Deployment 3

Kenneth Tan

Successful test build with screenshot

In the jenkins I created a new multibranch pipeline connecting the branch source to my repo, https://github.com/KennethT404/DEPLOY03_TEST, where I have my stored jenkins script.

In my jenkins script, I run a bash script to run test_calc.py in a python virtual environment. It then generates a report based on the tests.

```
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'
                }
         }
}
```

Additional Feature: Subtraction

The additional feature I decided to include into the app was a subtraction function named *sub2*. The goal of this function was to find the difference between two numeric inputs. I added this function into calc.py.

```python
def sub2(var1,var2):
    # Convert 'arg1' and 'arg2' to their appropriate types
    arg1conv = conv(var1)
    arg2conv = conv(var2)
    # If either 'arg1' or 'arg2' is a string, ensure they're both strings.
    if isinstance(arg1conv, str) or isinstance(arg2conv, str):
        return 'cannot subtract non-numeric values'
    return arg1conv - arg2conv
```

In order to test my new function, I had to create new tests for it in the test_calc.py. I had used the same test cases that were used to test add2 but I adjusted the expected result to match the functionality of sub2.

```python
"""
Testing the new feature
"""
def test_sub_integers(self):

    result = calc.sub2(1, 3)
    self.assertEqual(result, -2)

def test_sub_floats(self):

    result = calc.sub2('10.5', 2)
    self.assertEqual(result, 8.5)

def test_sub_strings(self):

    result = calc.sub2('abc', 'def')
    self.assertEqual(result, 'cannot subtract non-numeric values')

def test_sub_string_and_integer(self):

    result = calc.sub2('abc', 3)
    self.assertEqual(result, 'cannot subtract non-numeric values')

def test_sub_string_and_number(self):

    result = calc.sub2('abc', '5.5')
    self.assertEqual(result, 'cannot subtract non-numeric values')
```

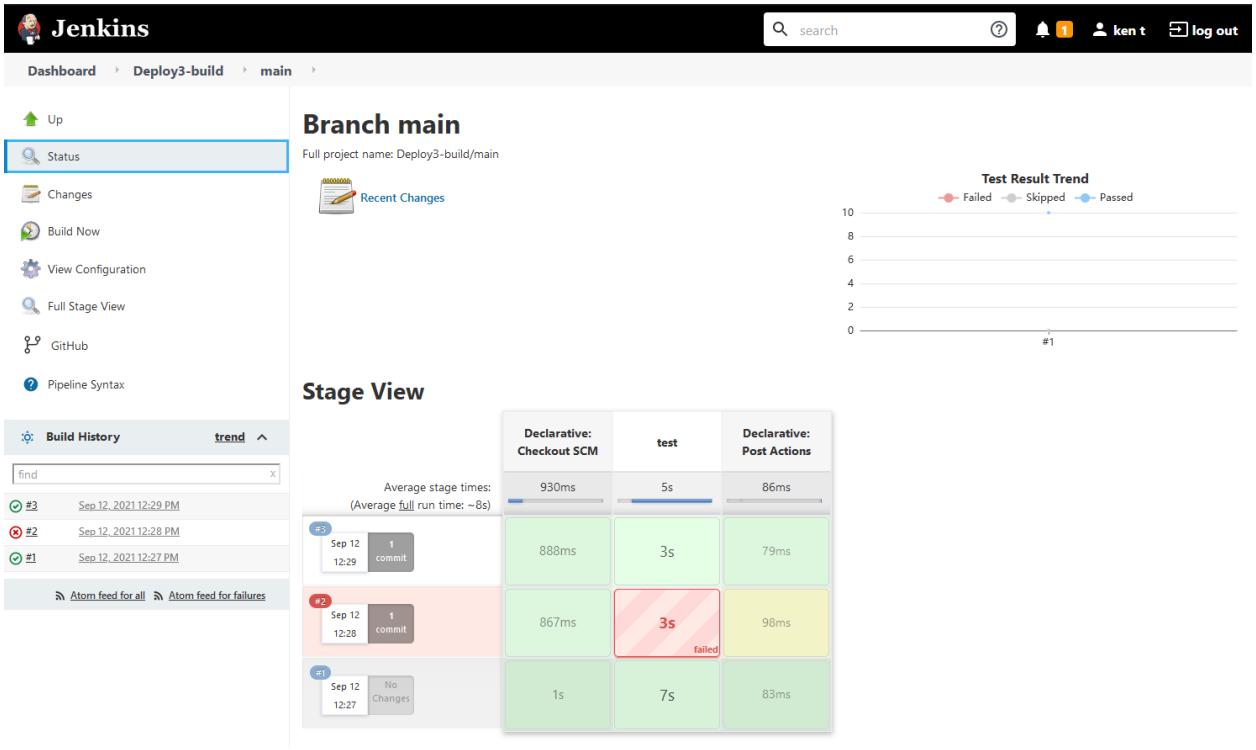
Testing new the feature

When I initially included the subtraction feature to the app, I did not take into account the error in logic that would arise from attempting to perform subtraction on non-numeric values. You cannot subtract “Orange” from “Apple”, nor can you subtract 5 from “Cheese”. To work around this, I realized that add2 has a function that checks inputs for their datatypes before performing operations on them.

In the case with sub2, I would need to check to see if the inputs are numeric or strings. If they were numeric, I can proceed to return the difference, otherwise I will return a message letting the user know their inputs are wrong for this function.

Successful build with screenshot

Below is the build with the addition of the new feature and its corresponding tests.



Build 1 = Initial build that connected this pipeline to the repo and had the test pass for all cases
Build 2 = My initial implementation of the new feature. It had failed due to the logic in the function attempting to subtract strings
Build 3 = Revised function that addresses the issue that came up in Build 2.

*Note: I had done a lot of testing in the IDE so I recreated the scenarios to show my process. This is also why the commits in my repo are few.