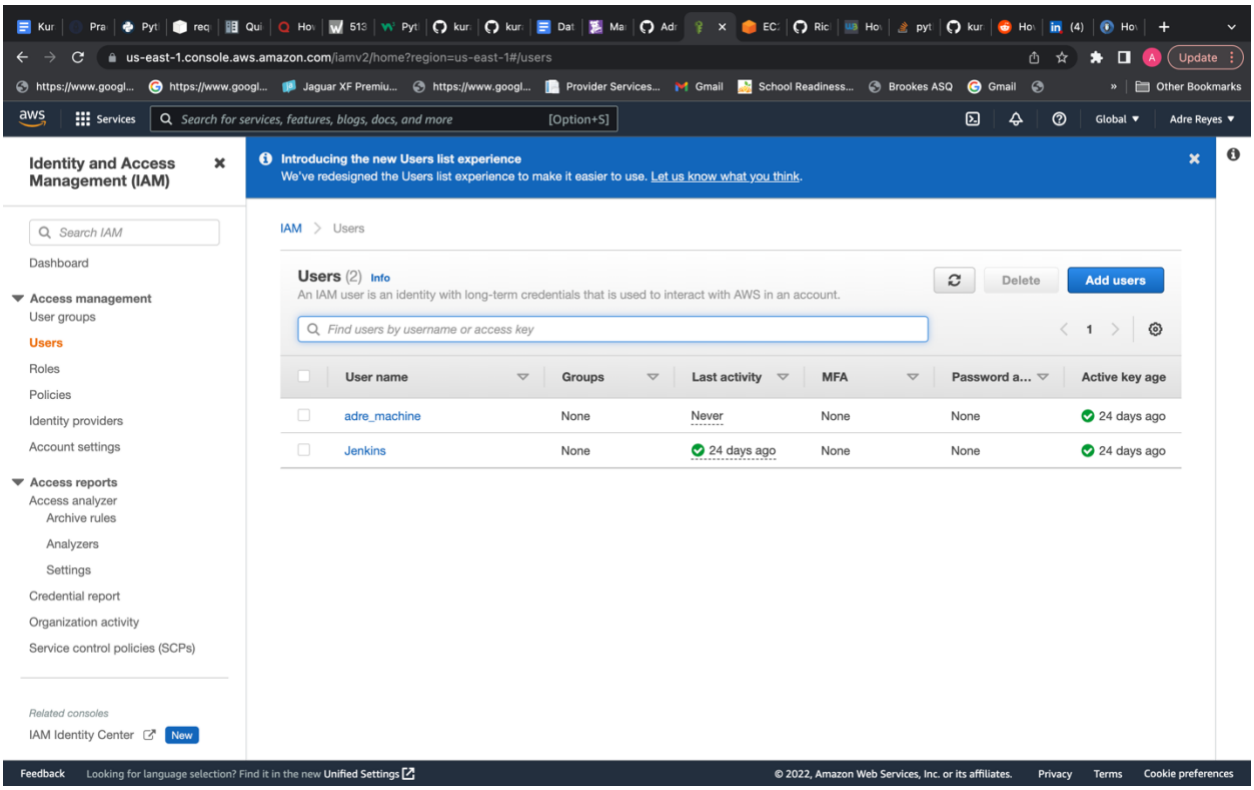
Documentation

- First, I created an EC2 Instance with ports 22, 80, and 8080 open to access Jenkins server

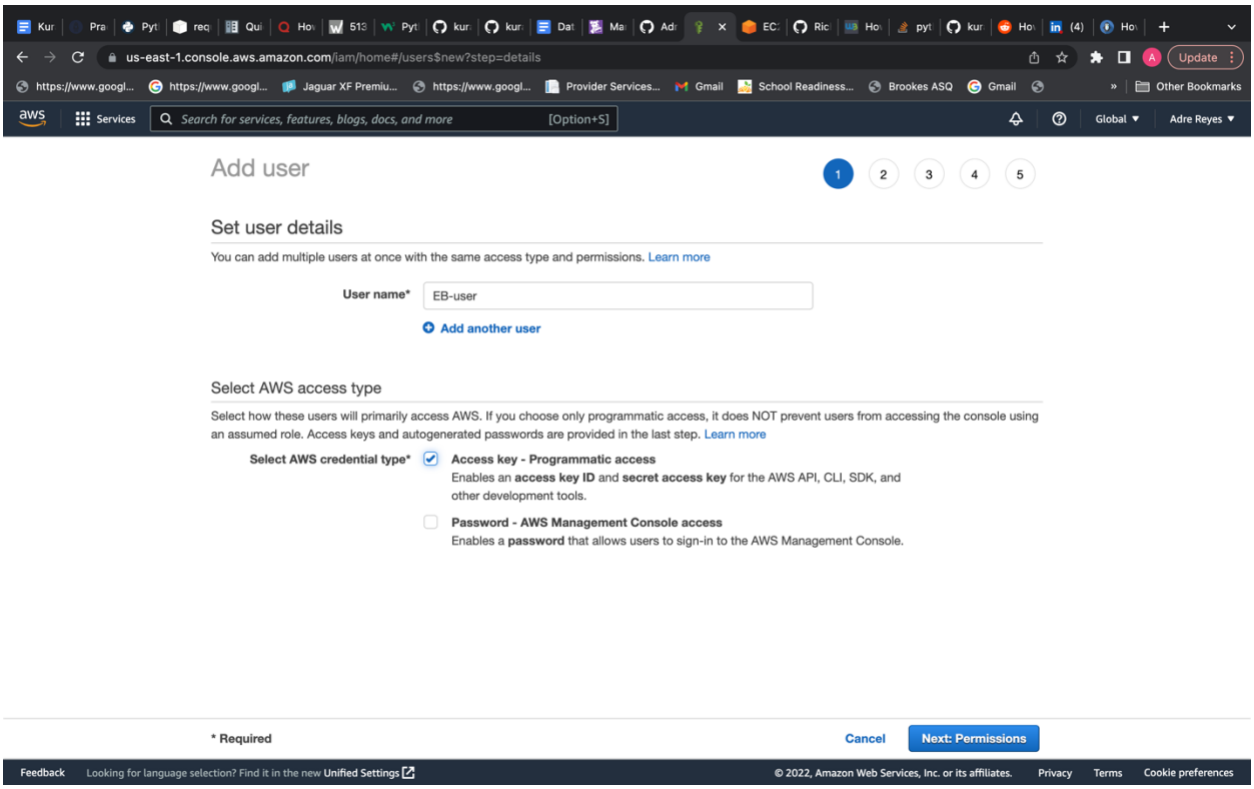
Type	Protocol	Port range	Source
Custom TCP	TCP	8080	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0

- I SSHed into the EC2 from Ubuntu virtual machine via this line: <ssh -i pemfile ubuntu@publicipaddress>. Switched over to Jenkins user by running the command <sudo su - jenkins -s /bin/bash>

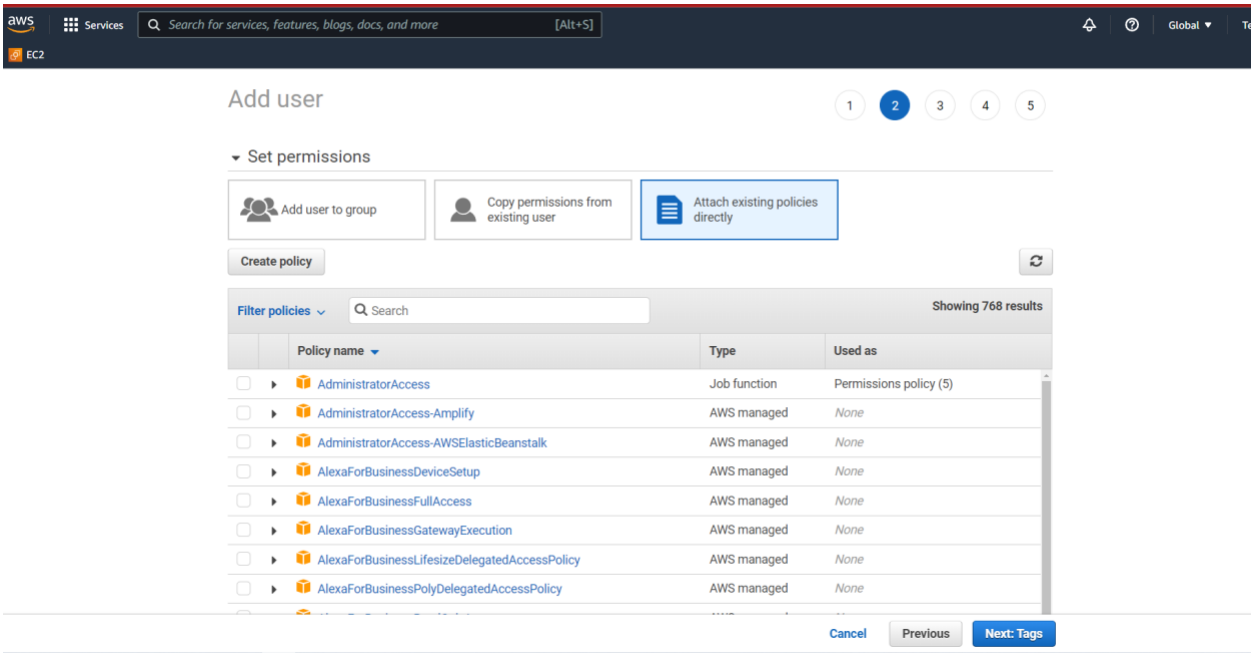
- I went over to IAM in the AWS console and create an IAM user to grant the Jenkins applications on the Amazon EC2 instance access to the Flask app being deployed via Elastic Beanstalk



- Selected programmatic access where you will receive an Access Key ID and Secret Access Key (I made sure to save it so that I can use it when prompted during AWS configuration)



- Selected “Attach existing policies directly” and selected administrator access.



- Installed AWS CLI (command line interface) on the via my ubuntu user by running <curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip” -o "awscliv2.zip”> The -

o option specifies the file name that the downloaded package is written to, in this case its written to the current directory with the local name awscliv2.zip.

- Then I ran <unzip awscliv2.zip> which unzips the package and creates a directory named aws under the current directory. Extracting the downloaded package enabled me to interact with AWS services using commands in the command-line shell and the functionality of the commands are equivalent to the browser-based AWS Management Console. Then I ran these commands:

```
$sudo ./aws/install
```

- <aws –version> just shows me what version I have in my terminal.
- I ran <aws configure> so that I could configure the settings that the AWS CLI uses to interact with AWS. These include my security credentials, the default output format, and the default AWS Region. I saved my Access Key ID and Secret Access Key just for this.

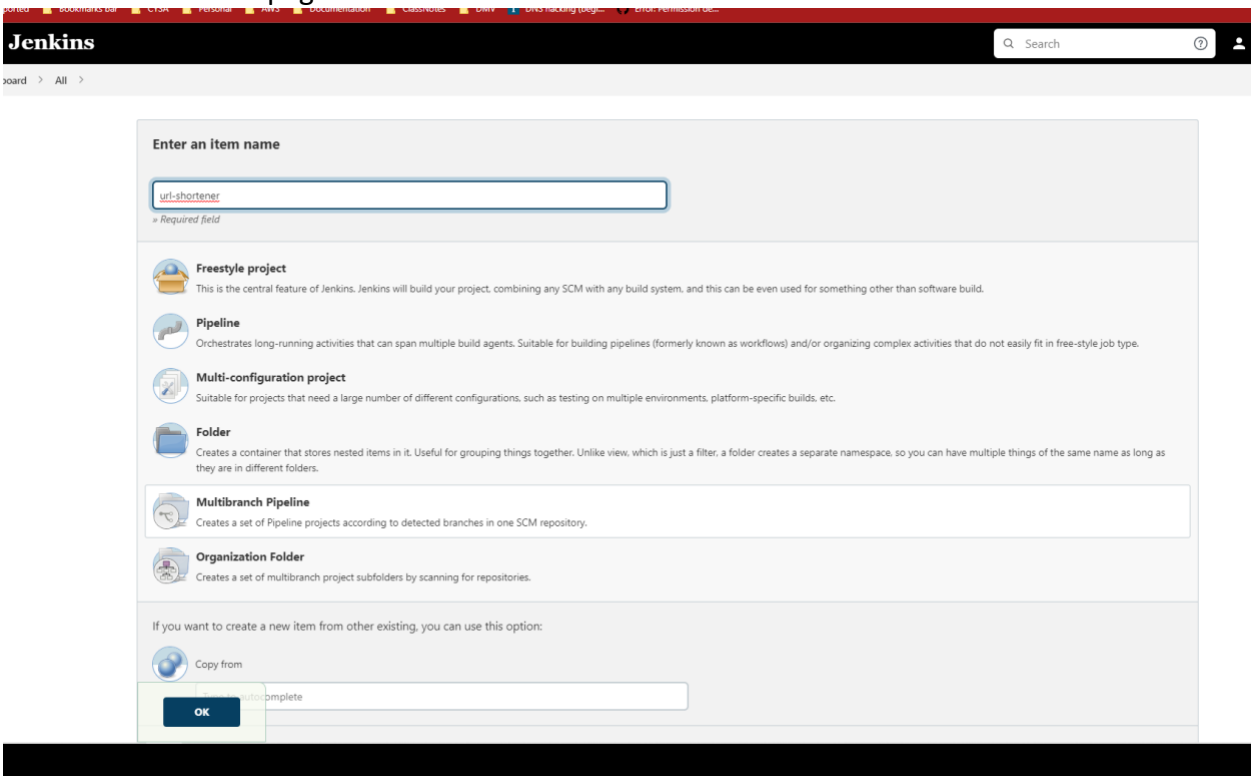
```
$aws configure
  - Set Access Key ID
  - Set Secret Access Key
  - Set region to: us-east-1
  - Set Output format: json
```

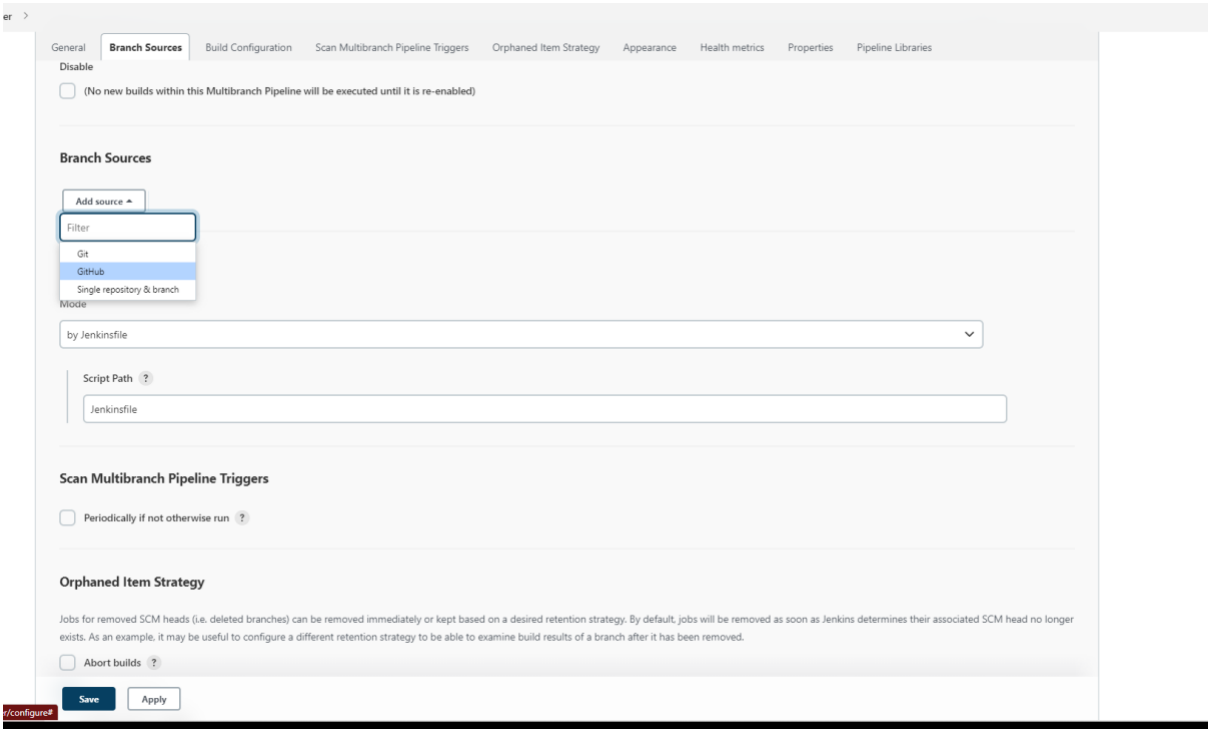
- I installed Elastic Beanstalk Command Line Interface (EB CLI) (I had to keep in mind the path where the eb file is located for later) via the commands below. However, when I ran eb - -version the terminal didn't recognize "eb" as a command when I ran the last command so I couldn't see the version of Elastic Beanstalk:

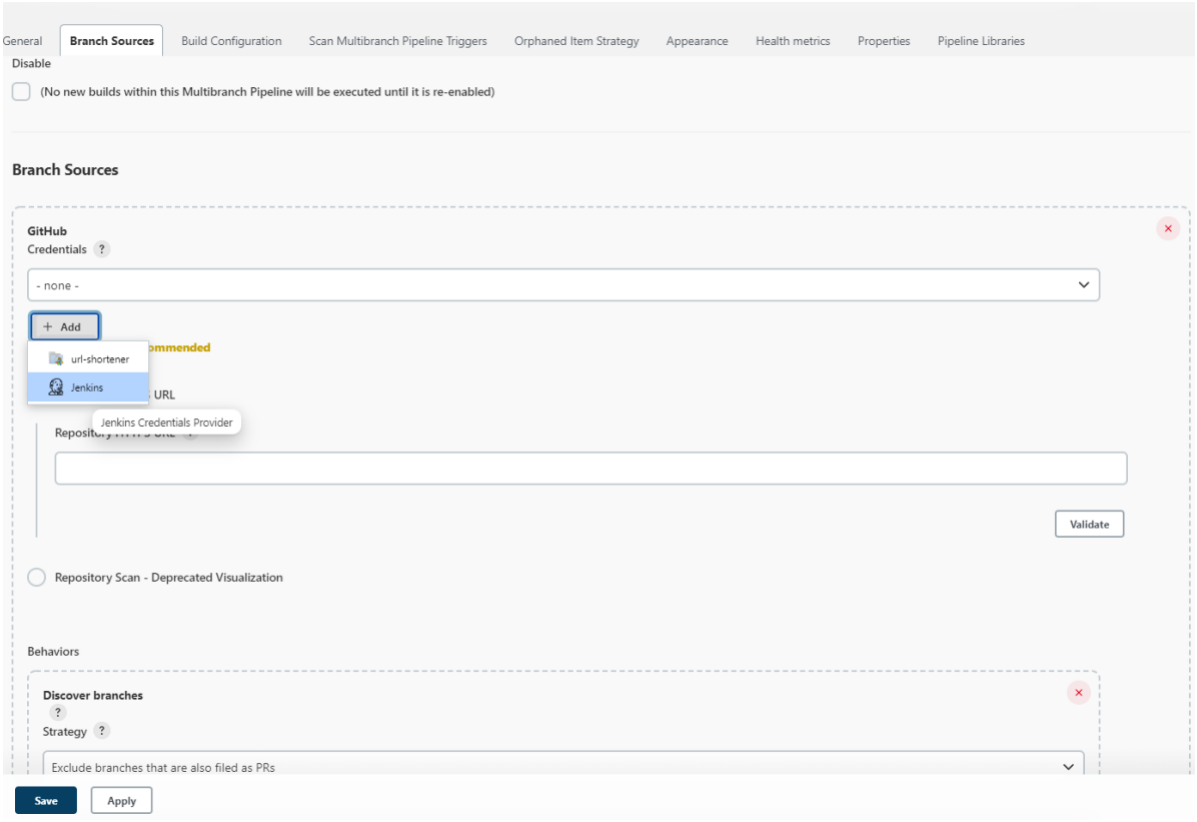
```
$pip install awsebcli --upgrade --user
$eb –-version
```

- Next, I forked the deployment repo and connected Github to the Jenkins server when I set up my multi build branch and provided the Github access token under the "Branch

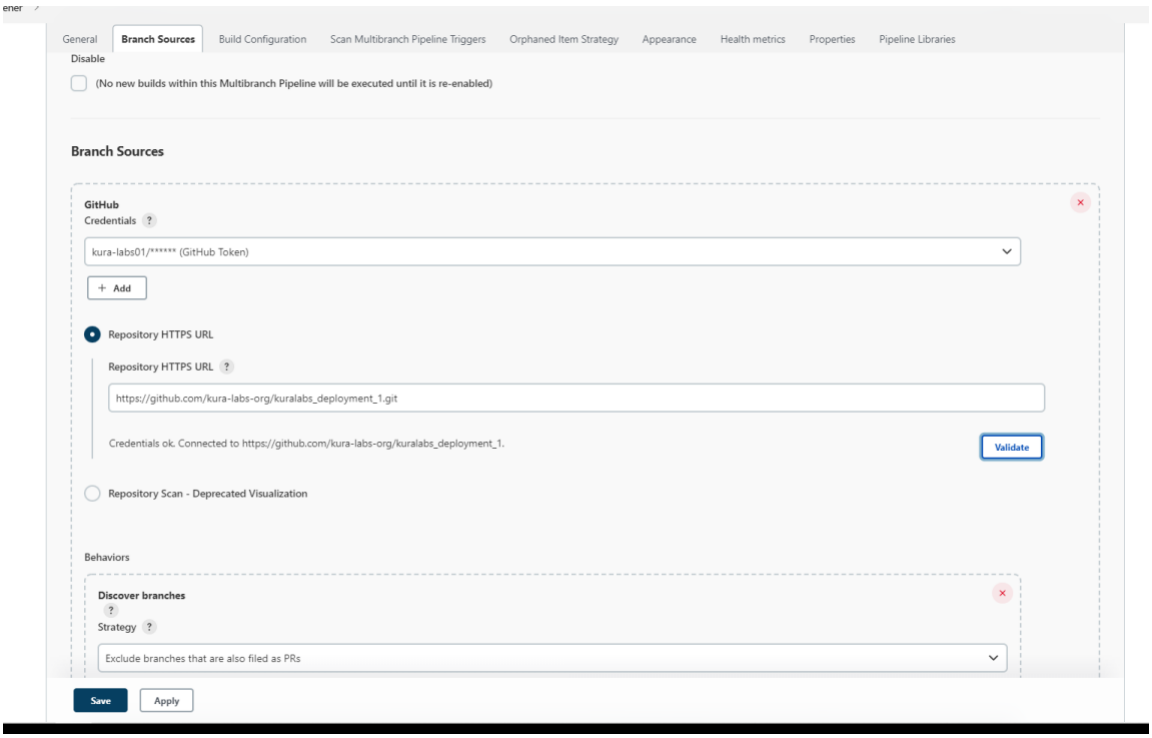
Sources” option >> selected “Github” >> selected “Jenkins” >> entered info for ”Github Credentials” page



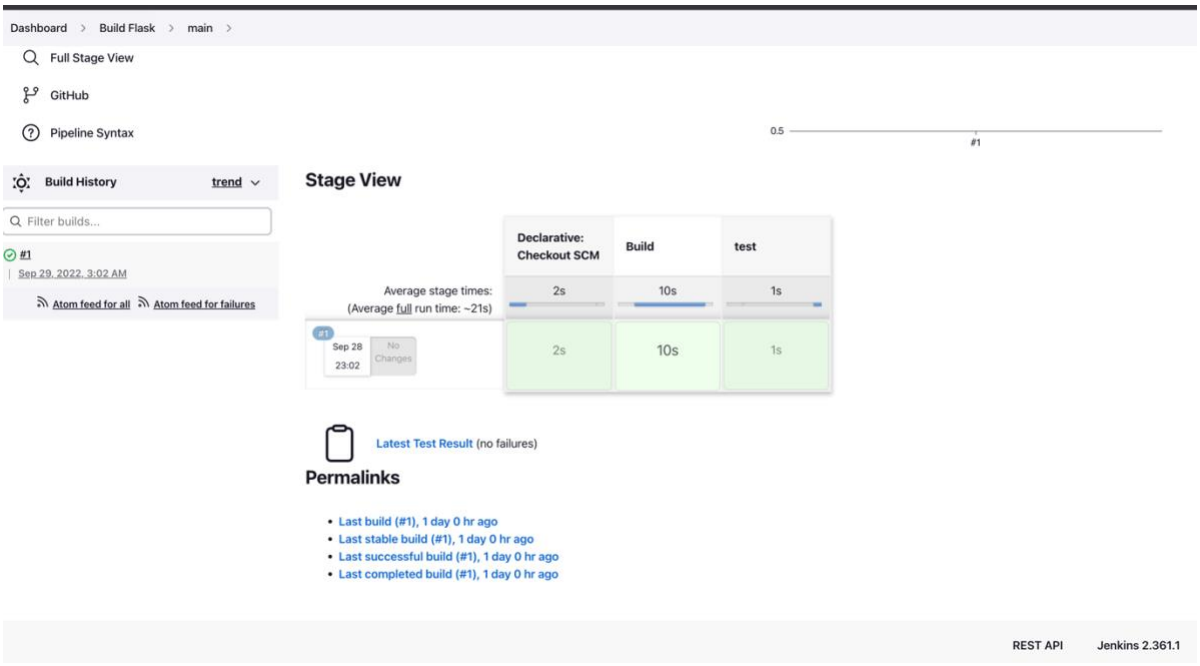




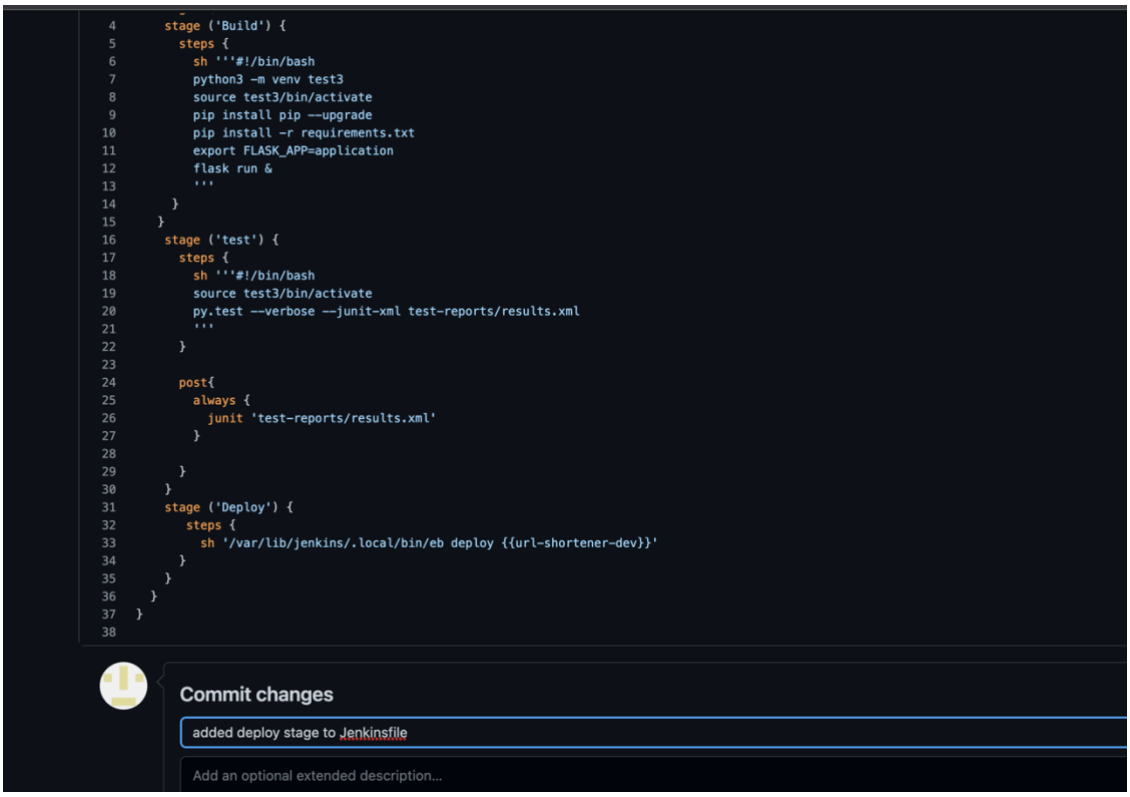
- Entered Deployment 2 URL to the repository and validated by selecting validate >> selected “Apply” >> selected “Save”

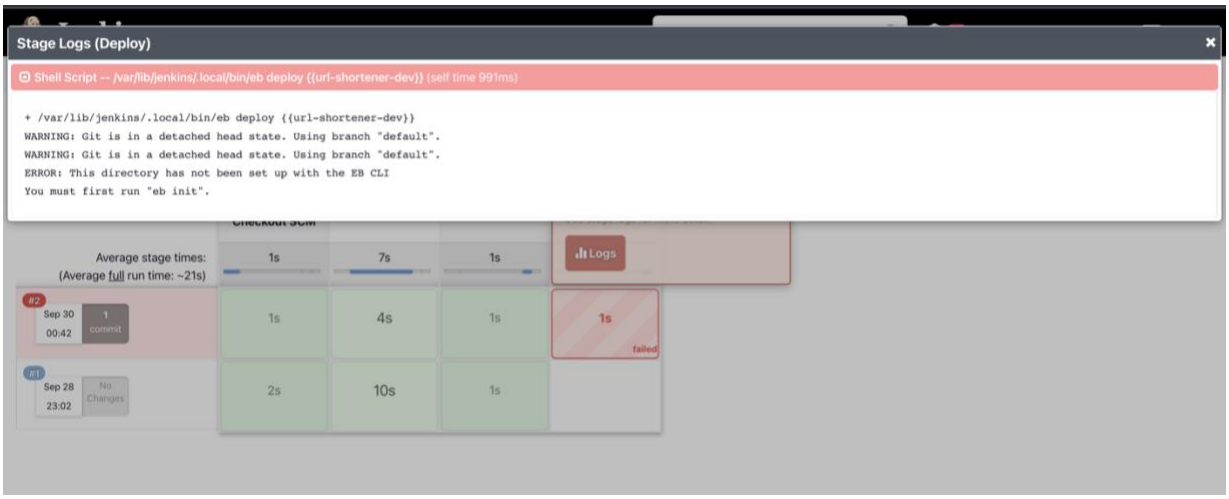


- I created my build which was successful.

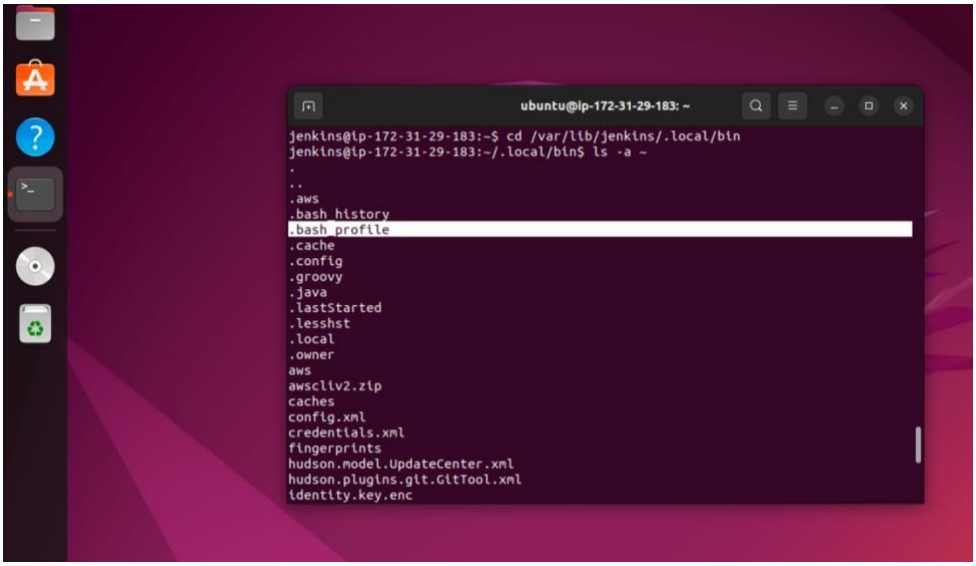


- Lastly, I added the deploy stage in my “Jenkinsfile” but the deployment failed.

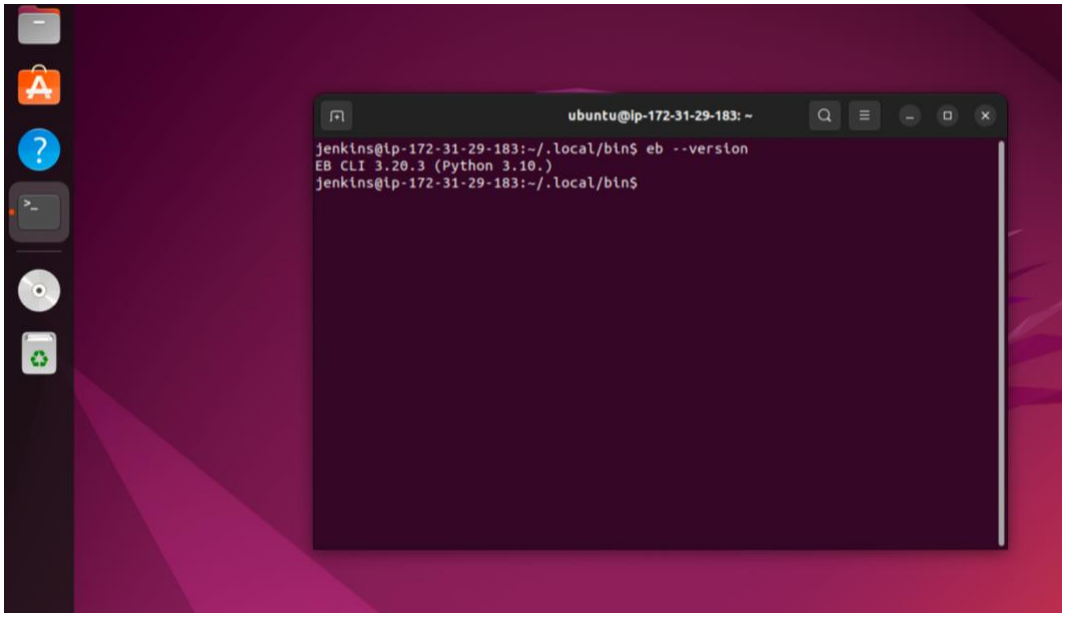




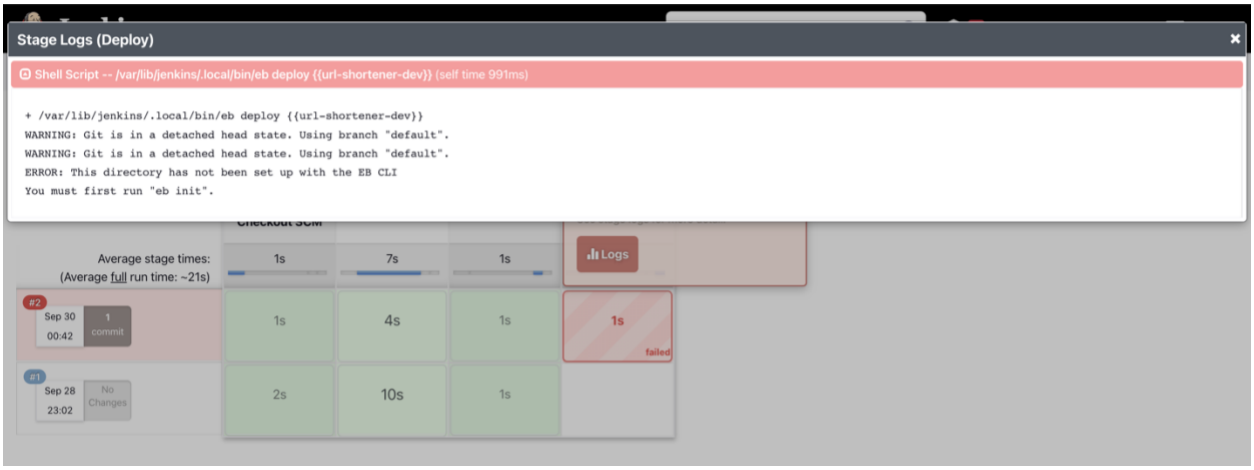
- Why? Because I didn't add the eb executable file to my shell's profile script in the user folder so it could execute Elastic Beanstalk CLI. I actually didn't have a profile script so I made one with <nano .bash_profile> and placed the path in there via the <export PATH=LOCAL_PATH:$PATH> that I received when I installed EB CLI. I then loaded the profile script into my session via <source ~/>.



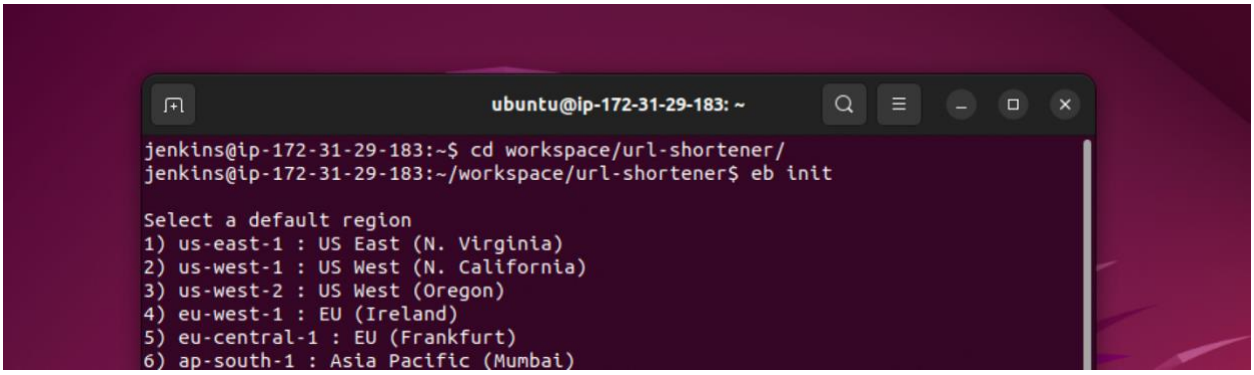
- To make sure EB CLI was installed correctly I ran eb - -version and this time it showed me the version of eb I had.



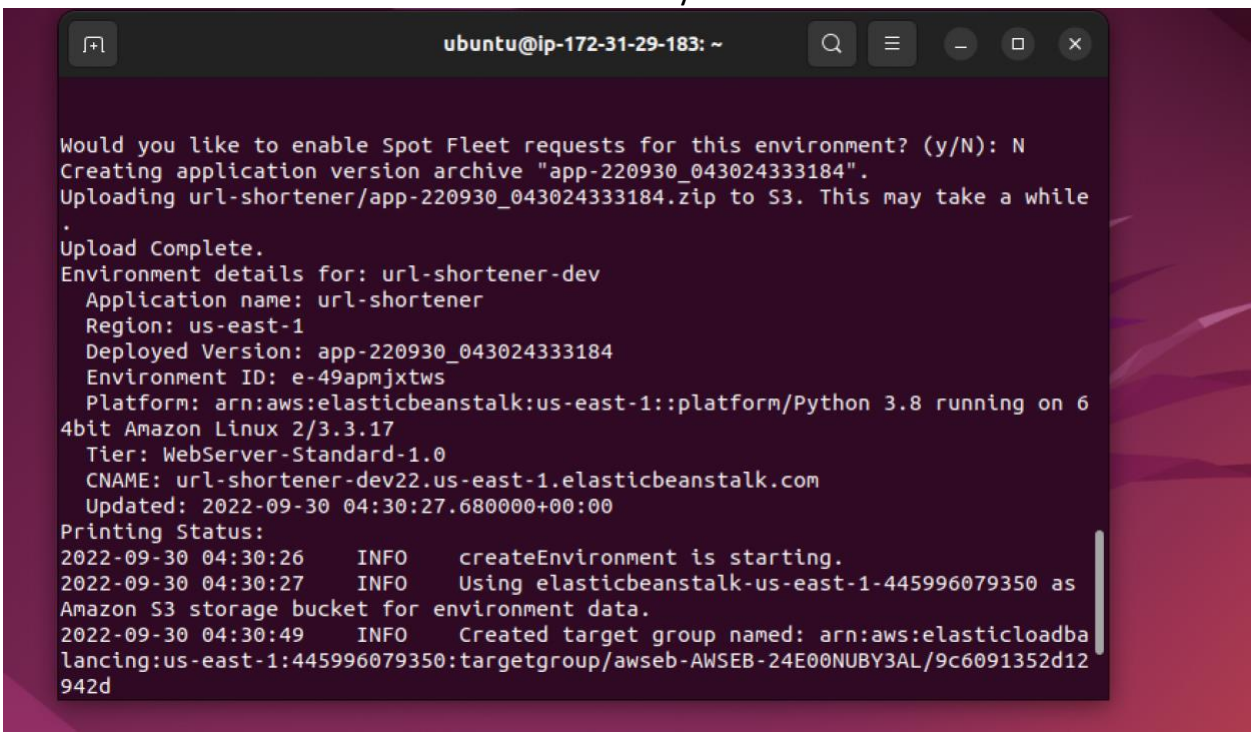
- Equally important, I had to create my environment with the <eb init> and <eb create> command in my /var/lib/jenkins/workspace/url-shortener directory (where url-shortener-main is located which is the name of my repo) directory because without it the eb command was unrecognized and I kept getting the error: “This directory has not been set up with the EB CLI” when I would deploy with Jenkins.



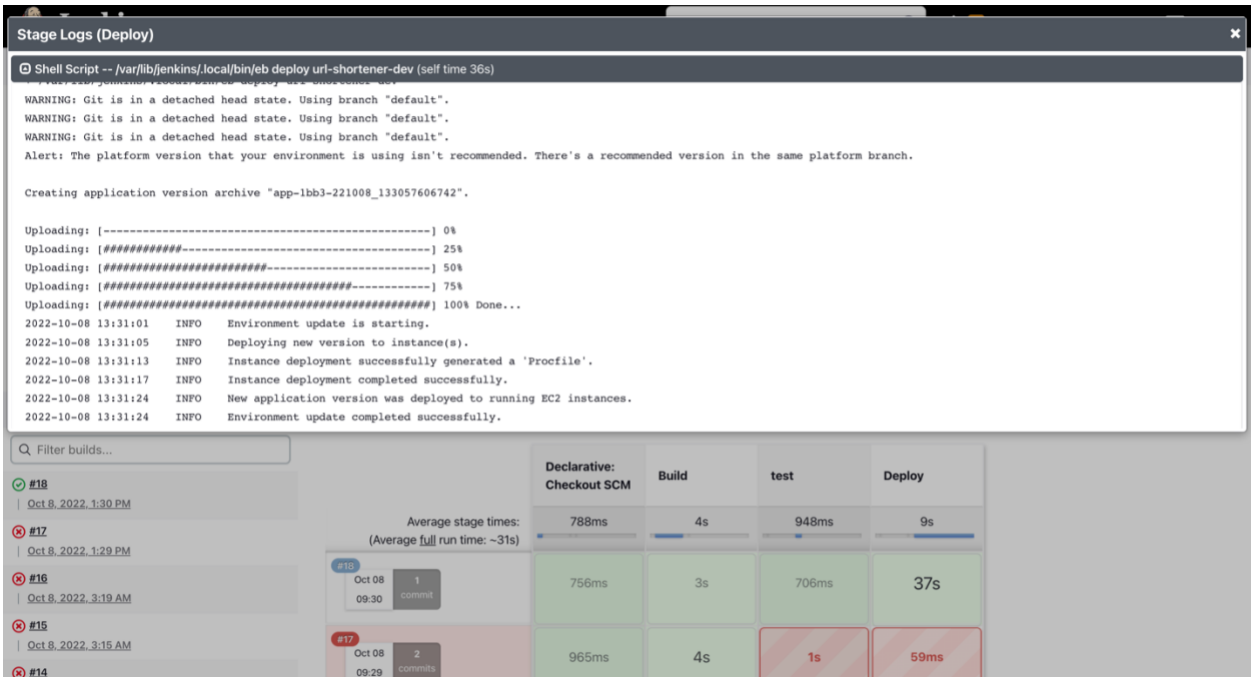
- Why? Because I didn’t open up my elastic beanstalk environment from the command line and create my flask application in the directory that my Git project was in so that’s why it didn’t recognize the command “eb” in the deploy stage in my Jenkinsfile in Github until **after** I did those steps.



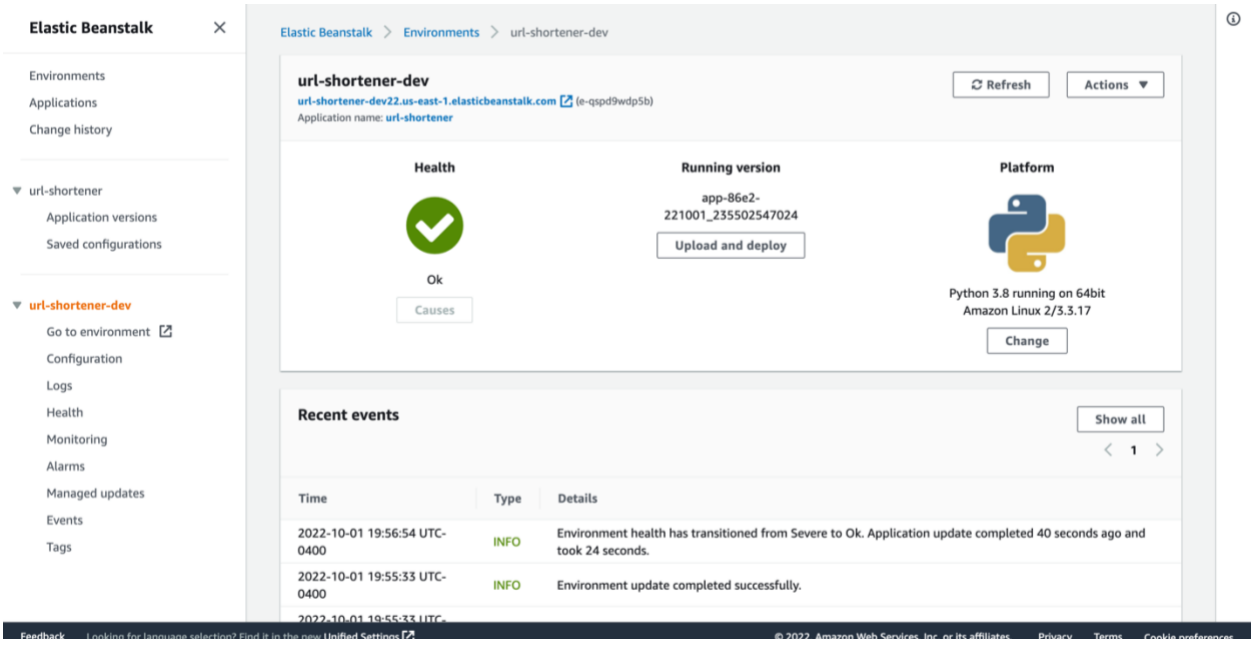
- The environment was created successfully. Refer to the screenshot below.



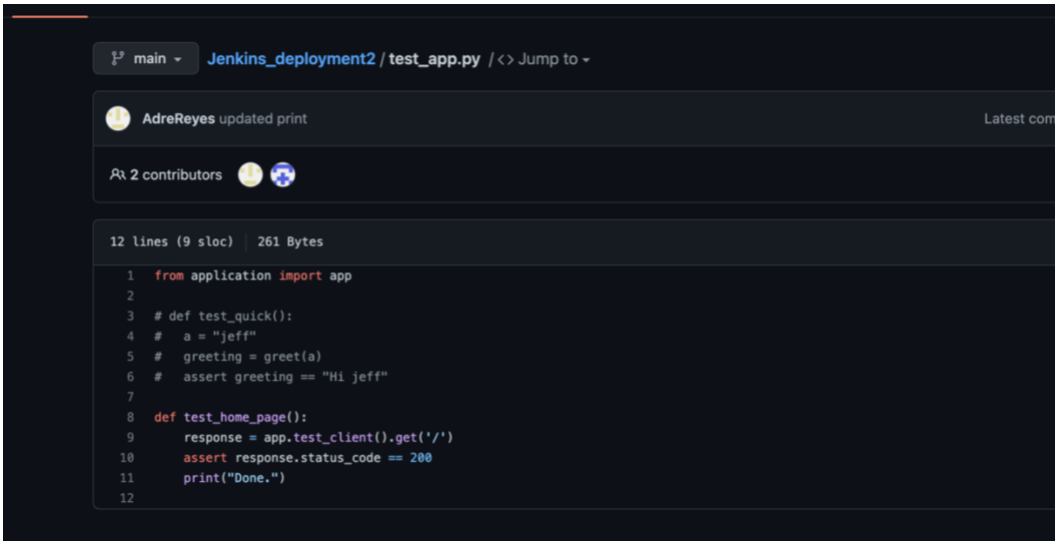
- Deployment stage finally ran successfully in Jenkins. Refer to the screenshot below.

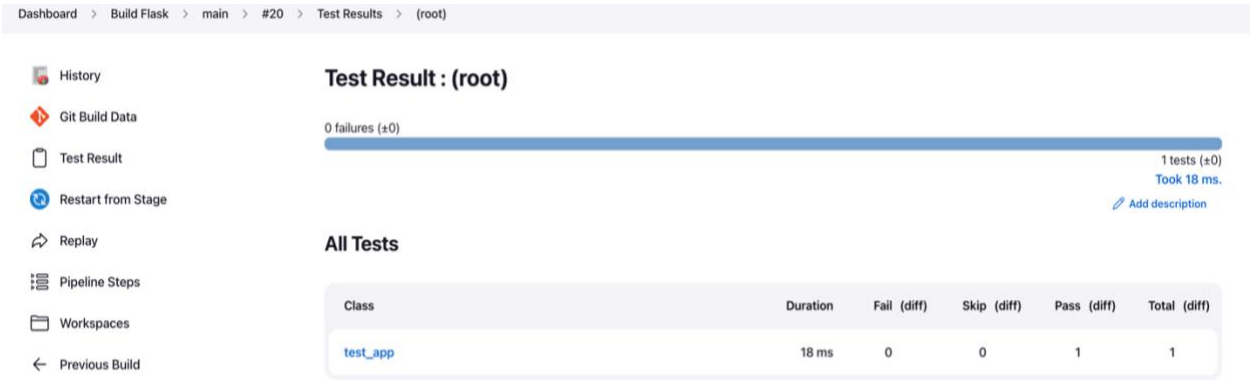


- I was able to see the Flask app in Elastic Beanstalk. Refer to the screenshot below.



- The last thing I did was ran a very simple test to see if my application would break. I inserted a print statement in my test_app.py file and ran the pipeline again in Jenkins. The flask environment was still OK which is good! That means the test was successful. It didn't break my Jenkins build so nothing failed.





- Lastly, I made sure to terminate the Elastic Beanstalk environment and stop my EC2 that Jenkins was running on.