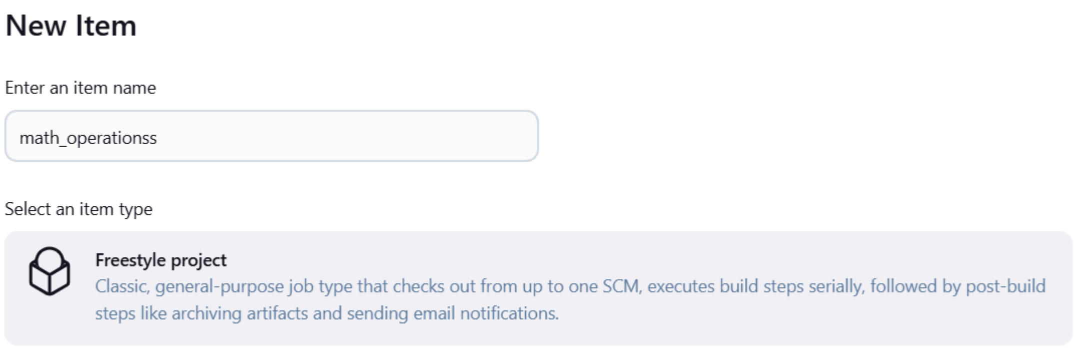
Homework 1

Step 1: Install Jenkins service on my machine.
Step 2: Browse to *http://localhost:8080* (the port that I configured for Jenkins when installing it) and wait until the Unlock Jenkins page appears.
Step 3: Create a new Freestyle project Jenkins item.



New Item

Enter an item name

math_operationss

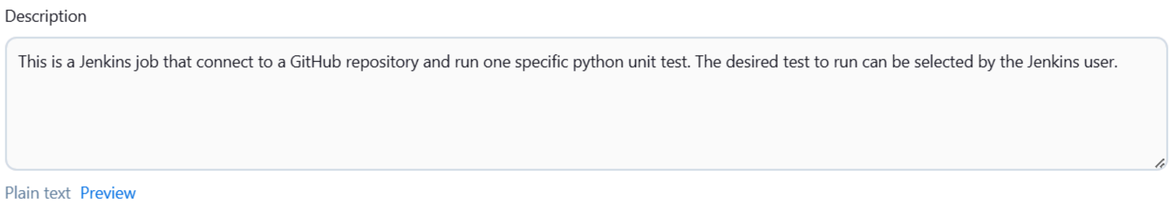
Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Step 4: Configure the job:

General

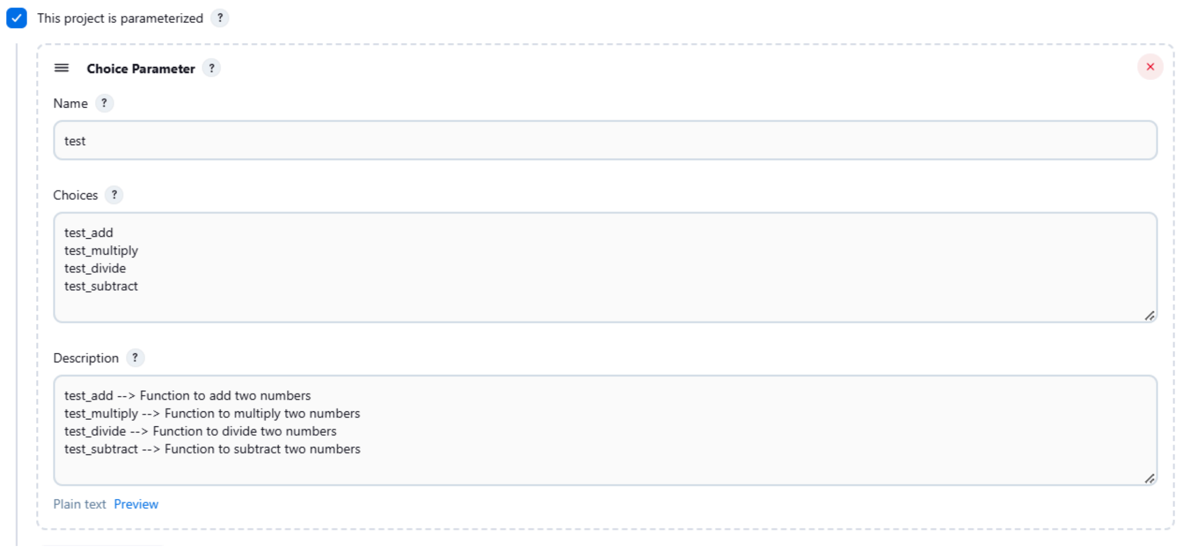
- add a description



Description

This is a Jenkins job that connect to a GitHub repository and run one specific python unit test. The desired test to run can be selected by the Jenkins user.

- select ***This project is parameterized*** option. Add a ***Choice Parameter*** and complete the fields



This project is parameterized ✓

Choice Parameter

Name: test

Choices:

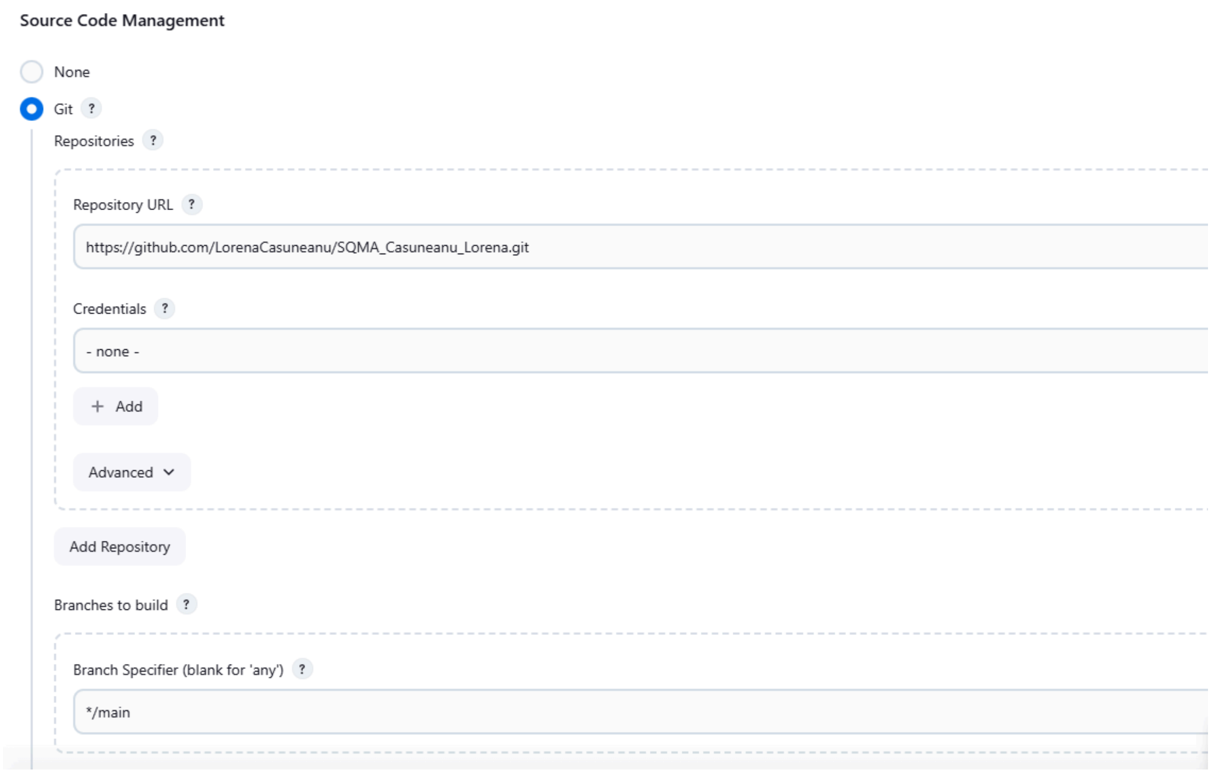
```
test_add
test_multiply
test_divide
test_subtract
```

Description:

```
test_add --> Function to add two numbers
test_multiply --> Function to multiply two numbers
test_divide --> Function to divide two numbers
test_subtract --> Function to subtract two numbers
```

Source Code Management

- select Git as source code. Provide the repository URL (https://github.com/LorenaCasuneanu/SQMA_Casuneanu_Lorena.git) and specify the branch to build:



Source Code Management

None

[✓ Git]

Repositories

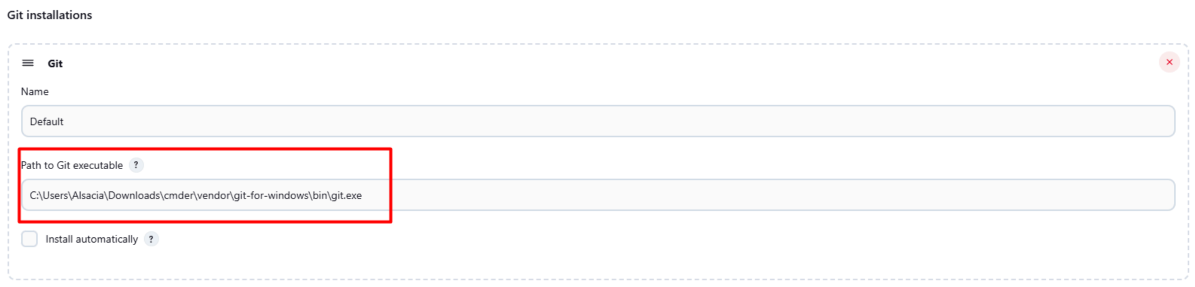
Repository URL: https://github.com/LorenaCasuneanu/SQMA_Casuneanu_Lorena.git

Credentials: - none -

Branches to build

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

At this step, because the Git path was not visible to the Jenkins job, I had to go to Dashboard → Manage Jenkins →Tools and provide the Git local path.



Git installations

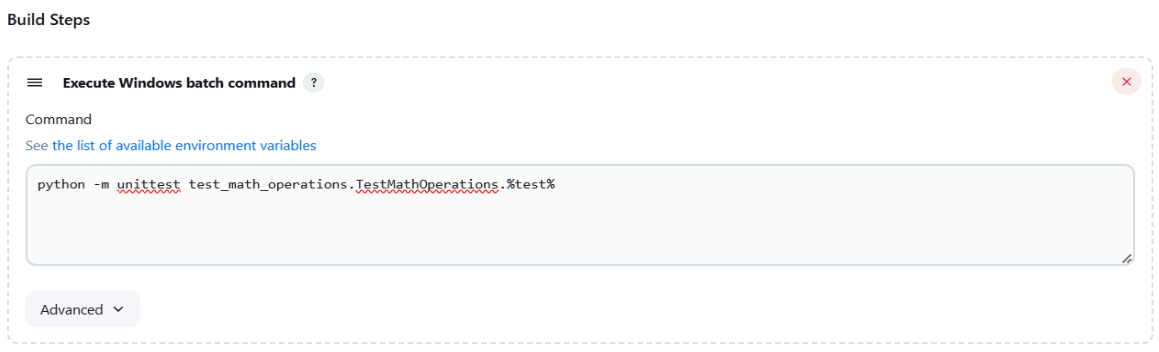
Git

Name: Default

Path to Git executable: C:\Users\Alsacia\Downloads\cmder\vendor\git-for-windows\bin\git.exe

Build Steps

- add a build step like ***Execute Windows batch command***



Build Steps

Execute Windows batch command

Command

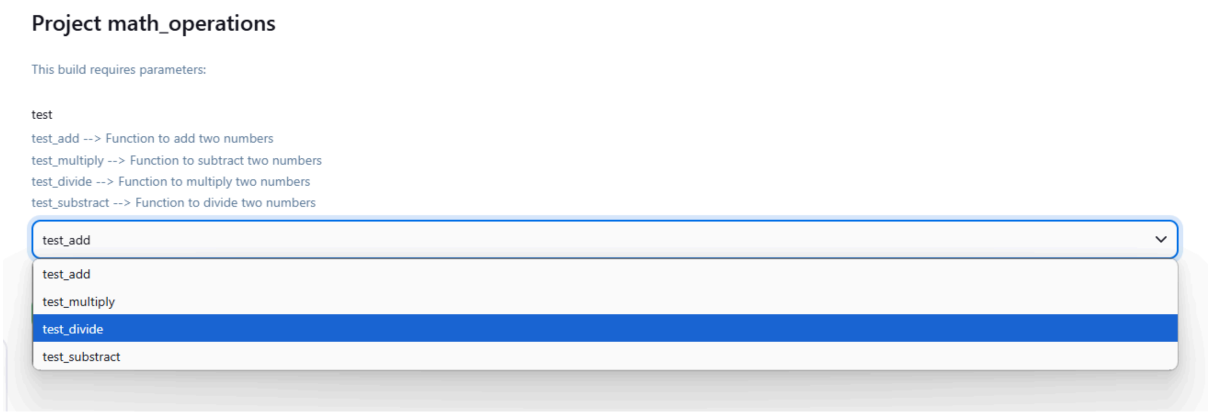
See the list of available environment variables

```
python -m unittest test_math_operations.TestMathOperations.%test%
```

The provided command will run **only** the `%test%` method from the `TestMathOperations` class, where `%test%` is an environment variable that will be replaced when running with the user selection.

Step 5: ***Apply*** and ***Save***.

Step 6: Go to ***Build with parameters*** tab and select the test to be run. Then press the ***Build*** button.



Project math_operations

This build requires parameters:

test

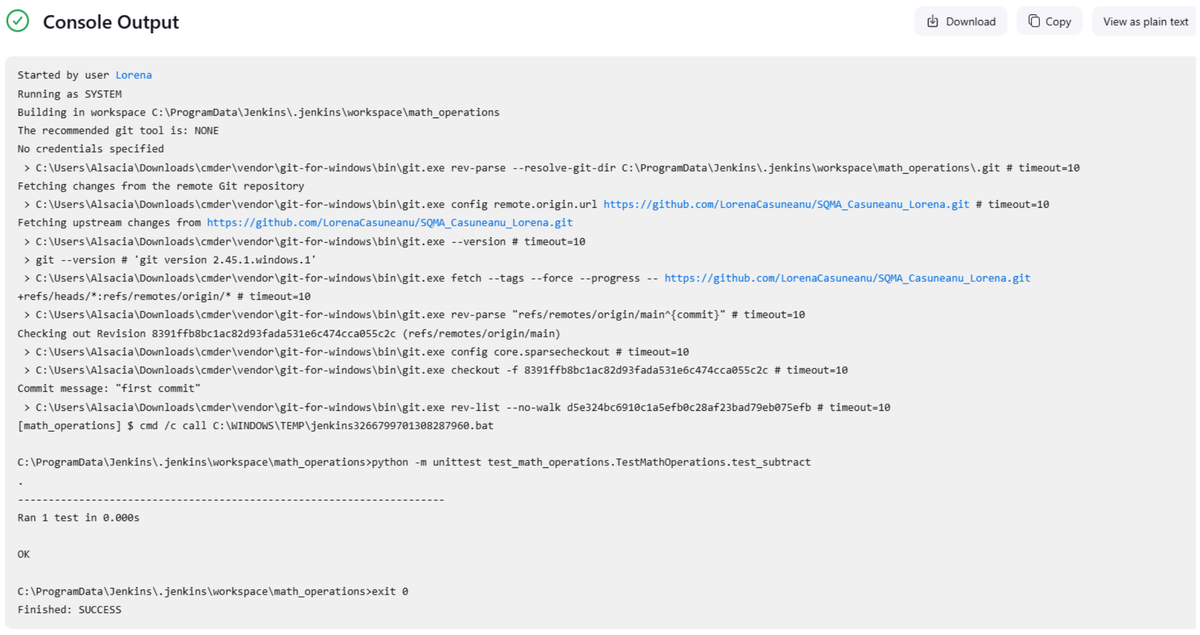
test_add --> Function to add two numbers

test_multiply --> Function to subtract two numbers

test_divide --> Function to multiply two numbers

test_substract --> Function to divide two numbers

Select the build then go to ***Console output***:

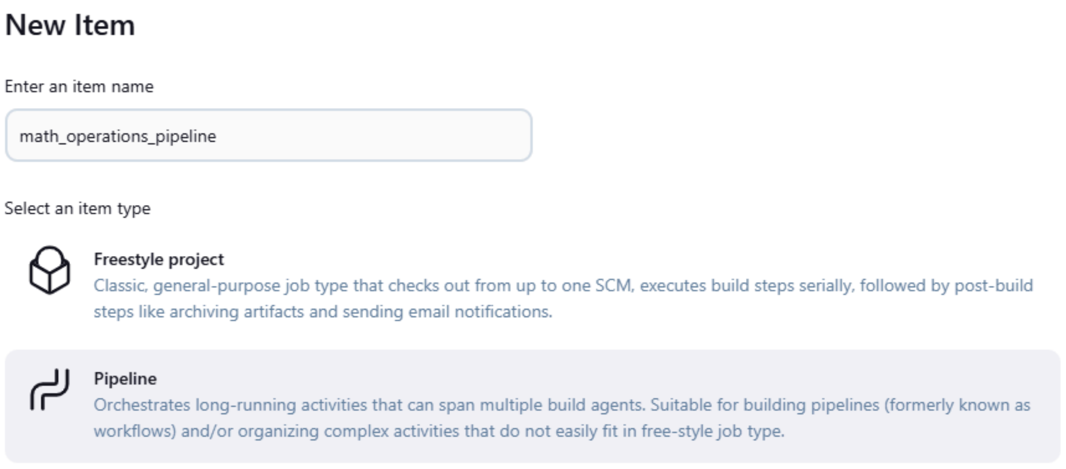


```
Started by user Lorena
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\math_operations
The recommended git tool is: NONE
No credentials specified
 > C:\Users\Alsacia\Downloads\cmder\vendor\git-for-windows\bin\git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\math_operations\.git # timeout=10
Fetching changes from the remote Git repository
 > C:\Users\Alsacia\Downloads\cmder\vendor\git-for-windows\bin\git.exe config remote.origin.url https://github.com/LorenaCasuneanu/SQMA_Casuneanu_Lorena.git # timeout=10
Fetching upstream changes from https://github.com/LorenaCasuneanu/SQMA_Casuneanu_Lorena.git
 > C:\Users\Alsacia\Downloads\cmder\vendor\git-for-windows\bin\git.exe --version # timeout=10
 > git --version # 'git version 2.45.1.windows.1'
 > C:\Users\Alsacia\Downloads\cmder\vendor\git-for-windows\bin\git.exe fetch --tags --force --progress -- https://github.com/LorenaCasuneanu/SQMA_Casuneanu_Lorena.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > C:\Users\Alsacia\Downloads\cmder\vendor\git-for-windows\bin\git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision 8391ffb8bc1ac82d93fada531e6c474cca055c2c (refs/remotes/origin/main)
 > C:\Users\Alsacia\Downloads\cmder\vendor\git-for-windows\bin\git.exe config core.sparsecheckout # timeout=10
 > C:\Users\Alsacia\Downloads\cmder\vendor\git-for-windows\bin\git.exe checkout -f 8391ffb8bc1ac82d93fada531e6c474cca055c2c # timeout=10
Commit message: "first commit"
 > C:\Users\Alsacia\Downloads\cmder\vendor\git-for-windows\bin\git.exe rev-list --no-walk d5e324bc6910c1a5efb0c28af23bad79eb075efb # timeout=10
[math_operations] $ cmd /c call C:\WINDOWS\TEMP\jenkins3266799701308287960.bat

C:\ProgramData\Jenkins\.jenkins\workspace\math_operations>python -m unittest test_math_operations.TestMathOperations.test_subtract
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

C:\ProgramData\Jenkins\.jenkins\workspace\math_operations>exit 0
Finished: SUCCESS
```

<u>Homework 2</u>

Approach: In this Jenkins pipeline, I triggered the previously parameterized job with all possible combinations of parameters. This approach allows us to systematically run all test cases by covering every valid parameter combination

Step 1: Create a new PipelineJenkins item and provide a description:



New Item

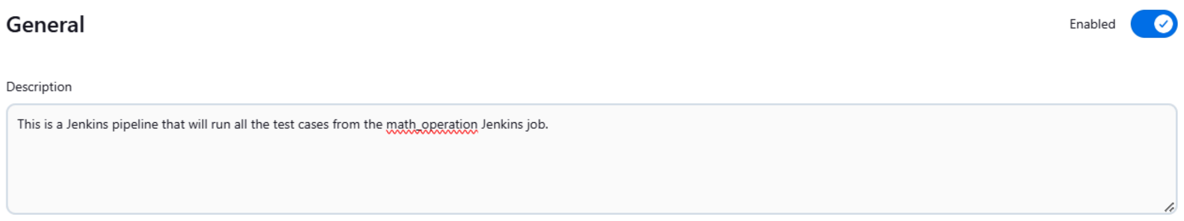
Enter an item name

math_operations_pipeline

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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.



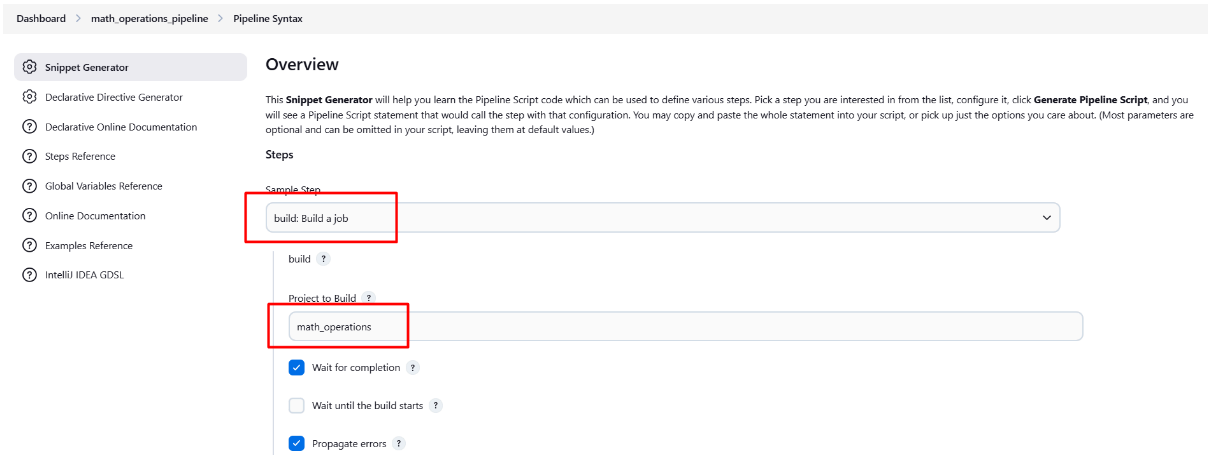
General Enabled

Description

This is a Jenkins pipeline that will run all the test cases from the math_operation Jenkins job.

Step 2: To generate the ***Pipeline*** script I use the ***Pipeline Syntax*** ***Snippet Generator***.

- select as ***Sample Step*** → build: Build a job. Use the previously created job (math_operations) and one of the possible parameters.



Dashboard > math_operations_pipeline > Pipeline Syntax

Snippet Generator
Declarative Directive Generator
Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

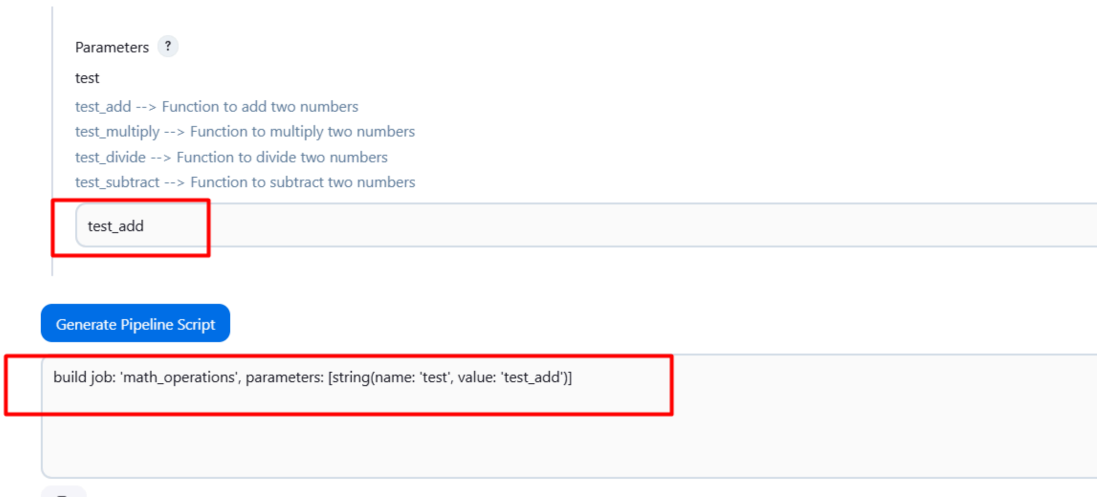
Steps

Sample Step: build: Build a job

build

Project to Build: math_operations

- [x] Wait for completion
- [ ] Wait until the build starts
- [x] Propagate errors



Parameters ?

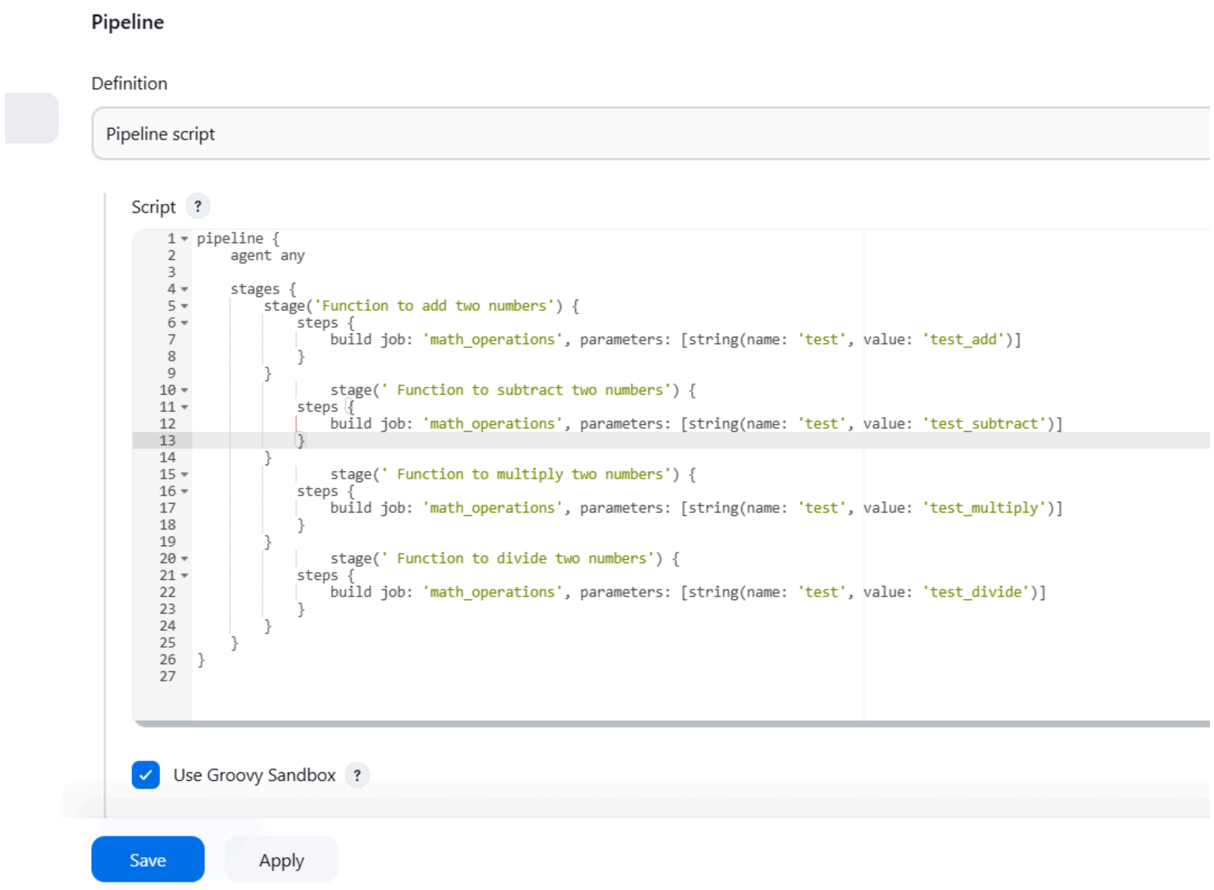
test

test_add --> Function to add two numbers
test_multiply --> Function to multiply two numbers
test_divide --> Function to divide two numbers
test_subtract --> Function to subtract two numbers

test_add

```
build job: 'math_operations', parameters: [string(name: 'test', value: 'test_add')]
```

After I repeat the steps for all the parameters, I obtained the pipeline script:



Pipeline

Definition

Pipeline script

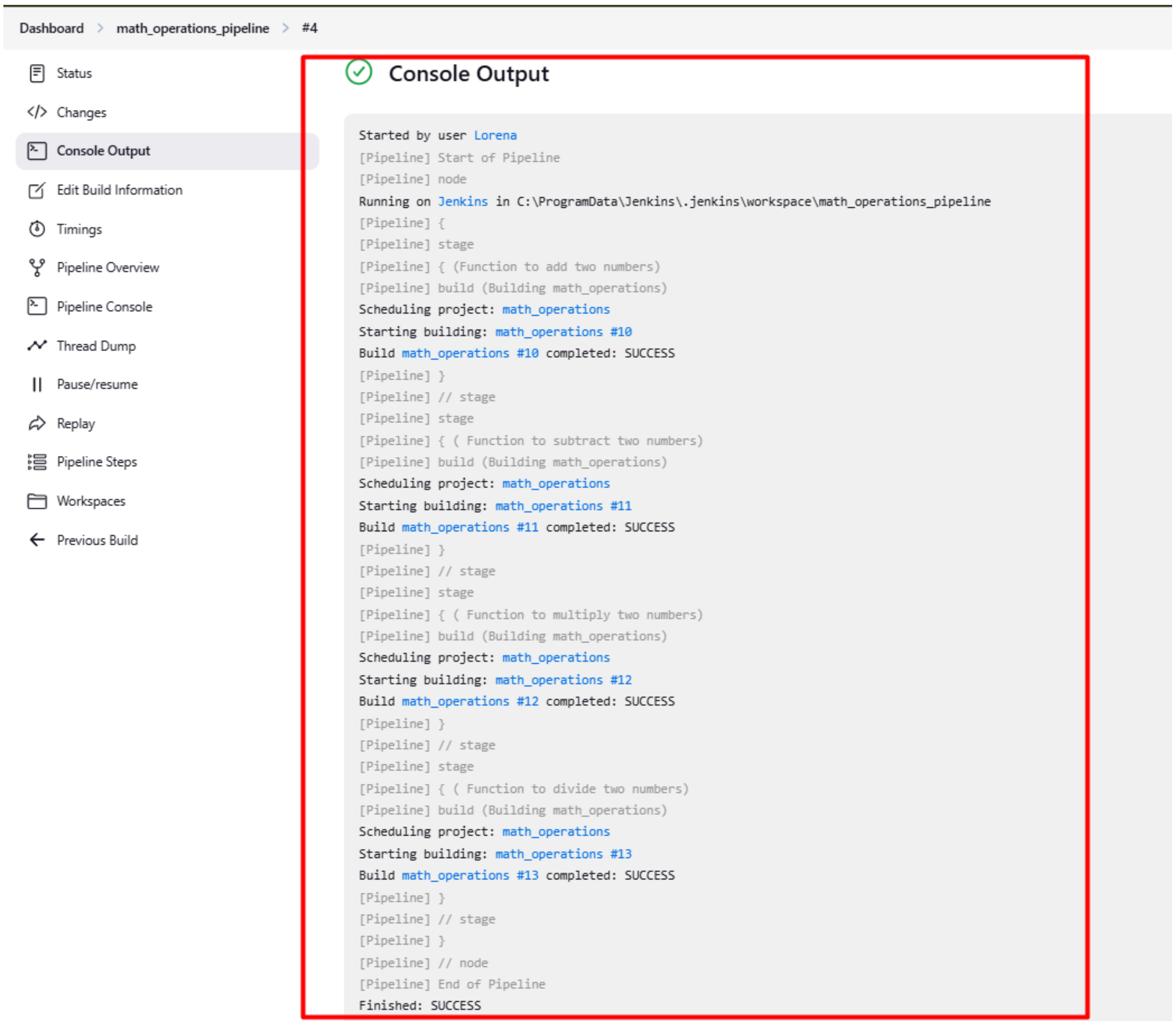
Script ?

```
pipeline {
    agent any

    stages {
        stage('Function to add two numbers') {
            steps {
                build job: 'math_operations', parameters: [string(name: 'test', value: 'test_add')]
            }
        }
                stage(' Function to subtract two numbers') {
            steps {
                build job: 'math_operations', parameters: [string(name: 'test', value: 'test_subtract')]
            }
        }
                stage(' Function to multiply two numbers') {
            steps {
                build job: 'math_operations', parameters: [string(name: 'test', value: 'test_multiply')]
            }
        }
                stage(' Function to divide two numbers') {
            steps {
                build job: 'math_operations', parameters: [string(name: 'test', value: 'test_divide')]
            }
        }
    }
}
```

Use Groovy Sandbox [✓]

Step 3: ***Apply*** and ***Save***, then Build.

Select the build then go to ***Console output***:


Dashboard > math_operations_pipeline > #4

Console Output

```
Started by user Lorena
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\math_operations_pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Function to add two numbers)
[Pipeline] build (Building math_operations)
Scheduling project: math_operations
Starting building: math_operations #10
Build math_operations #10 completed: SUCCESS
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { ( Function to subtract two numbers)
[Pipeline] build (Building math_operations)
Scheduling project: math_operations
Starting building: math_operations #11
Build math_operations #11 completed: SUCCESS
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { ( Function to multiply two numbers)
[Pipeline] build (Building math_operations)
Scheduling project: math_operations
Starting building: math_operations #12
Build math_operations #12 completed: SUCCESS
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { ( Function to divide two numbers)
[Pipeline] build (Building math_operations)
Scheduling project: math_operations
Starting building: math_operations #13
Build math_operations #13 completed: SUCCESS
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```