

Deployment #2

Step 1: I installed Jenkins on an EC2

Step 2: I activated the Jenkins user on the EC2

```
$sudo passwd jenkins
```

```
$sudo su - jenkins -s /bin/bash
```

Step 3: Created a Jenkins user in my AWS account through IAM

aws Services Search for services, features, blogs, docs, and more [Alt+S]

EC2

Add user

1 2 3 4 5

✓ **Success**

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://tech-t.signin.aws.amazon.com/console>

Download .csv

	User	Access key ID	Secret access key
▶	✓ EB-user	AKIAT4F577X7WOLMS673	***** Show

Close

Step 4: Install AWS CLI on the Jenkins EC2 and configure:

- Using the curl command I downloaded AWS CLI files from the AWS CLI website
- Issues: I had difficulty with this step because the user was not part of the sudoers list and unzip was not installed on the jenkins user
- So I had to exit the jenkins user, and add jenkins to sudoers on the ubuntu EC2.
- Then as a Jenkins user I installed the unzip command and proceeded to configure.

```
$sudo usermod -aG sudo jenkins
$sudo su - jenkins -s /bin/bash
$curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
$sudo apt install unzip
```

Step 5: Install EB CLI in the jenkins EC2 user:

- **Issues: I tried to install EB CLI but my jenkins user did not have the pip command**
- So I had to install the pip command using `$sudo apt install python3-pip`
- Then, I used the below commands to install the EB CLI in the Jenkins EC2
- `$nano .bashrc`
- I entered the following commands into the bash.rc `export PATH="/var/lib/jenkins/.local/bin:$PATH"`
- I then was able to install AWS EB CLI with the below command

```
$pip install awsebcli --upgrade --user
$eb --version
```

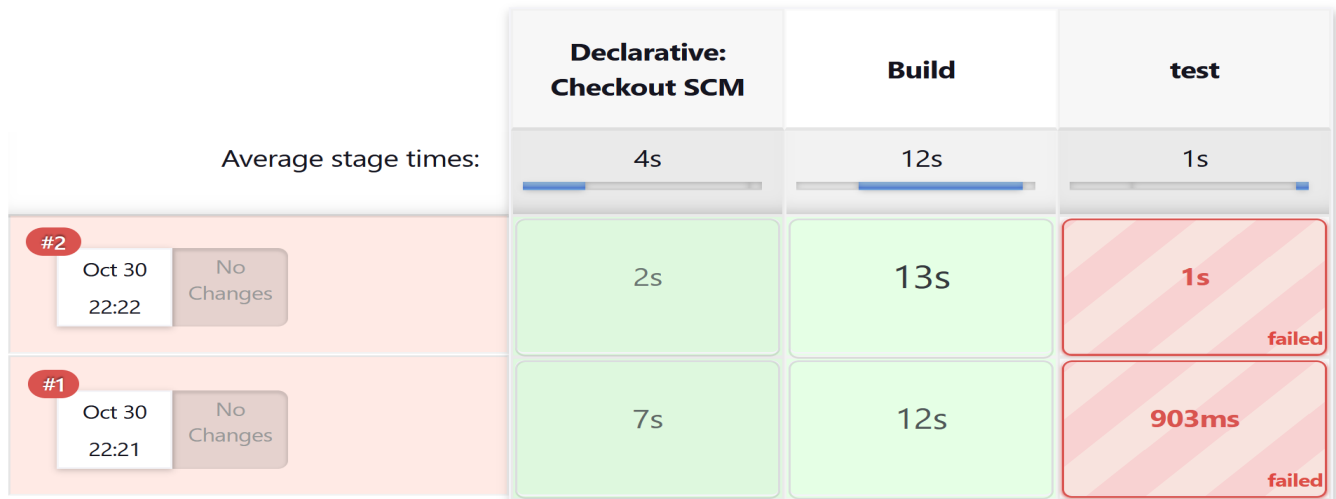
Step 6: Connect GitHub to Jenkins Server:

- I first forked the Deployment repo:
https://github.com/kura-labs-org/kuralabs_deployment_2.git
- Next, I connected my Github credentials to Jenkins using an access token from GitHub.

Step 7: I built a multibranch pipeline on Jenkins

- I clicked Build Now. AND found that my first two builds failed.

Stage View

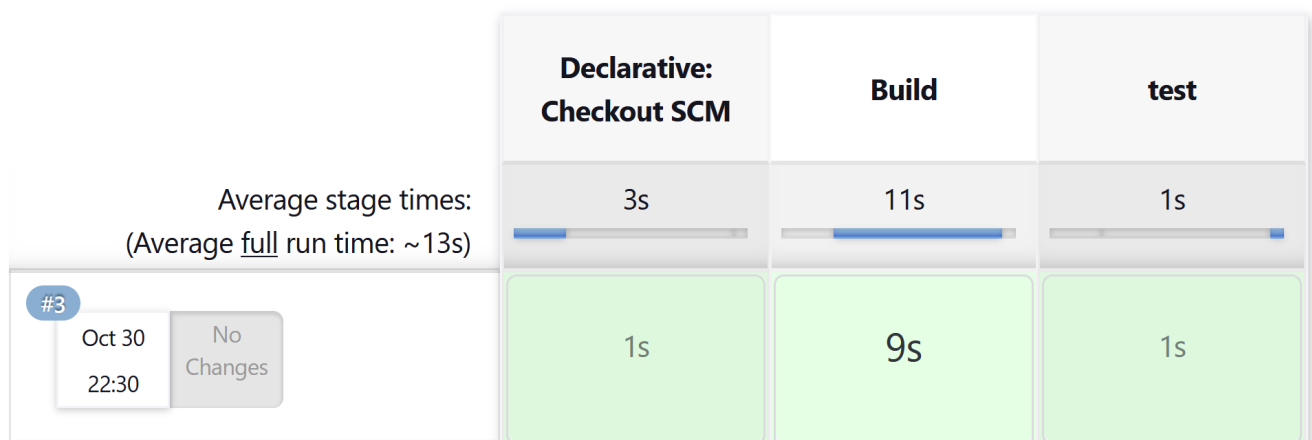


- I checked the logs and found that I needed to install python on my EC2.

```
ubuntu@ip-172-31-31-68:~$ sudo apt install python3.10-venv
```

- After I installed python, I ran the Build again and the Build was successful.

Stage View



Step 8: Deploy the application from Elastic Beanstalk CLI:

- I used sudo to initialize the jenkins user again.
- I changed to directories in order to enter the workspace where the url-shortener_main is located
- I ran \$pip install awsebcli --upgrade --user
- Then, I ran export PATH="/var/lib/jenkins/.local/bin:\$PATH"

```
$sudo su - jenkins -s /bin/bash
```

```
$cd workspace/url-shortener_main/
```

```
$eb init
```

- Select: us-east-1
- Press enter
- Select application to use: url-shortener3
- Select: Python
- Select: (The latest version of python available)
- Select: N (for CodeCommit)
- Set up SSH for instance: Yes
- Select keypair: I selected the keypair ssh2

```
$eb create
```

- I entered the default for the next 3 questions by hitting enter.
- My environment name was url-shortener-dev.

- Spot Fleet: No
- I waited for the application to be made.

Elastic Beanstalk > Applications

All applications Actions ▾ Create a new application

Filter results matching the display values < 1 > ⚙

Application name ▲	Environments ▾	Date created ▾	Last modified ▾	ARN ▾
<input type="radio"/> url-shortener		2022-10-30 18:36:22 UTC-0400	2022-10-30 18:36:22 UTC-0400	arn:aws:elasticbeanstalk:us-east-1:380668261599:application/url-shortener

Step 9 I added a deployment stage to the pipeline in the Jenkinsfile:

```

pipeline {
  agent any
  stages {
    stage ('Build') {
      steps {
        sh '''#!/bin/bash
        python3 -m venv test3
        source test3/bin/activate
        pip install pip --upgrade
        pip install -r requirements.txt
        export FLASK_APP=application
        flask run &
        '''
      }
    }
    stage ('test') {
      steps {
        sh '''#!/bin/bash
        source test3/bin/activate
        py.test --verbose --junit-xml test-reports/results.xml
        '''
      }
    }
  }
  post{
    always {
  
```

```
        junit 'test-reports/results.xml'
    }

}
stage ('Deploy') {
    steps {
        sh '/var/lib/jenkins/.local/bin/eb deploy url-shortener-dev'
    }
}
}
```

-

Step 10: I added an extra test to the pipeline.

- I added the below code to the test_app.py file

```
# def test_fast():  
#     b = "Bobby"  
#     new_greeting = greet(b)  
#     assert new_greeting == "Good Evening Bobby"
```

```
def test_home_page():  
  
    response = app.test_client().get('/')  
    assert response.status_code == 200
```

- I built the pipeline again and the test ran successfully.

What could have been done differently:

- To install the eb command I could have added the path to the bash.rc file so that eb would have been added to my path. Since I only installed the eb on the Jenkins user, it needed to be reinstalled every time I stopped and restarted the instance.
- To prevent my initial build from failing, I should have installed python on my EC2, especially because the url-shortener is a python application.