**Deployment 3 Documentation**

Objectives

Build a Successful Test

Add New Features

Build a Bad Test

Fix the Test

Building a Successful Test

1. We want to first fork the desired repo with the code we want to test.
2. After forking, we want to create a Jenkinsfile with a pipeline. The pipeline will look like:

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'

}

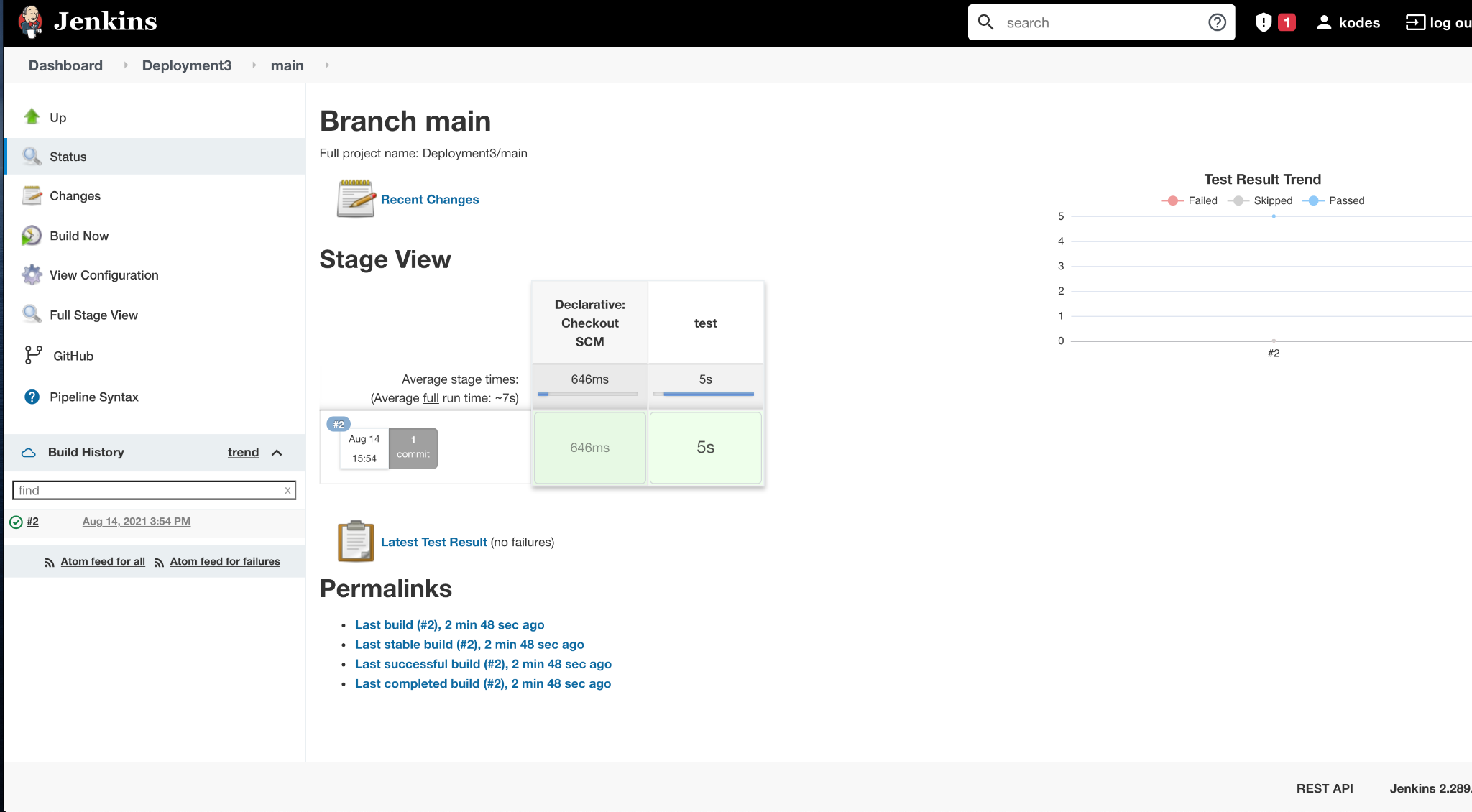
}

}

}

}

Notes: When I initially ran the build, there was a failure. I took a look at the pipeline in the Jenkinsfile and noticed some blatant errors. Always proofread!

1. Boot up your EC2 instance with Jenkins installed. Create a multi-branch pipeline and connect the repo with your Jenkinsfile & code to Jenkins.
2. Build the pipeline and the tests will run successfully!

Adding New Features & Building a Bad Test

I decided to add a minus2 function that would take the difference between the arguments.

1. I added this function to the calc.py program:

def minus2(arg1, arg2):

arg1conv = conv(arg1)

arg2conv = conv(arg2)

if isinstance(arg1conv, str) or isinstance(arg2conv, str):

return "We can't subtract those args, please enter two numbers if you wish to know the difference."

else:

return arg1conv - arg2conv

Note: One issue that arose initially is that you can’t subtract strings! I implemented the if/else statement to resolve this issue.

1. I added this command to the add2val.py program:

print(f"Bonus Info: The difference between the first and the second numbers is {str(calc.minus2(str(sys.argv[1]), str(sys.argv[2]))))}

1. I added this test to the test\_calc.py program

class TestSecondCalc(unittest.TestCase):

"""

Test the minus function from the calc library

"""

def test\_minus\_integers(self):

"""

Test that the subtraction of two integers returns the correct total

"""

result = calc.minus2(1, 2)

self.assertEqual(result, 3)

def test\_minus\_floats(self):

"""

Test that the subtractions of two floats returns the correct result

"""

result = calc.minus2('10.5', 2)

self.assertEqual(result, 12.5)

def test\_minus\_strings(self):

"""

Test the subtraction of two strings returns the two strings as one

concatenated string

"""

result = calc.minus2('abc', 'def')

self.assertEqual(result, 'abcdef')

def test\_minus\_string\_and\_integer(self):

"""

Test the subtraction of a string and an integer returns them as one

concatenated string (in which the integer is converted to a string)

"""

result = calc.minus2('abc', 3)

self.assertEqual(result, 'abc3')

def test\_minus\_string\_and\_number(self):

"""

Test the subtraction of a string and a float returns them as one

concatenated string (in which the float is converted to a string)

"""

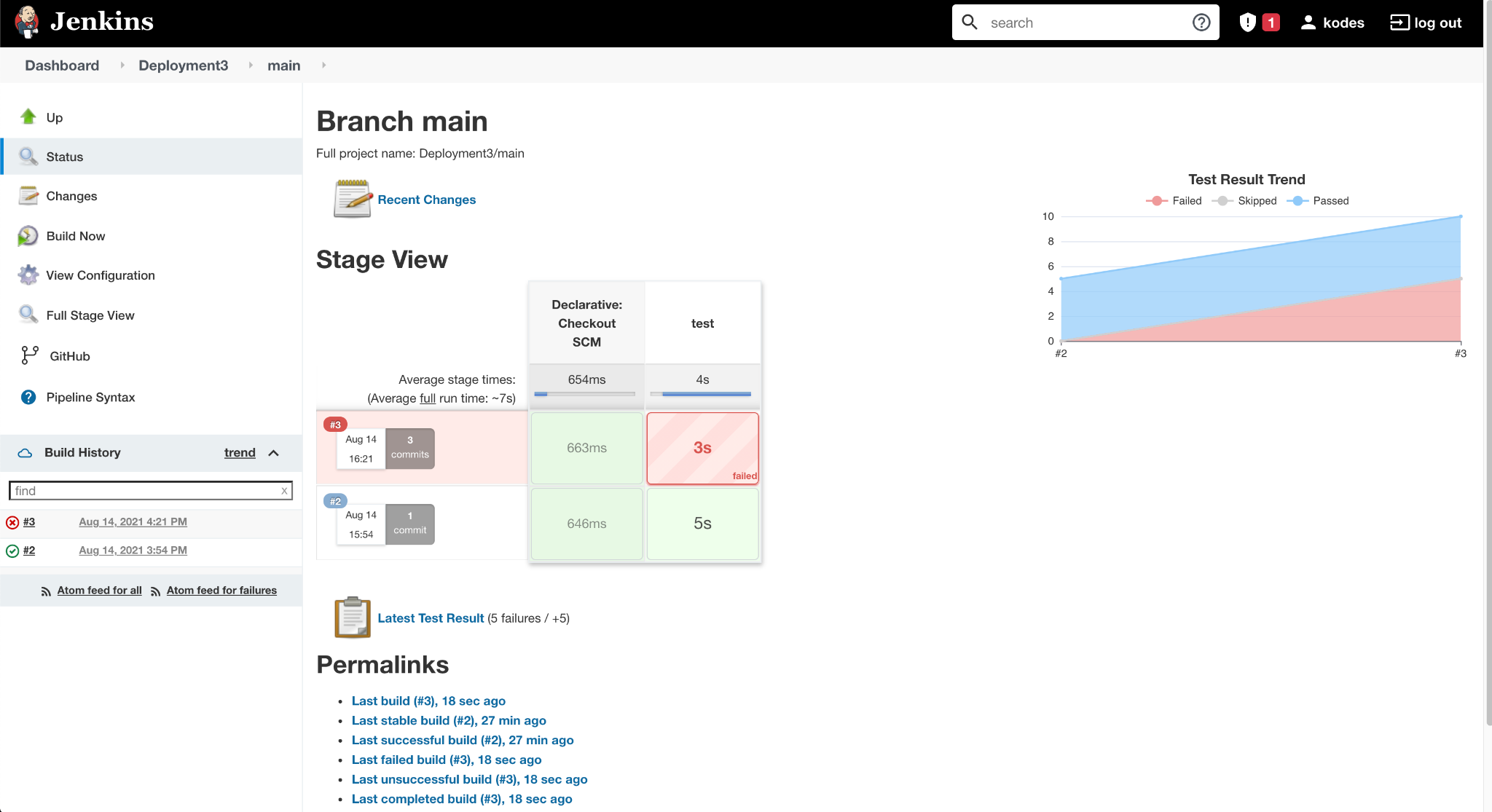
result = calc.minus2('abc', '5.5')

self.assertEqual(result, 'abc5.5')

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

Note: This test is also flawed, but I will go into further detail during the next section. Below is the result of this pipeline build.



Fixing the Test

The test failed! The task was to build a test that would do so, but I actually just ran a test that I thought had some chance of working believing it would fail, because what doesn’t fail the first time?

Upon failing, I inspected the console logs and saw the error was originating from the portion of the test that attempted to subtract strings. This was before I made adjustments to the minus2 function in the calc.py page.

I fixed the minus2 function with the aforementioned if/else statement so that strings would no longer be viable arguments to pass through the function.

I hadn’t changed any of the expected values in the self.assertEqual() functions in the calc\_test.py program. After correcting those, I deleted the functions testing for string subtraction.

With these changes, the test ran successfully.

