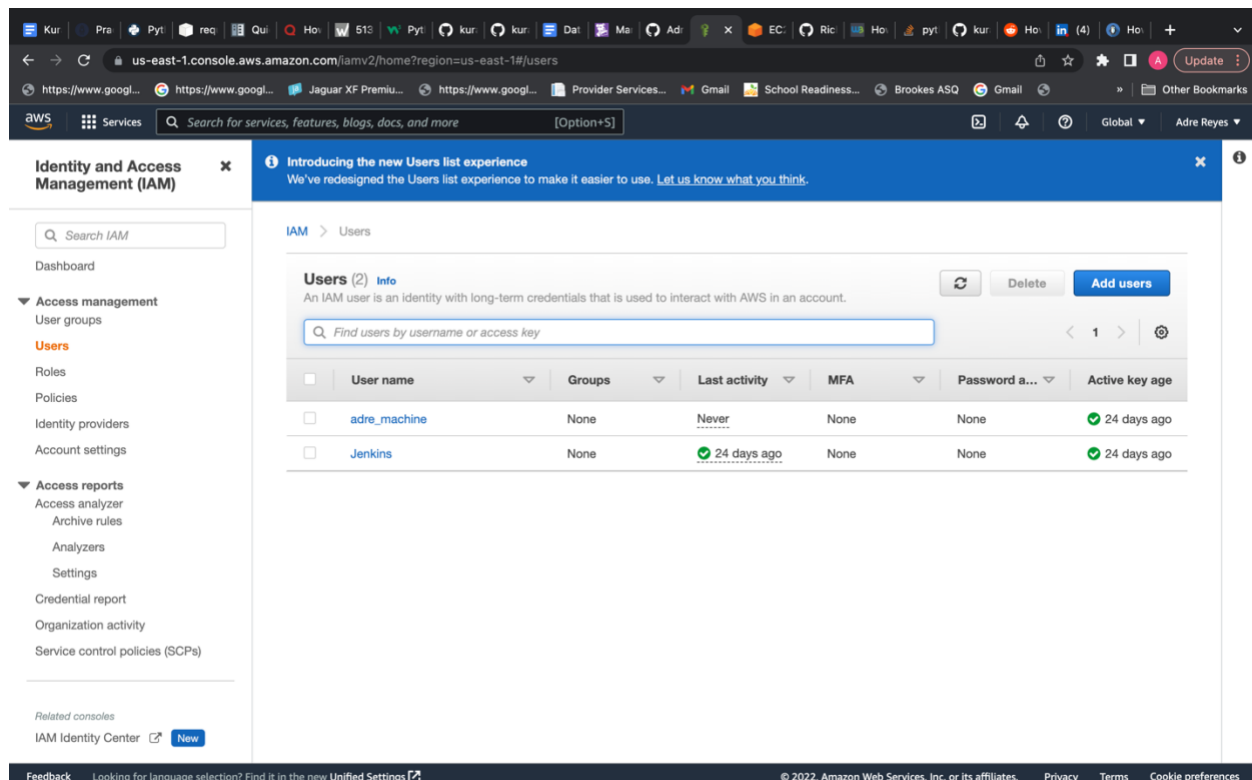


Documentation

- 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)

Add user

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[Add another user](#)

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Select AWS credential type* ☒ **Access key - Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☐ **Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

* Required

[Cancel](#) [Next: Permissions](#)

Feedback Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

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

Add user

Set permissions

[Add user to group](#) [Copy permissions from existing user](#) [Attach existing policies directly](#)

[Create policy](#)

Filter policies Showing 768 results

	Policy name	Type	Used as
<input type="checkbox"/>	AdministratorAccess	Job function	Permissions policy (5)
<input type="checkbox"/>	AdministratorAccess-Amplify	AWS managed	None
<input type="checkbox"/>	AdministratorAccess-AWSElasticBeanstalk	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessFullAccess	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessLifesizeDelegatedAccessPolicy	AWS managed	None
<input type="checkbox"/>	AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None

[Cancel](#) [Previous](#) [Next: Tags](#)

- 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

The screenshot shows the Jenkins 'Enter an item name' dialog box. At the top, there's a search bar with the text 'url-shortener' and a 'Required field' label. Below this, there are several project type options, each with an icon and a description:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

Below these options, there's a section titled 'If you want to create a new item from other existing, you can use this option:' with a 'Copy from' option and a text input field containing 'url-shortener'. At the bottom, there's an 'OK' button.

The screenshot shows the Jenkins 'Branch Sources' configuration page. The page has a tabbed interface with 'Branch Sources' selected. The 'Disable' checkbox is unchecked, with a note: '(No new builds within this Multibranch Pipeline will be executed until it is re-enabled)'. The 'Branch Sources' section has an 'Add source' button and a 'Filter' input field. Below the filter, there's a list of sources: 'Git', 'GitHub', and 'Single repository & branch'. The 'Mode' dropdown is set to 'by Jenkinsfile'. The 'Script Path' input field contains 'Jenkinsfile'. The 'Scan Multibranch Pipeline Triggers' section has a 'Periodically if not otherwise run' checkbox, which is unchecked. The 'Orphaned Item Strategy' section has an 'Abort builds' checkbox, which is unchecked. At the bottom, there are 'Save' and 'Apply' buttons.

General **Branch Sources** Build Configuration Scan Multibranch Pipeline Triggers Orphaned Item Strategy Appearance Health metrics Properties Pipeline Libraries

Disable
☐ (No new builds within this Multibranch Pipeline will be executed until it is re-enabled)

Branch Sources

GitHub
 Credentials ?

- none -

+ Add

url-shortener
 Jenkins
 Jenkins Credentials Provider

Repository

Validate

☐ Repository Scan - Depreciated Visualization

Behaviors

Discover branches
 ?

Strategy ?

Exclude branches that are also filed as PRs

Save Apply

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

General **Branch Sources** Build Configuration Scan Multibranch Pipeline Triggers Orphaned Item Strategy Appearance Health metrics Properties Pipeline Libraries

Disable
☐ (No new builds within this Multibranch Pipeline will be executed until it is re-enabled)

Branch Sources

GitHub
 Credentials ?

kura-labs01/***** (GitHub Token)

+ Add

☒ Repository HTTPS URL

Repository HTTPS URL ?

https://github.com/kura-labs-org/kuralabs_deployment_1.git

Credentials ok. Connected to https://github.com/kura-labs-org/kuralabs_deployment_1.

Validate

☐ Repository Scan - Depreciated Visualization

Behaviors

Discover branches
 ?

Strategy ?

Exclude branches that are also filed as PRs

Save Apply

- I created my build which was successful.

Dashboard > Build Flask > main >

Full Stage View

GitHub

Pipeline Syntax

Build History trend

Filter builds...

#1 Sep 29, 2022, 3:02 AM

Atom feed for all Atom feed for failures

Stage View

Average stage times:
(Average full run time: ~21s)

	Declarative: Checkout SCM	Build	test
Average stage times:	2s	10s	1s
Actual stage times:	2s	10s	1s

Latest Test Result (no failures)

Permalinks

- Last build (#1), 1 day 0 hr ago
- Last stable build (#1), 1 day 0 hr ago
- Last successful build (#1), 1 day 0 hr ago
- Last completed build (#1), 1 day 0 hr ago

REST API Jenkins 2.361.1

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

```

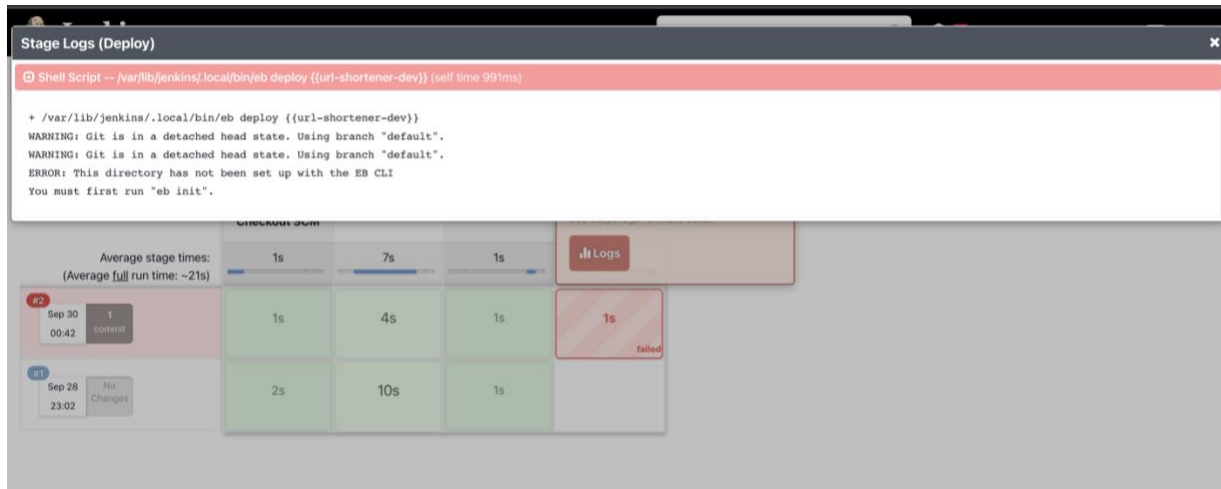
4     stage ('Build') {
5       steps {
6         sh '''#!/bin/bash
7         python3 -m venv test3
8         source test3/bin/activate
9         pip install pip --upgrade
10        pip install -r requirements.txt
11        export FLASK_APP=application
12        flask run &
13        '''
14      }
15    }
16    stage ('test') {
17      steps {
18        sh '''#!/bin/bash
19        source test3/bin/activate
20        py.test --verbose --junit-xml test-reports/results.xml
21        '''
22      }
23    }
24    post{
25      always {
26        junit 'test-reports/results.xml'
27      }
28    }
29  }
30  stage ('Deploy') {
31    steps {
32      sh '/var/lib/jenkins/.local/bin/eb deploy {{url-shortener-dev}}'
33    }
34  }
35  }
36  }
37  }
38

```

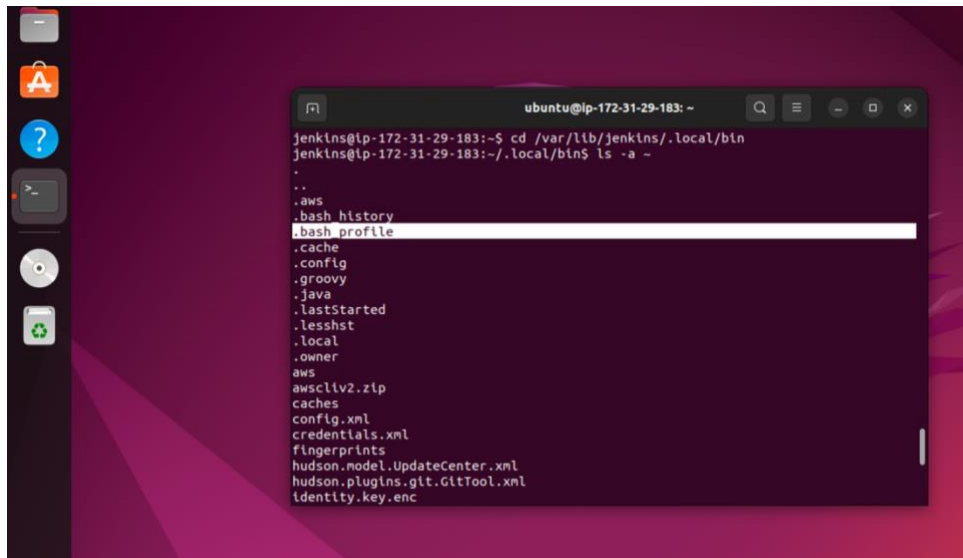
Commit changes

added deploy stage to Jenkinsfile

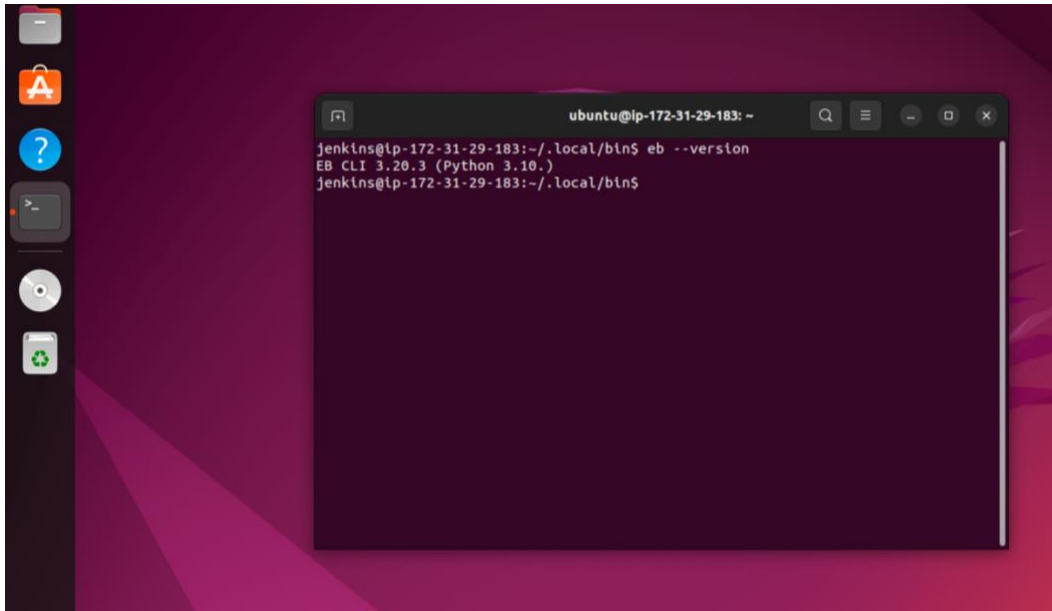
Add an optional extended description...



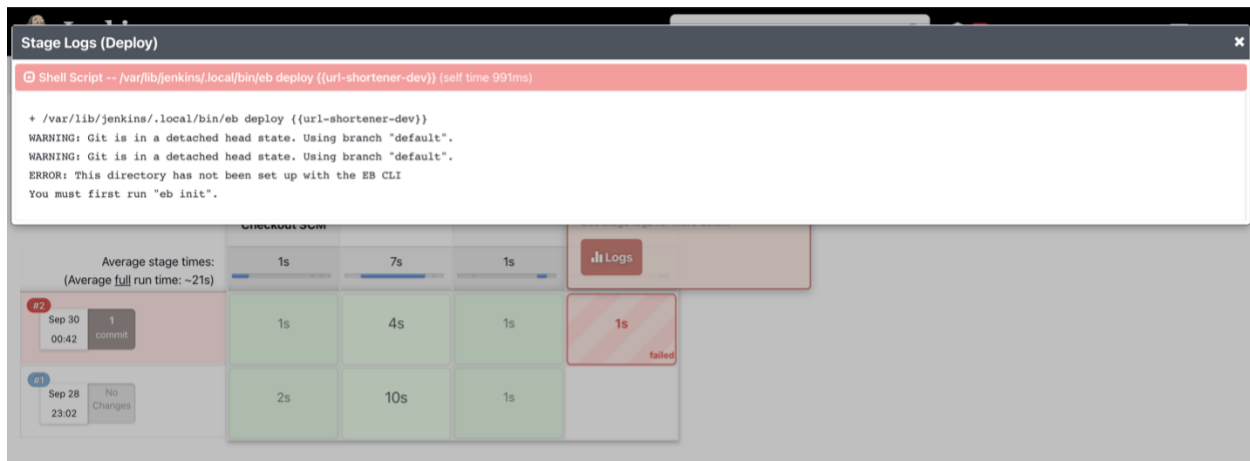
- Why? Because I didn't add the eb executable file to my shell's profile script in the user folder so it could execute Elastic Beanstalk CLI. I actually didn't have a profile script so I made one with <nano .bash_profile> and placed the path in there via the <export PATH=LOCAL_PATH:\$PATH> that I received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


```
ubuntu@ip-172-31-29-183: ~  
jenkins@ip-172-31-29-183:~$ cd workspace/url-shortener/  
jenkins@ip-172-31-29-183:~/workspace/url-shortener$ eb init  
  
Select a default region  
1) us-east-1 : US East (N. Virginia)  
2) us-west-1 : US West (N. California)  
3) us-west-2 : US West (Oregon)  
4) eu-west-1 : EU (Ireland)  
5) eu-central-1 : EU (Frankfurt)  
6) ap-south-1 : Asia Pacific (Mumbai)
```

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

```
ubuntu@ip-172-31-29-183: ~  
Would you like to enable Spot Fleet requests for this environment? (y/N): N  
Creating application version archive "app-220930_043024333184".  
Uploading url-shortener/app-220930_043024333184.zip to S3. This may take a while  
.  
Upload Complete.  
Environment details for: url-shortener-dev  
  Application name: url-shortener  
  Region: us-east-1  
  Deployed Version: app-220930_043024333184  
  Environment ID: e-49apmjxtws  
  Platform: arn:aws:elasticbeanstalk:us-east-1::platform/Python 3.8 running on 64bit Amazon Linux 2/3.3.17  
  Tier: WebServer-Standard-1.0  
  CNAME: url-shortener-dev22.us-east-1.elasticbeanstalk.com  
  Updated: 2022-09-30 04:30:27.680000+00:00  
Printing Status:  
2022-09-30 04:30:26 INFO createEnvironment is starting.  
2022-09-30 04:30:27 INFO Using elasticbeanstalk-us-east-1-445996079350 as Amazon S3 storage bucket for environment data.  
2022-09-30 04:30:49 INFO Created target group named: arn:aws:elasticloadbalancing:us-east-1:445996079350:targetgroup/awseb-AWSEB-24E00NUBY3AL/9c6091352d12942d
```

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

Stage Logs (Deploy)

```

Shell Script -- /var/lib/jenkins/local/bin/job deploy url-shortener-dev (self time 36s)

WARNING: Git is in a detached head state. Using branch "default".
WARNING: Git is in a detached head state. Using branch "default".
WARNING: Git is in a detached head state. Using branch "default".
Alert: The platform version that your environment is using isn't recommended. There's a recommended version in the same platform branch.

Creating application version archive "app-lbb3-221008_133057606742".

Uploading: [-----] 0%
Uploading: [#####] 25%
Uploading: [#####] 50%
Uploading: [#####] 75%
Uploading: [#####] 100% Done...

2022-10-08 13:31:01 INFO Environment update is starting.
2022-10-08 13:31:05 INFO Deploying new version to instance(s).
2022-10-08 13:31:13 INFO Instance deployment successfully generated a 'Procfile'.
2022-10-08 13:31:17 INFO Instance deployment completed successfully.
2022-10-08 13:31:24 INFO New application version was deployed to running EC2 instances.
2022-10-08 13:31:24 INFO Environment update completed successfully.

```

	Declarative: Checkout SCM	Build	test	Deploy
Average stage times: (Average full run time: ~31s)	788ms	4s	948ms	9s
#18 Oct 8, 2022, 1:30 PM				
#17 Oct 8, 2022, 1:29 PM				
#16 Oct 8, 2022, 3:19 AM	756ms	3s	706ms	37s
#15 Oct 8, 2022, 3:15 AM	965ms	4s	1s	59ms
#14 Oct 8, 2022, 3:15 AM				

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

Elastic Beanstalk

Environments > url-shortener-dev

url-shortener-dev
url-shortener-dev22.us-east-1.elasticbeanstalk.com (e-qspd9wdp5b)
Application name: url-shortener

Health
Ok
Causes

Running version
app-86e2-221001_235502547024
Upload and deploy

Platform
Python 3.8 running on 64bit Amazon Linux 2/3.3.17
Change

Recent events
Show all

Time	Type	Details
2022-10-01 19:56:54 UTC-0400	INFO	Environment health has transitioned from Severe to Ok. Application update completed 40 seconds ago and took 24 seconds.
2022-10-01 19:55:33 UTC-0400	INFO	Environment update completed successfully.
2022-10-01 19:55:33 UTC-0400		

- 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.

main Jenkins_deployment2 / test_app.py / <> Jump to

AdreReyes updated print Latest commit

2 contributors

12 lines (9 sloc) 261 Bytes

```
1 from application import app
2
3 # def test_quick():
4 #     a = "jeff"
5 #     greeting = greet(a)
6 #     assert greeting == "Hi jeff"
7
8 def test_home_page():
9     response = app.test_client().get('/')
10    assert response.status_code == 200
11    print("Done.")
12
```

Dashboard > Build Flask > main > #20 > Test Results > (root)

History

Git Build Data

Test Result

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

Test Result : (root)

0 failures (±0)

1 tests (±0)
Took 18 ms.

[Add description](#)

All Tests

Class	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
test_app	18 ms	0	0	1	1

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