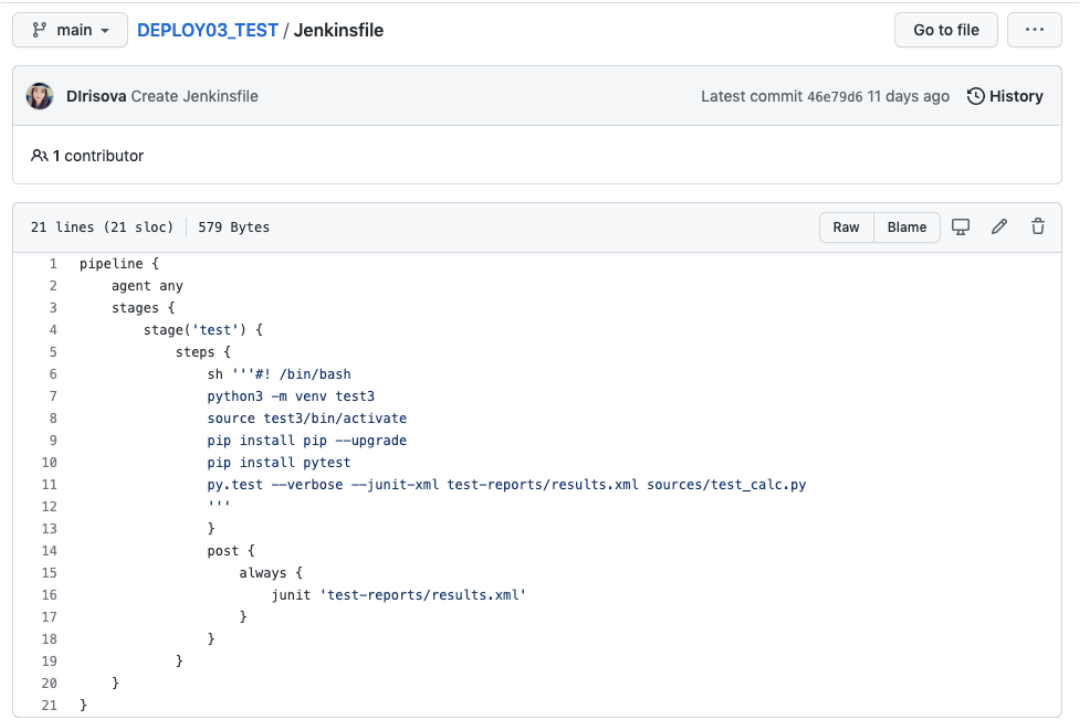
Documentation for Deployment 3

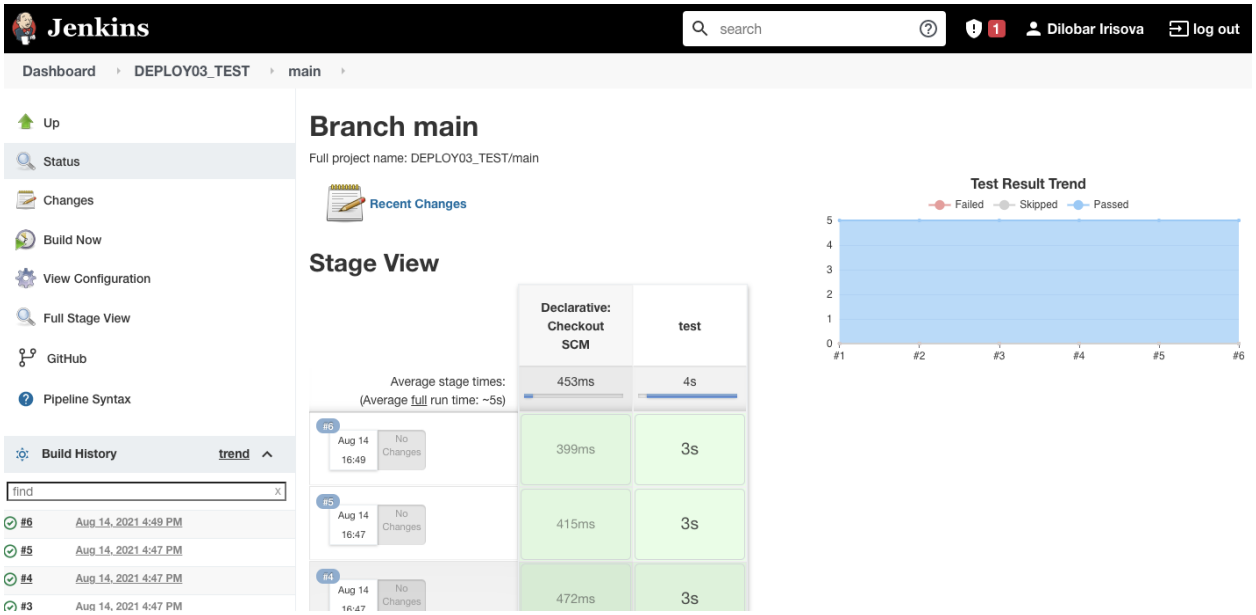
Deployment 3 Requirements:

- 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



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



8. Added component or feature.

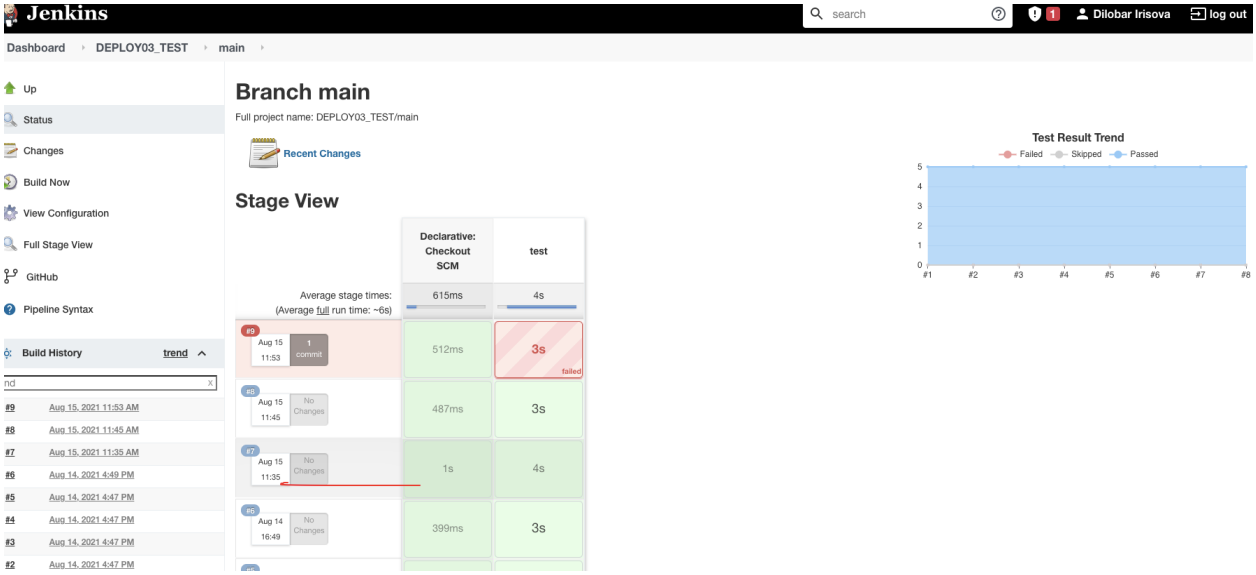
9. In the calc I added a method/function

```
'''
The 'calc' library contains the 'add2' function that takes 2 values and adds
them together. If either value is a string (or both of them are) 'add2' ensures
they are both strings, thereby resulting in a concatenated result.
NOTE: If a value submitted to the 'add2' function is a float, it must be done so
in quotes (i.e. as a string).
'''

# 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):
    # Convert 'arg1' and 'arg2' to their appropriate types
    arg1conv = conv(arg1)
    arg2conv = conv(arg2)
    # If either 'arg1' or 'arg2' is a string, ensure they're both strings.
    if isinstance(arg1conv, str) or isinstance(arg2conv, str):
        arg1conv = str(arg1conv)
        arg2conv = str(arg2conv)
    return arg1conv + arg2conv

def subtract2(arg1, arg2, arg3):
    arg1conv = conv(arg1)
    arg2conv = conv(arg2)
    arg3conv = conv(arg3)

    if isinstance(arg1conv,str) or isinstance(arg2conv, str) or isinstance(arg3conv, str):
        arg1conv = str(arg1conv)
        arg2conv = str(arg2conv)
        arg3conv = str(arg3conv)
    return arg1conv - arg2conv - arg3conv
```

10. Test in test_calc.py

```
    def test_add_strings(self):
        """
        Test the addition of two strings returns the two strings as one
        concatenated string
        """
        result = calc.add2('abc', 'def')
        self.assertEqual(result, 'abcdef')

    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)
        self.assertEqual(result, 'abc3')

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

    def test_subtract_integers(self):

        result = calc.subtract2(1, 2)
        self.assertEqual(result, -1)

    def test_subtract_floats(self):
        """
        Test that the addition of two floats returns the correct result
        """
        result = calc.subtract2('7.6', 2)
        self.assertEqual(result, 5.6)

if __name__ == '__main__':
    unittest.main()
```

11. My failed test build.



12. Successful test after fixed.