

Documentation for Deployment 3

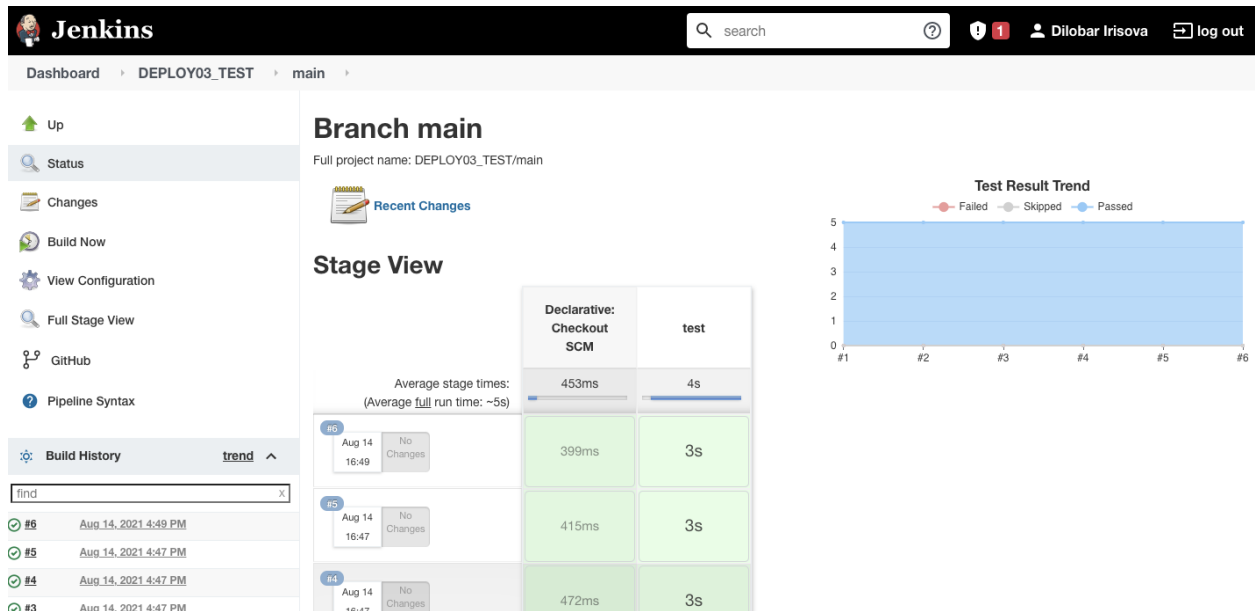
Deployment 3 Requirements:

- 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_TEST 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.

1. The repo was forked https://github.com/DIrisova/DEPLOY03_TEST/
2. I use my already created EC2 instance
3. I create a new ssh key with a new Public IP4 address
4. The repository was cloned to my EC2 instance
5. I created a new Jenkinsfile in my forked repo

```
1 pipeline {
2   agent any
3   stages {
4     stage('test') {
5       steps {
6         sh '''#!/bin/bash
7           python3 -m venv test3
8           source test3/bin/activate
9           pip install pip --upgrade
10          pip install pytest
11          py.test --verbose --junit-xml test-reports/results.xml sources/test_calc.py
12          '''
13       }
14       post {
15         always {
16           junit 'test-reports/results.xml'
17         }
18       }
19     }
20   }
21 }
```

- Then I created a new Jenkins multibranch pipeline and connected it to my github.
- So Jenkins successfully pulled the python code and run the build.



- Added component or feature.

2 contributors

31 lines (26 sloc) | 947 Bytes

```
1 '''
2 A simple command line tool that takes 2 values and adds them together using
3 the calc.py library's 'add2' function.
4 '''
5
6 import sys
7 import calc
8
9 argnumbers = len(sys.argv) - 1
10
11 if argnumbers == 3 :
12     print("")
13     print("The result is " + str(calc.add2(str(sys.argv[1]), str(sys.argv[2])))
14     print("")
15     sys.exit(0)
16
17 elif argnumbers == 2:
18     print("")
19     print("The result is " + str(calc.add2(str(sys.argv[1]), str(sys.argv[2])))
20     print("")
21     sys.exit(0)
22
23 if argnumbers != 3 and argnumbers != 2:
24     print("")
25     print("You entered " + str(argnumbers) + " value/s.")
26     print("")
27     print("Usage: 'add2vals X Y' where X and Y are individual values, or 'subtract3vals' XYZ where xyz are individual values.")
28     print("    If add2vals is not in your path, usage is './add2vals X Y'.")
29     print("    If unbundled, usage is 'python add2vals.py X Y'.")
30     print("")
31     sys.exit(1)
```

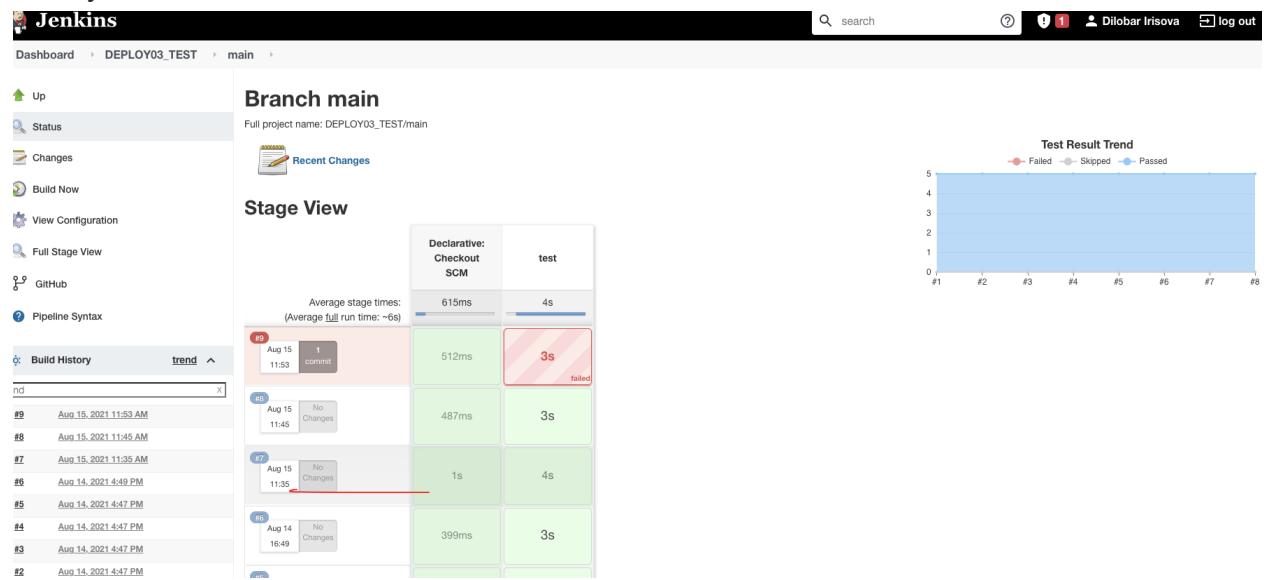
9. In the calc I added a method/function

```
1
2 The 'calc' library contains the 'add2' function that takes 2 values and adds
3 them together. If either value is a string (or both of them are) 'add2' ensures
4 they are both strings, thereby resulting in a concatenated result.
5 NOTE: If a value submitted to the 'add2' function is a float, it must be done so
6 in quotes (i.e. as a string).
7 '''
8
9 # If 'value' is not an integer, convert it to a float and failing that, a string.
10 def conv(value):
11     try:
12         return int(value)
13     except ValueError:
14         try:
15             return float(value)
16         except ValueError:
17             return str(value)
18
19 # The 'add2' function itself
20 def add2(arg1, arg2):
21     # Convert 'arg1' and 'arg2' to their appropriate types
22     arg1conv = conv(arg1)
23     arg2conv = conv(arg2)
24     # If either 'arg1' or 'arg2' is a string, ensure they're both strings.
25     if isinstance(arg1conv, str) or isinstance(arg2conv, str):
26         arg1conv = str(arg1conv)
27         arg2conv = str(arg2conv)
28     return arg1conv + arg2conv
29
30 def subtract2(arg1, arg2, arg3):
31     arg1conv = conv(arg1)
32     arg2conv = conv(arg2)
33     arg3conv = conv(arg3)
34
35     if isinstance(arg1conv, str) or isinstance(arg2conv, str) or isinstance(arg3conv, str):
36         arg1conv = str(arg1conv)
37         arg2conv = str(arg2conv)
38         arg3conv = str(arg3conv)
39     return arg1conv - arg2conv - arg3conv
```

10. Test in test_calc.py

```
22
23     def test_add_strings(self):
24         """
25         Test the addition of two strings returns the two strings as one
26         concatenated string
27         """
28         result = calc.add2('abc', 'def')
29         self.assertEqual(result, 'abcdef')
30
31     def test_add_string_and_integer(self):
32         """
33         Test the addition of a string and an integer returns them as one
34         concatenated string (in which the integer is converted to a string)
35         """
36         result = calc.add2('abc', 3)
37         self.assertEqual(result, 'abc3')
38
39     def test_add_string_and_number(self):
40         """
41         Test the addition of a string and a float returns them as one
42         concatenated string (in which the float is converted to a string)
43         """
44         result = calc.add2('abc', '5.5')
45         self.assertEqual(result, 'abc5.5')
46
47     def test_subtract_integers(self):
48
49         result = calc.subtract2(1, 2)
50         self.assertEqual(result, -1)
51
52     def test_subtract_floats(self):
53         """
54         Test that the addition of two floats returns the correct result
55         """
56         result = calc.subtract2('7.6', 2)
57         self.assertEqual(result, 5.6)
58
59 if __name__ == '__main__':
60     unittest.main()
```

11. My failed test build.



12. Successful test after fixed.

