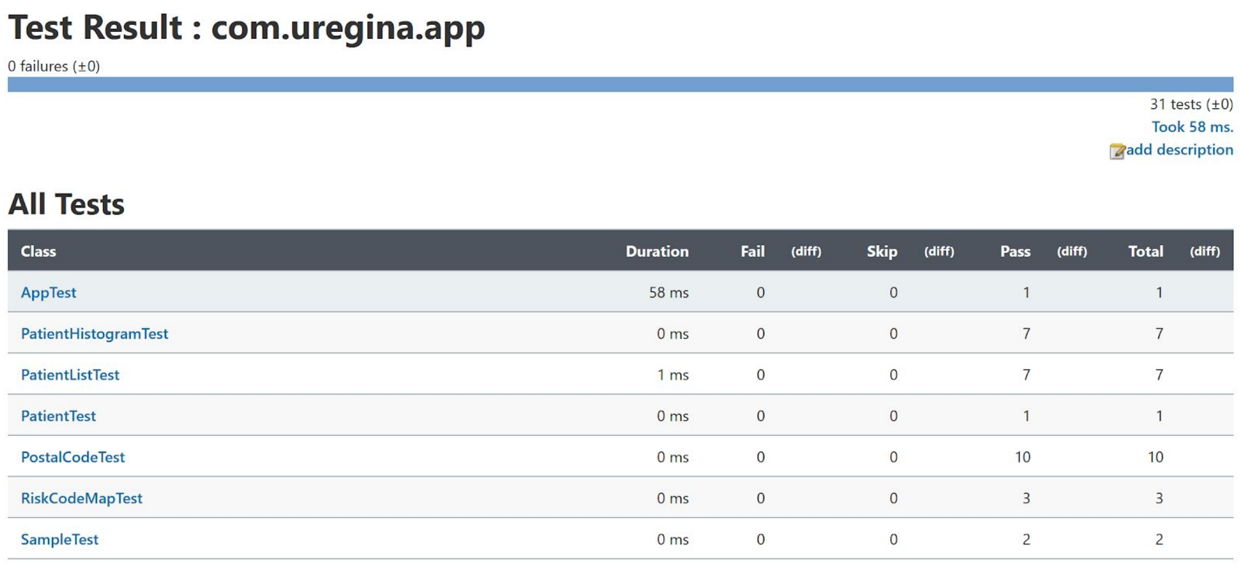
Started by user Karlee Fidek

Revision: 5e0183811b304f1397504395487d117aa024770f

- origin/karlee
- origin/main

Test Result (no failures)

Going into the test results we can view which tests pass and fail.



Test Result : com.uregina.app

0 failures (±0)

31 tests (±0)
Took 58 ms.
add description

All Tests

Class	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
AppTest	58 ms	0		0		1		1	
PatientHistogramTest	0 ms	0		0		7		7	
PatientListTest	1 ms	0		0		7		7	
PatientTest	0 ms	0		0		1		1	
PostalCodeTest	0 ms	0		0		10		10	
RiskCodeMapTest	0 ms	0		0		3		3	
SampleTest	0 ms	0		0		2		2	

Since all of our tests passed, Jenkins merged our branch into the main.

If the tests were to fail, nothing would’ve been merged to main. This process allows for easy collaboration of work, while still ensuring a high quality of code.