Deployment 3 Documentation

Goals:

- Fork (https://github.com/kura-labs-org/DEPLOY03_TEST)
- Screenshot the first successful test build.
- Document your added component or feature.
- Screenshot your failed test and document why your test failed.
- Screenshot your successful test build and document what you did to fix your failed test build
- Initiate a pull request to the kura_labs_org/DEPLOY_3_TESTING repo with your documentation, screenshots(add screenshots to documentation), test_calc.py with your added test, and add2vals.py with your added feature or component.

Goal 1 and 2:

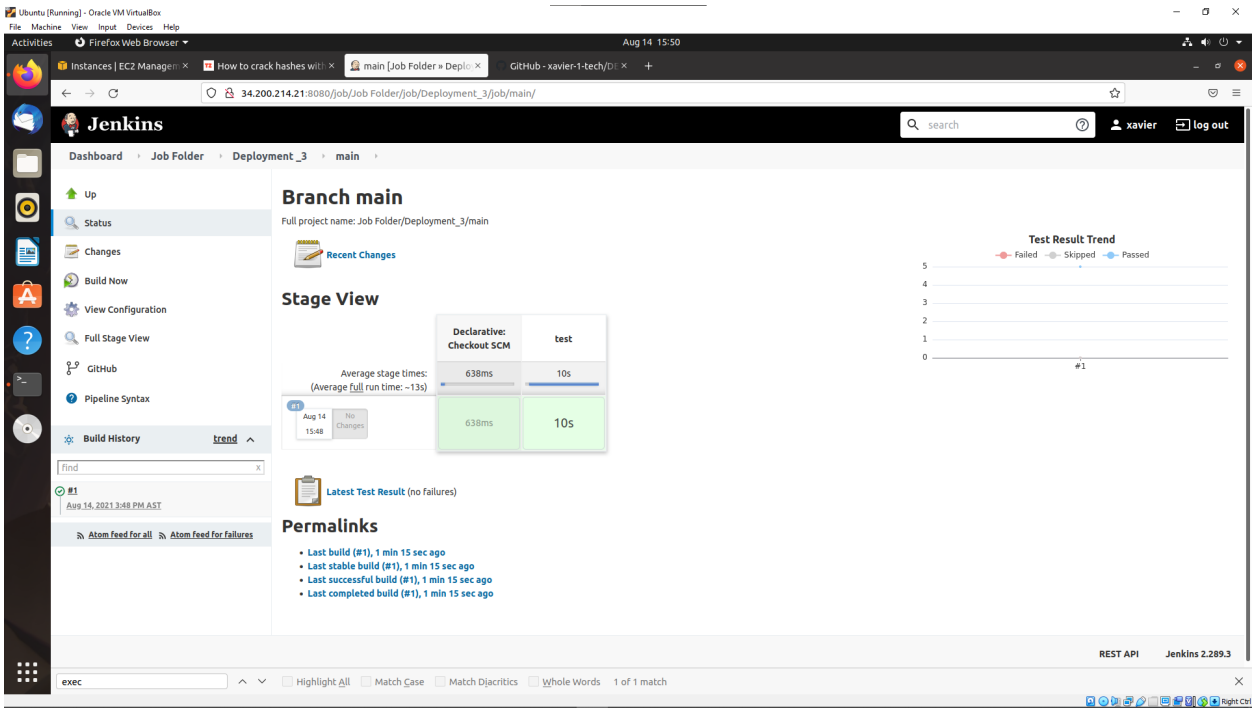
The repo was successfully forked. Forked Repo URL:
https://github.com/xavier-1-tech/DEPLOY03_TEST

With the following Jenkinsfile, Jenkins was able successfully pull the python code and successfully run the build:

```
pipeline {
    agent any
    stages {
        stage('test') {
            steps {
```

```
            sh'''#!/bin/bash
            python3 -m venv test3
            source test3/bin/activate
            pip install pip --upgrade
            pip install pytest
            py.test --verbose --junit-xml test-reports/results.xml sources/test_calc.py
            '''
        }
        post {
            always {
                junit 'test-reports/results.xml'
            }
        }
    }
}
```

First Build Success Image:



Goal 3:

The feature I decided to add was the ability to process 5 numbers instead of 2.

Updated add2vals.py code:

```
import sys
import calc

argnumbers = len(sys.argv) - 1

if argnumbers == 5 :
    print("")
    print("The result is " + str(calc.add2(str(sys.argv[1]),
str(sys.argv[2]))))
    print("")
    sys.exit(0)

if argnumbers != 5 :
    print("")
    print("You entered " + str(argnumbers) + " value/s.")
    print("")
    print("Usage: 'add2vals X Y' where X and Y are individual values.")
    print("       If add2vals is not in your path, usage is './add2vals X
Y'.")
    print("       If unbundled, usage is 'python add2vals.py X Y'.")
    print("")
    sys.exit(1)
```

Updated calc.py code:

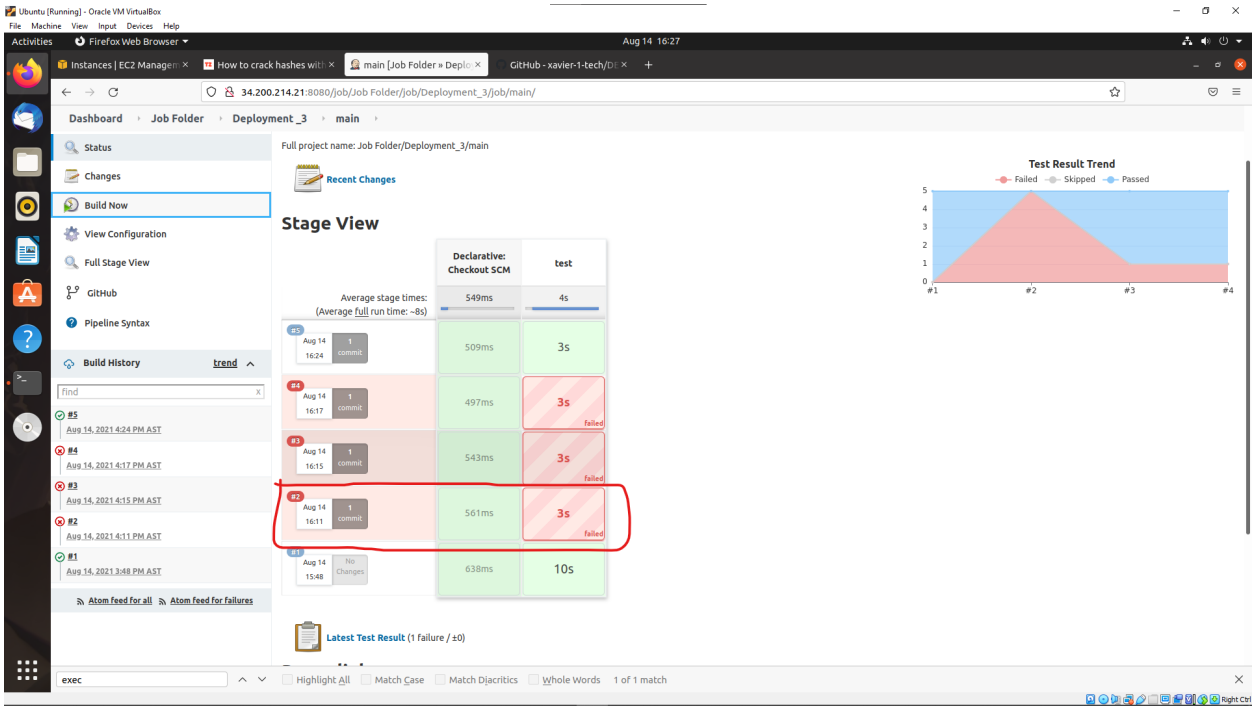
```
# If 'value' is not an integer, convert it to a float and failing that, a
string.
def conv(value):
    try:
        return int(value)
    except ValueError:
        try:
```

```
            return float(value)
        except ValueError:
            return str(value)

# The 'add2' function itself
def add2(arg1, arg2, arg3, arg4, arg5):
    # Convert 'arg1' and 'arg2' to their appropriate types
    arg1conv = conv(arg1)
    arg2conv = conv(arg2)
    arg3conv = conv(arg3)
    arg4conv = conv(arg4)
    arg5conv = conv(arg5)
    # If either 'arg1' or 'arg2' is a string, ensure they're both strings.
    if isinstance(arg1conv, str) or isinstance(arg2conv, str) or
isinstance(arg3conv, str) or isinstance(arg4conv, str) or
isinstance(arg5conv, str):
        arg1conv = str(arg1conv)
        arg2conv = str(arg2conv)
        arg3conv = str(arg3conv)
        arg4conv = str(arg4conv)
        arg5conv = str(arg5conv)
    return arg1conv + arg2conv + arg3conv + arg4conv + arg5conv
```

Goal 4:

The first error I encountered was due to my test file(test_calc.py) being incomplete. After I added the ability to process 5 values, I did not add the additional value to the test cases in the test file. The file was still testing for two values only, resulting in the error.



Error:

```
platform linux -- Python 3.7.10, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 --
/var/lib/jenkins/workspace/Job_Folder_Deployment_3_main/test3/bin/python3
cachedir: .pytest_cache
rootdir: /var/lib/jenkins/workspace/Job_Folder_Deployment_3_main
collecting ... collected 5 items

sources/test_calc.py::TestCalc::test_add_floats FAILED                [ 20%]
sources/test_calc.py::TestCalc::test_add_integers FAILED              [ 40%]
sources/test_calc.py::TestCalc::test_add_string_and_integer FAILED    [ 60%]
sources/test_calc.py::TestCalc::test_add_string_and_number FAILED     [ 80%]
sources/test_calc.py::TestCalc::test_add_strings FAILED               [100%]

================================== FAILURES
==================================
_____________________________ TestCalc.test_add_floats _____________________________

self = <test_calc.TestCalc testMethod=test_add_floats>

    def test_add_floats(self):
        '''
```

```
        Test that the addition of two floats returns the correct result
        """
>       result = calc.add2('10.5', 2)
E       TypeError: add2() missing 3 required positional arguments: 'arg3', 'arg4', and
'arg5'

sources/test_calc.py:20: TypeError
_____________________________ TestCalc.test_add_integers ______________________________

self = <test_calc.TestCalc testMethod=test_add_integers>

    def test_add_integers(self):
        """
        Test that the addition of two integers returns the correct total
        """
>       result = calc.add2(1, 2)
E       TypeError: add2() missing 3 required positional arguments: 'arg3', 'arg4', and
'arg5'

sources/test_calc.py:13: TypeError
_____________________ TestCalc.test_add_string_and_integer _____________________

self = <test_calc.TestCalc testMethod=test_add_string_and_integer>

    def test_add_string_and_integer(self):
        """
        Test the addition of a string and an integer returns them as one
        concatenated string (in which the integer is converted to a string)
        """
>       result = calc.add2('abc', 3)
E       TypeError: add2() missing 3 required positional arguments: 'arg3', 'arg4', and
'arg5'

sources/test_calc.py:36: TypeError
_____________________ TestCalc.test_add_string_and_number _____________________

self = <test_calc.TestCalc testMethod=test_add_string_and_number>

    def test_add_string_and_number(self):
        """
        Test the addition of a string and a float returns them as one
```

```
        concatenated string (in which the float is converted to a string)
        """
>       result = calc.add2('abc', '5.5')
E       TypeError: add2() missing 3 required positional arguments: 'arg3', 'arg4', and
'arg5'

sources/test_calc.py:44: TypeError
_____________________________ TestCalc.test_add_strings _____________________________

self = <test_calc.TestCalc testMethod=test_add_strings>

    def test_add_strings(self):
        """
        Test the addition of two strings returns the two strings as one
        concatenated string
        """
>       result = calc.add2('abc', 'def')
E       TypeError: add2() missing 3 required positional arguments: 'arg3', 'arg4', and
'arg5'

sources/test_calc.py:28: TypeError
- generated xml file:
/var/lib/jenkins/workspace/Job_Folder_Deployment_3_main/test-reports/results.xm
l -
=========================== short test summary info
============================
FAILED sources/test_calc.py::TestCalc::test_add_floats - TypeError: add2() mi...
FAILED sources/test_calc.py::TestCalc::test_add_integers - TypeError: add2() ...
FAILED sources/test_calc.py::TestCalc::test_add_string_and_integer - TypeErro...
FAILED sources/test_calc.py::TestCalc::test_add_string_and_number - TypeError...
FAILED sources/test_calc.py::TestCalc::test_add_strings - TypeError: add2() m...
============================== 5 failed in 0.06s
==============================
```

Error test file code snippet:

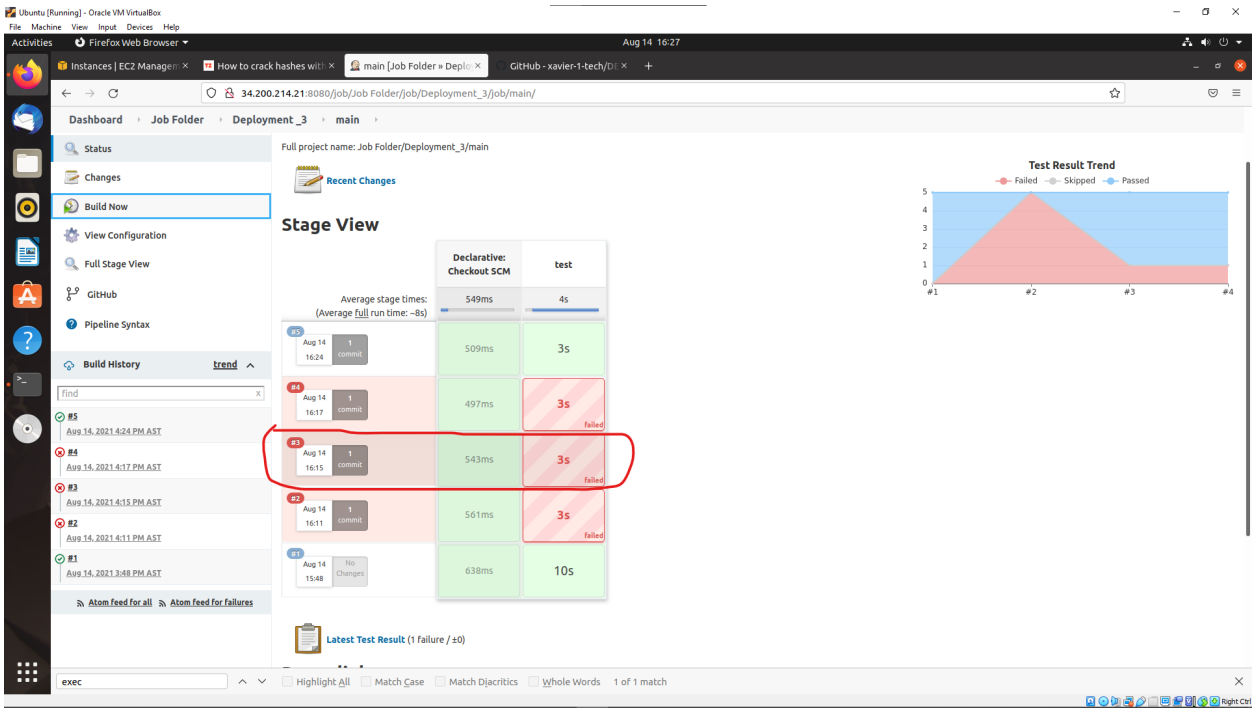
```
def test_add_integers(self):
        """
        Test that the addition of two integers returns the correct total
```

```
    """
    result = calc.add2(1, 2)
    self.assertEqual(result, 3)
```

Correctly modified test file code snippet for new feature:

```
def test_add_integers(self):
    """
    Test that the addition of two integers returns the correct total
    """
    result = calc.add2(1, 2, 4, 6, 10)
    self.assertEqual(result, 23)
```

The second error I encountered was due to human error while editing the test file. I failed to add single quotes around the floats in one test case and that resulted in an error being thrown. Human formatting error.



Error:

```
platform linux -- Python 3.7.10, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 --
/var/lib/jenkins/workspace/Job_Folder_Deployment_3_main/test3/bin/python3
cachedir: .pytest_cache
rootdir: /var/lib/jenkins/workspace/Job_Folder_Deployment_3_main
collecting ... collected 5 items

sources/test_calc.py::TestCalc::test_add_floats FAILED                   [
20%]
sources/test_calc.py::TestCalc::test_add_integers PASSED                 [
40%]
sources/test_calc.py::TestCalc::test_add_string_and_integer PASSED       [
60%]
sources/test_calc.py::TestCalc::test_add_string_and_number PASSED        [
80%]
sources/test_calc.py::TestCalc::test_add_strings PASSED
[100%]
```

```
=================================== FAILURES
===================================
_____________________________ TestCalc.test_add_floats
_____________________________

self = <test_calc.TestCalc testMethod=test_add_floats>

    def test_add_floats(self):
        """
        Test that the addition of two floats returns the correct result
        """
        result = calc.add2('10.5', 2, 22.1, 44, 0.3)
>       self.assertEqual(result, 78.9)
E       AssertionError: 78.5 != 78.9

sources/test_calc.py:21: AssertionError
- generated xml file:
/var/lib/jenkins/workspace/Job_Folder_Deployment_3_main/test-reports/result
s.xml -
============================ short test summary info
============================
FAILED sources/test_calc.py::TestCalc::test_add_floats - AssertionError:
78.5...
========================= 1 failed, 4 passed in 0.06s
=========================
```

Error test file snippet code snippet:

```
def test_add_floats(self):
    """
    Test that the addition of two floats returns the correct result
    """
    result = calc.add2('10.5', 2, 22.1, 44, 0.3)
    self.assertEqual(result, 78.9)
```

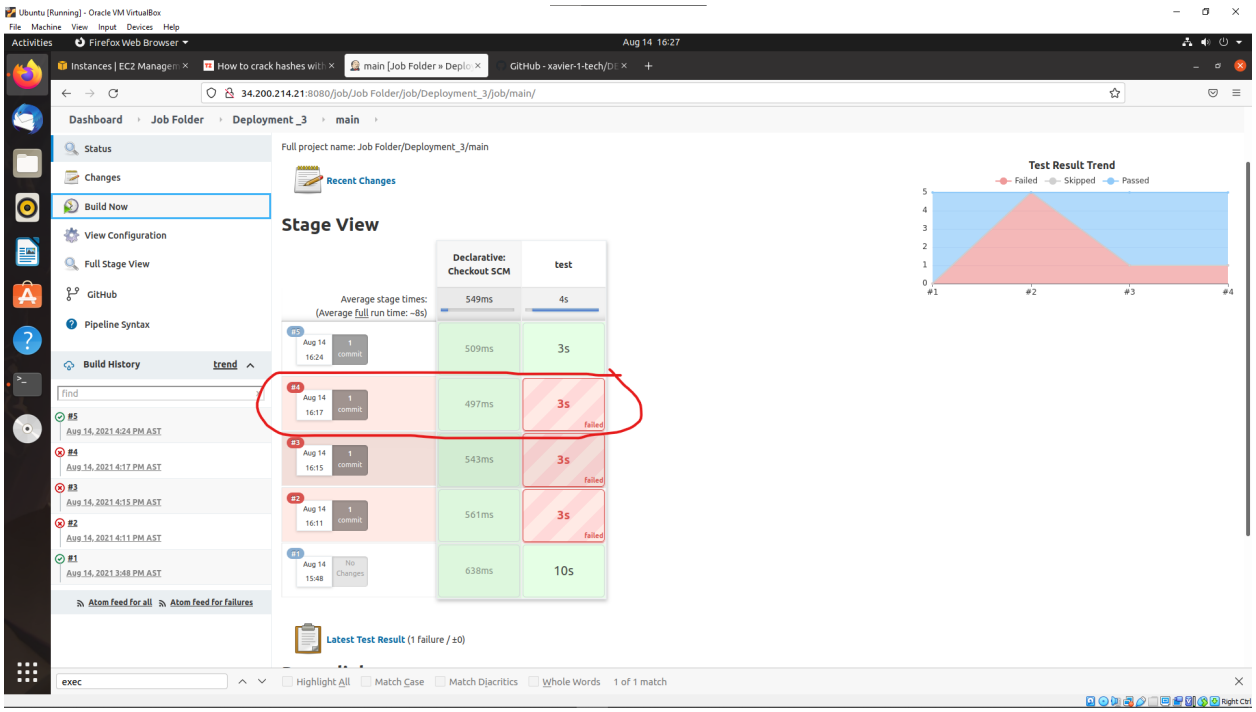
Correctly modified test file code snippet for new feature:

```
def test_add_floats(self):
    """
```

```
    Test that the addition of two floats returns the correct result
    """
    result = calc.add2('10.5', 2, '22.1', 44, '0.3')
    self.assertEqual(result, 78.9)
```

The final error I encountered was a calculation error with the floats test case. The purpose of this test case was to add the integers and floats together to produce a result. As the test case had predetermined values for the floats and integers, a predetermined sum was also given to ensure that the calculation was correct. The issue was when I updated those predetermined values and it's sum, the calculated sum did not match the predetermined sum. The sum of the values I originally updated caused a recurring number loop and thus did not match my predetermined value sum.

Solution: update the predetermined values with ones that do not cause a recurring number loop.



Error:

```
platform linux -- Python 3.7.10, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 --
/var/lib/jenkins/workspace/Job_Folder_Deployment_3_main/test3/bin/python3
cachedir: .pytest_cache
rootdir: /var/lib/jenkins/workspace/Job_Folder_Deployment_3_main
collecting ... collected 5 items

sources/test_calc.py::TestCalc::test_add_floats FAILED                   [
20%]
sources/test_calc.py::TestCalc::test_add_integers PASSED                 [
40%]
sources/test_calc.py::TestCalc::test_add_string_and_integer PASSED       [
60%]
sources/test_calc.py::TestCalc::test_add_string_and_number PASSED        [
80%]
sources/test_calc.py::TestCalc::test_add_strings PASSED
[100%]

================================== FAILURES
==================================
_____________________________ TestCalc.test_add_floats
_____________________________

self = <test_calc.TestCalc testMethod=test_add_floats>

    def test_add_floats(self):
        """
        Test that the addition of two floats returns the correct result
        """
        result = calc.add2('10.5', 2, '22.1', 44, '0.3')
>       self.assertEqual(result, 78.9)
E       AssertionError: 78.89999999999999 != 78.9

sources/test_calc.py:21: AssertionError
- generated xml file:
/var/lib/jenkins/workspace/Job_Folder_Deployment_3_main/test-reports/result
s.xml -
============================ short test summary info
============================
FAILED sources/test_calc.py::TestCalc::test_add_floats - AssertionError:
78.8...
======================== 1 failed, 4 passed in 0.06s
```

```
==========================
```

Error test file snippet code snippet:

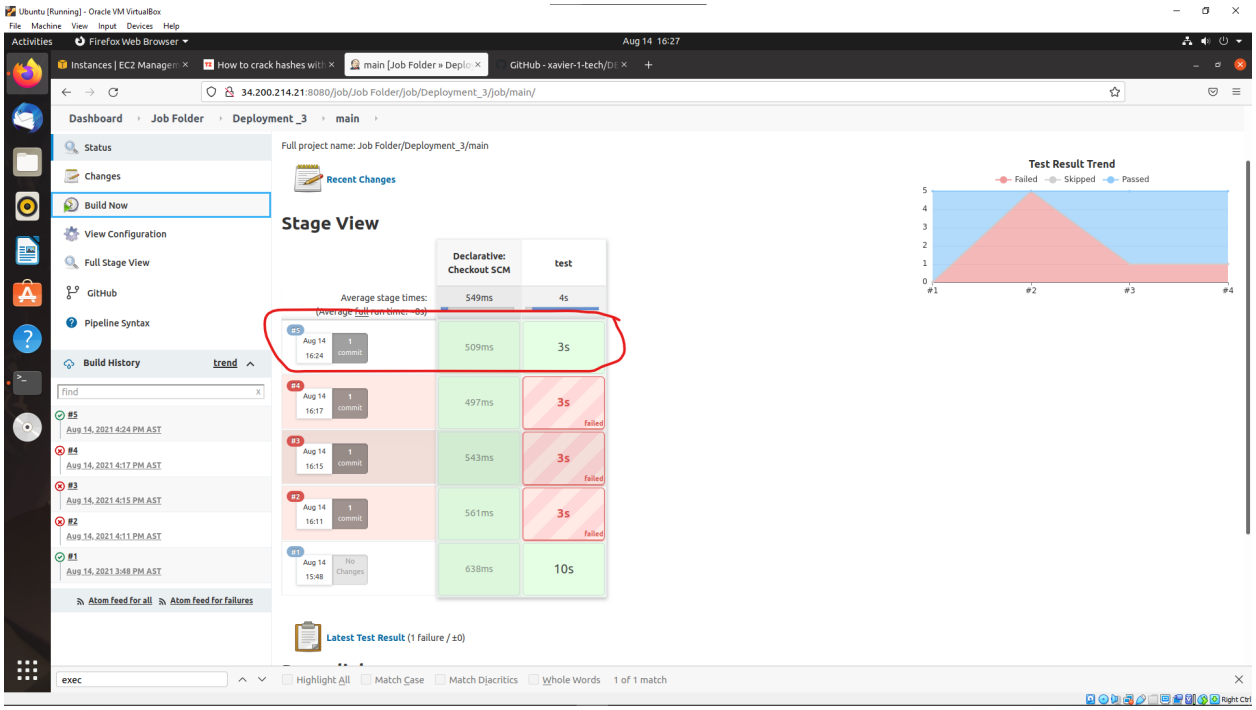
```
def test_add_floats(self):
        """
        Test that the addition of two floats returns the correct result
        """
        result = calc.add2('10.5', 2, '22.1', 44, '0.3')
        self.assertEqual(result, 78.9) #The code calculated 78.9999999
```

Correctly modified test file code snippet for new feature:

```
def test_add_floats(self):
        """
        Test that the addition of two floats returns the correct result
        """
        result = calc.add2('10.5', 2, '22.5', 44, '0.5')
        self.assertEqual(result, 79.5)
```

Goal 5:

After fixing my errors the build successfully ran:



Success log:

```
platform linux -- Python 3.7.10, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 --
/var/lib/jenkins/workspace/Job_Folder_Deployment_3_main/test3/bin/python3
cachedir: .pytest_cache
rootdir: /var/lib/jenkins/workspace/Job_Folder_Deployment_3_main
collecting ... collected 5 items

sources/test_calc.py::TestCalc::test_add_floats PASSED                   [
20%]
sources/test_calc.py::TestCalc::test_add_integers PASSED                 [
40%]
sources/test_calc.py::TestCalc::test_add_string_and_integer PASSED       [
60%]
sources/test_calc.py::TestCalc::test_add_string_and_number PASSED        [
80%]
sources/test_calc.py::TestCalc::test_add_strings PASSED
[100%]

- generated xml file:
/var/lib/jenkins/workspace/Job_Folder_Deployment_3_main/test-reports/result
s.xml -
============================== 5 passed in 0.03s
```

```
==============================
```