

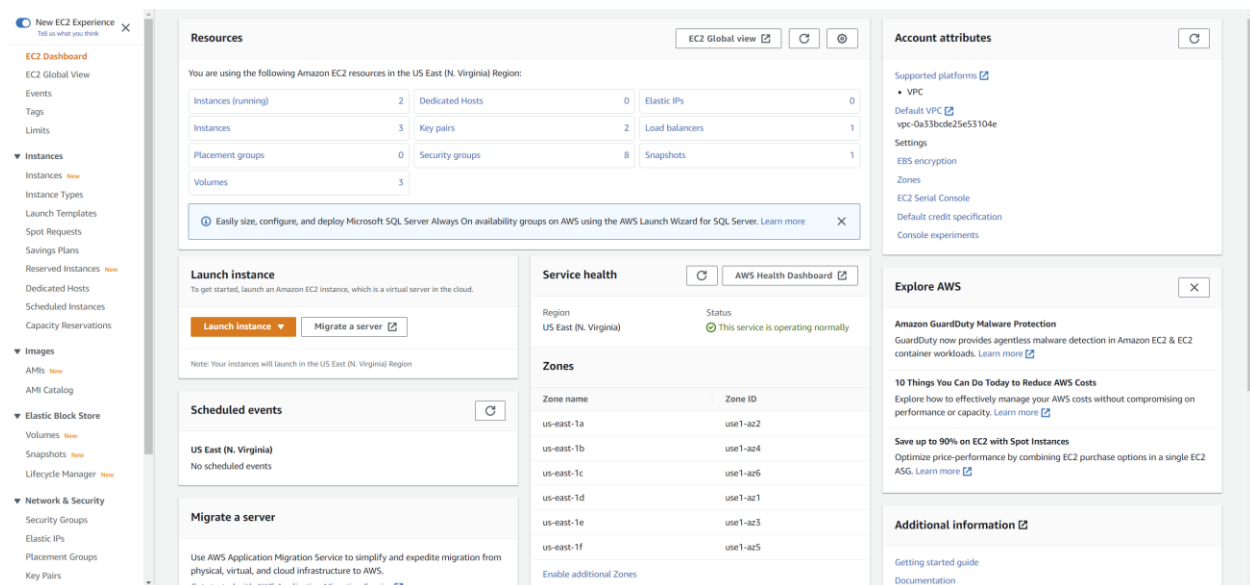
# Antonio Rios

## Deployment 1

This deployment is meant to show the steps necessary to set up a pipeline using Jenkins and elastic beanstalk.

**This is the process when using Jenkins.**

1. To get started it is necessary to create an EC2 using an AWS account.
2. To get to the EC2 menu or dashboard, you can use the search bar or look for the services available.
3. To create a new EC2 you can look for the “Launch instance” option.



Once you select the launch instance you will be prompted with a new menu with multiple options for the EC2.

**For the EC2, the options I selected are the following:**

- Name and tags: Jenkinstest
- Application and OS images: Ubuntu Server 22.04 LTS
  - 64-bit (x86)
  - Free tier
- Instance type: t2.micro
- Key pair: Awskey
  - I selected the RSA key type because I am planning to use my virtual machine to connect to my EC2 using ssh. This key pair will generate a file that I can use to

ssh into the EC2. The file generated is going to be called awskey.pem. The extension .pem is used for rsa key pairs.

- Network Settings:
  - VPC: default
  - Subnet: No preference
  - Auto-assign public IP: Enable
  - Firewall
    - Create security groups has to be selected and enable access to the ports 22, 80 and 8080
    - We use 22 for ssh, 80 for http, and 8080 to access the web interface for Jenkins in the browser.

**Inbound security groups rules**

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) Remove

|                                  |  |   |
|----------------------------------|--|---|
| Type <a href="#">Info</a>        | Protocol <a href="#">Info</a>  | Port range <a href="#">Info</a>                         |
| ssh ▼                            | TCP  | 22  |
| Source type <a href="#">Info</a> | Source <a href="#">Info</a>  | Description - optional <a href="#">Info</a>             |
| Anywhere ▼                       | <input type="text" value="Add CIDR, prefix list or security group"/> | <input type="text" value="e.g. SSH for admin desktop"/> |
|                                  | 0.0.0.0/0 ✕  |   |

▼ Security group rule 2 (TCP, 8080) Remove

|                                  |  |   |
|----------------------------------|--|---|
| Type <a href="#">Info</a>        | Protocol <a href="#">Info</a>  | Port range <a href="#">Info</a>                         |
| Custom TCP ▼                     | TCP  | 8080  |
| Source type <a href="#">Info</a> | Source <a href="#">Info</a>  | Description - optional <a href="#">Info</a>             |
| Custom ▼                         | <input type="text" value="Add CIDR, prefix list or security group"/> | <input type="text" value="e.g. SSH for admin desktop"/> |

▶ Security group rule 3 (TCP, 80) Remove

▼ Security group rule 4 (TCP, 80) Remove

|                                  |  |   |
|----------------------------------|--|---|
| Type <a href="#">Info</a>        | Protocol <a href="#">Info</a>  | Port range <a href="#">Info</a>                         |
| HTTP ▼                           | TCP  | 80  |
| Source type <a href="#">Info</a> | Source <a href="#">Info</a>  | Description - optional <a href="#">Info</a>             |
| Custom ▼                         | <input type="text" value="Add CIDR, prefix list or security group"/> | <input type="text" value="e.g. SSH for admin desktop"/> |

- Configure Storage: 8 GiB gp2(general purpose SSD)

Those are all the details necessary when launching the new EC2.

It will take a few seconds until the EC2 is running. Once it is ready. I accessed the EC2 using the pem file I created earlier. After the key pair was created a file was automatically downloaded to my computer. I transferred this file to my virtual machine running ubuntu on VMware.

Once I am located on the same directory as the key. I can run a ssh command to connect to the newly created EC2.

```
ssh -i awskey.pem ubuntu@44.203.85.192
```

*Something I thought it was interesting about connecting to the EC2 is that the Public IP always changes, I was working on the project on multiple days and I was not able to connect to my EC2 using the same IP. I had to go back to AWS and check the current Public IP for my computer.*

Once the connection is successful, we can run commands remotely to the EC2. To set up the EC2 I ran the following commands:

```
$sudo apt update && sudo apt install default-jre
```

This command will make sure to update all the libraries of the system and download Java on the computer. This is essential because Jenkins is a java application and would have complication if it does not have java.

```
$wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo gpg --dearmor -o /usr/share/keyrings/jenkins.gpg
```

This command will import a key file for Jenkins to enable the installation.

```
$sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg] http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

This will add the Jenkins repository to the sources list and also the key for authentication.

```
$sudo apt update && sudo apt install jenkins -y
```

Now the system will update the libraries and install Jenkins at the same time.

```
$sudo systemctl start jenkins
```

It's time to run Jenkins now. This command will start the Jenkins web interface.

To make sure that Jenkins is running properly you can use the following command.

```
$sudo systemctl status Jenkins
```

```
Last login: Thu Sep  1 23:46:57 2022 from 18.206.107.27
ubuntu@ip-172-31-91-107:~$ sudo systemctl status jenkins
• jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-09-01 23:46:44 UTC; 3h 50min ago
     Main PID: 455 (java)
       Tasks: 40 (limit: 1143)
      Memory: 540.1M
         CPU: 2min 14.090s
    CGroup: /system.slice/jenkins.service
            └─455 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
```

The outcome of the last command should look like this. It is really easy to see that Jenkins is running and it's colored green to make it more accessible.

After Jenkins is running, we still need to make sure that it's able to run a python virtual environment. We use these two commands to install python and the virtual environment.

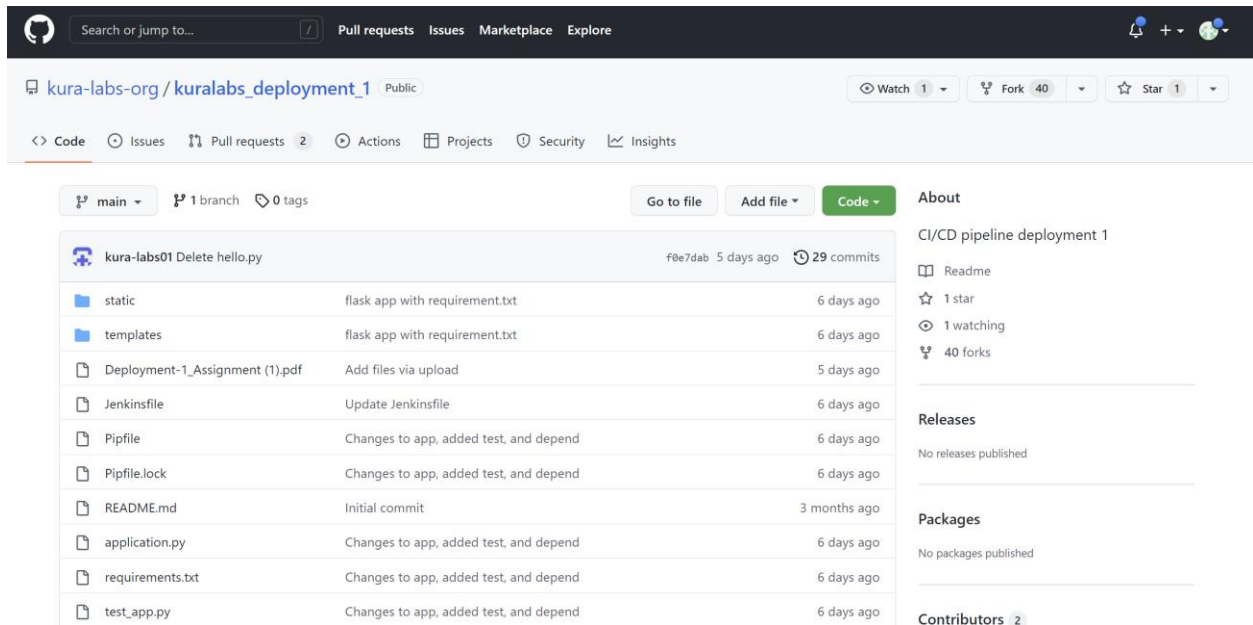
```
$sudo apt install python3-pip
$sudo apt-get install -y python3-venv
```

*Troubleshooting: When I was working on setting up the environment for the first time I tried. I ran into some issues that would make it not possible to see Jenkins. I saw that it was running but I was not able to access it.*

*To promptly figured that I didn't open the ports when I was creating the EC2 and I did try to open the ports using the command line but was still unsuccessful.*

*What ended up fixing my issue was changing the security groups from the AWS console, I enable access from everywhere to my EC2 using the ports 80 for http, 8080 for Jenkins. I didn't include 22 for ssh because that was working properly otherwise, I would not have been able to connect to the computer in the first place.*

To continue with the deployment the next step is to get the source code from the github repository at [https://github.com/kura-labs-org/kuralabs\\_deployment\\_1](https://github.com/kura-labs-org/kuralabs_deployment_1)



Once you are on the page, select the option “Fork” to create a copy of the repository on your own github account in a new repository.

To be able to connect Jenkins with the repository it has to have permissions or access to the git hub account. In this case we use an access token.

To create the access token go to the github profile and select settings, scroll all the way down to developer settings and then personal access tokens. If you had the app set up you will be prompted to type the same code on your phone to allow access.



Confirm access



### GitHub Mobile

We sent you a verification request on your GitHub Mobile app. Enter the digits shown below to enter sudo mode.

26

Having problems?

- [Use your password](#)

Tip: You are entering [sudo mode](#). After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.

GitHub Apps

OAuth Apps

Personal access tokens

## Personal access tokens

Generate new token

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

jenkins — admin:repo\_hook, repo

Never used

Delete

Expires on Sat, Oct 1 2022.

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

The important things to authorize access to are “repo” and “admin:repo\_hook”. The expiration can be left at the default 30 days.

## Expiration \*

30 days

The token will expire on Sun, Oct 2 2022

## Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

|   |                                      |
|---|--------------------------------------|
| <input checked="" type="checkbox"/> repo            | Full control of private repositories |
| <input checked="" type="checkbox"/> repo:status     | Access commit status                 |
| <input checked="" type="checkbox"/> repo_deployment | Access deployment status             |
| <input checked="" type="checkbox"/> public_repo     | Access public repositories           |
| <input checked="" type="checkbox"/> repo:invite     | Access repository invitations        |
| <input checked="" type="checkbox"/> security_events | Read and write security events       |

|   |                                  |
|---|----------------------------------|
| <input checked="" type="checkbox"/> admin:repo_hook | Full control of repository hooks |
| <input checked="" type="checkbox"/> write:repo_hook | Write repository hooks           |
| <input checked="" type="checkbox"/> read:repo_hook  | Read repository hooks            |

Once both options are selected you can continue to generate the token. This token will be used later.

Now it's time to go to the Jenkins website. I used <http://44.203.85.192:8080/> to access it.

To start building the pipeline you can select the new item option from the left menu.

- + New Item
- 👤 People
- 📁 Build History
- 😊 Project Relationship
- 🔍 Check File Fingerprint
- ⚙️ Manage Jenkins
- 👤 My Views
- 📁 New View

Assign a name and select the multibranch pipeline option and click OK.



### Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

For the general option you can assign the name and brief description.

Display Name ?

jenkins-pipeline-1

Description

CI/CD pipeline deployment 1

[Plain text] [Preview](#)

On the branch sources select github and then select Jenkins.

General

Branch Sources

Build Configuration

Scan Multibranch Pipeline Triggers

Orphaned Item Strategy

Appearance

Health metrics

Properties

Pipeline Libraries

### Branch Sources

Add source ^

Filter

Git

GitHub

Single repository & branch

Mode

### Branch Sources

GitHub

Credentials ?

- none -

+ Add

test

Jenkins

recommended

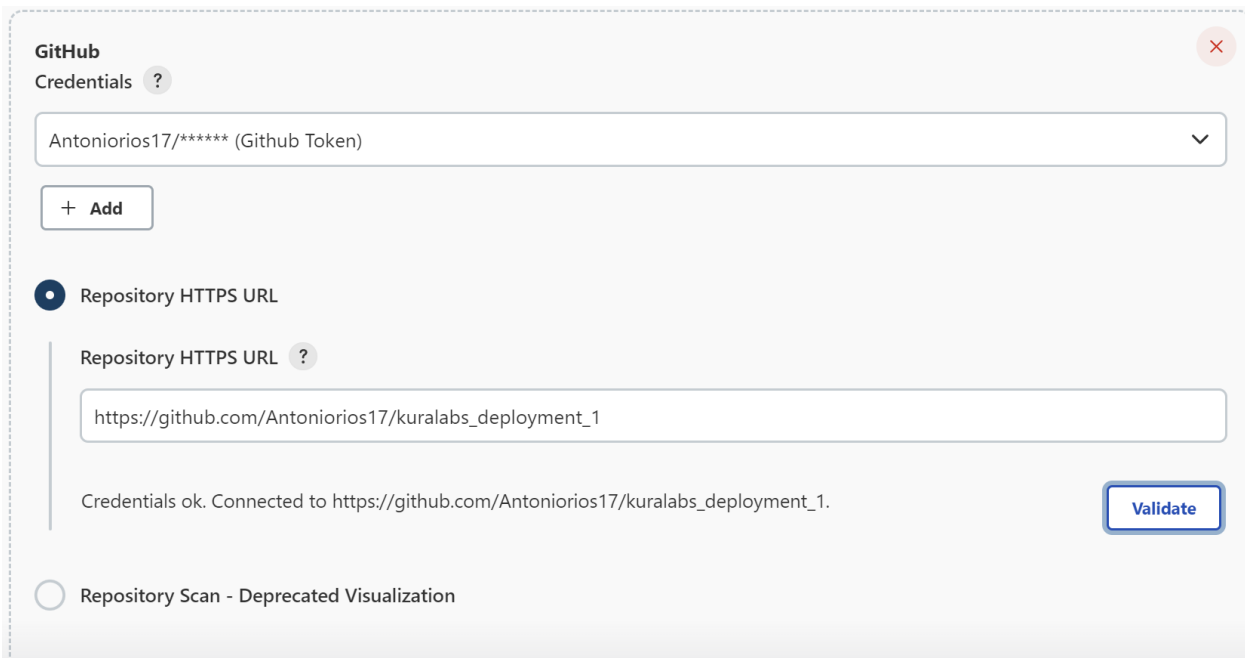
Repository HTTPS URL ?

Validate

After you select Jenkins and new menu will prompt where you can enter your username for github and password which is the token that was created earlier.

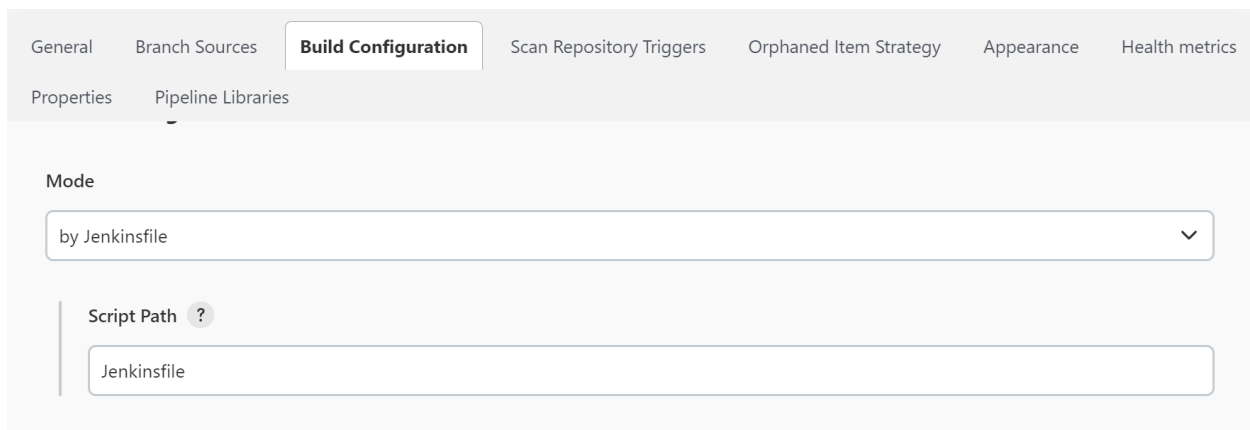


After adding the username and token you can add the link of the repository to be used. The button “validate” can be used to make sure the access to the repository is working properly.



The screenshot shows the 'GitHub Credentials' configuration page in Jenkins. At the top, there's a 'GitHub Credentials' section with a dropdown menu showing 'Antoniorios17/\*\*\*\*\* (Github Token)' and a '+ Add' button. Below this is the 'Repository HTTPS URL' section, which has a text input field containing 'https://github.com/Antoniorios17/kuralabs\_deployment\_1'. A status message below the input says 'Credentials ok. Connected to https://github.com/Antoniorios17/kuralabs\_deployment\_1.' and there is a 'Validate' button. At the bottom, there is an option for 'Repository Scan - Deprecated Visualization' which is currently unselected.

The build configurations needs to state that this is a jenkinsfile:



The screenshot shows the 'Build Configuration' tab in the Jenkins configuration interface. The 'Mode' dropdown menu is set to 'by Jenkinsfile'. Below it, the 'Script Path' text input field contains 'Jenkinsfile'. The top navigation bar includes tabs for 'General', 'Branch Sources', 'Build Configuration' (which is active), 'Scan Repository Triggers', 'Orphaned Item Strategy', 'Appearance', and 'Health metrics'. Below the main tabs, there are links for 'Properties' and 'Pipeline Libraries'.

To complete the build “apply” and “save” the configuration. After saving it will start building.

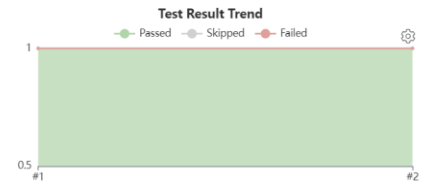
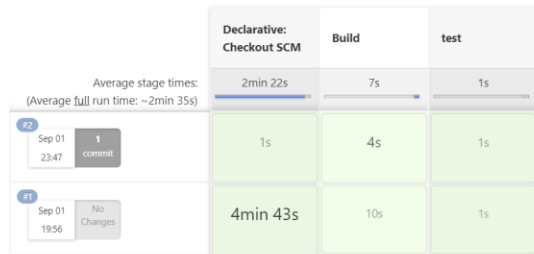
After the building process is complete it will look like this:

## Branch main

Full project name: jenkins-pipeline/main

 [Recent Changes](#)

### Stage View



The number #1 was the first time it was created and the number #2 is when I made a commit to update the application. It is interesting that it took a way shorter time the second time. It makes sense since the configuration was already done before hand.



## Build History

trend ▼

🔍 Filter builds...



#2

Sep 2, 2022, 3:47 AM



#1

Sep 1, 2022, 11:56 PM



Atom feed for all



Atom feed for failures

## Launching the application using Elastic beanstalk

First access the repository for the application at [https://github.com/kura-labs-org/kuralabs\\_deployment\\_1](https://github.com/kura-labs-org/kuralabs_deployment_1) , then use the green “code” button to clone the repository into the local computer.

I used the “download zip” option to get all the files into my local computer.

The screenshot displays a GitHub repository interface. At the top, there are buttons for 'Go to file', 'Add file', and 'Code'. Below these, a table lists repository files and folders. A 'Clone' dropdown menu is open, showing options for cloning the repository using HTTPS, SSH, or GitHub CLI. The 'Download ZIP' option is highlighted. Below the file list, the 'README.md' content is visible, showing the title 'kuralabs\_deployment\_1' and the subtitle 'CI/CD pipeline deployment 1'.

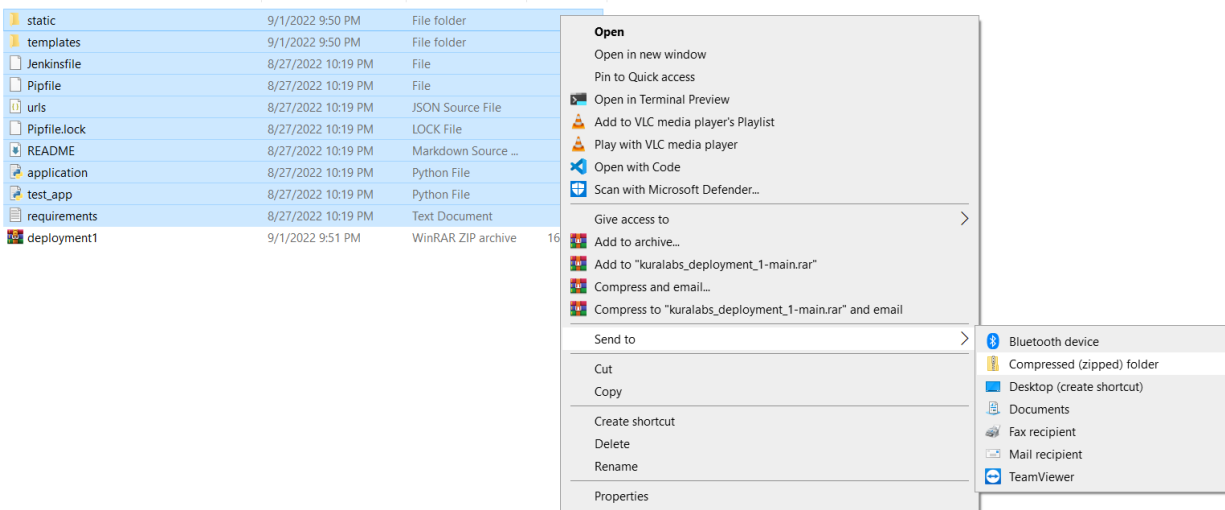
| File/Folder                     | Description                            | Commit Date  |
|---------------------------------|--|--------------|
| static                          | flask app with requirement.txt         |              |
| templates                       | flask app with requirement.txt         |              |
| Deployment-1_Assignment (1).pdf | Add files via upload                   |              |
| Jenkinsfile                     | Update Jenkinsfile                     |              |
| Pipfile                         | Changes to app, added test, and d      |              |
| Pipfile.lock                    | Changes to app, added test, and d      |              |
| README.md                       | Initial commit                         | 3 months ago |
| application.py                  | Changes to app, added test, and depend | 6 days ago   |
| requirements.txt                | Changes to app, added test, and depend | 6 days ago   |
| test_app.py                     | Changes to app, added test, and depend | 6 days ago   |
| urls.json                       | flask app with requirement.txt         | 6 days ago   |

README.md

### kuralabs\_deployment\_1

CI/CD pipeline deployment 1

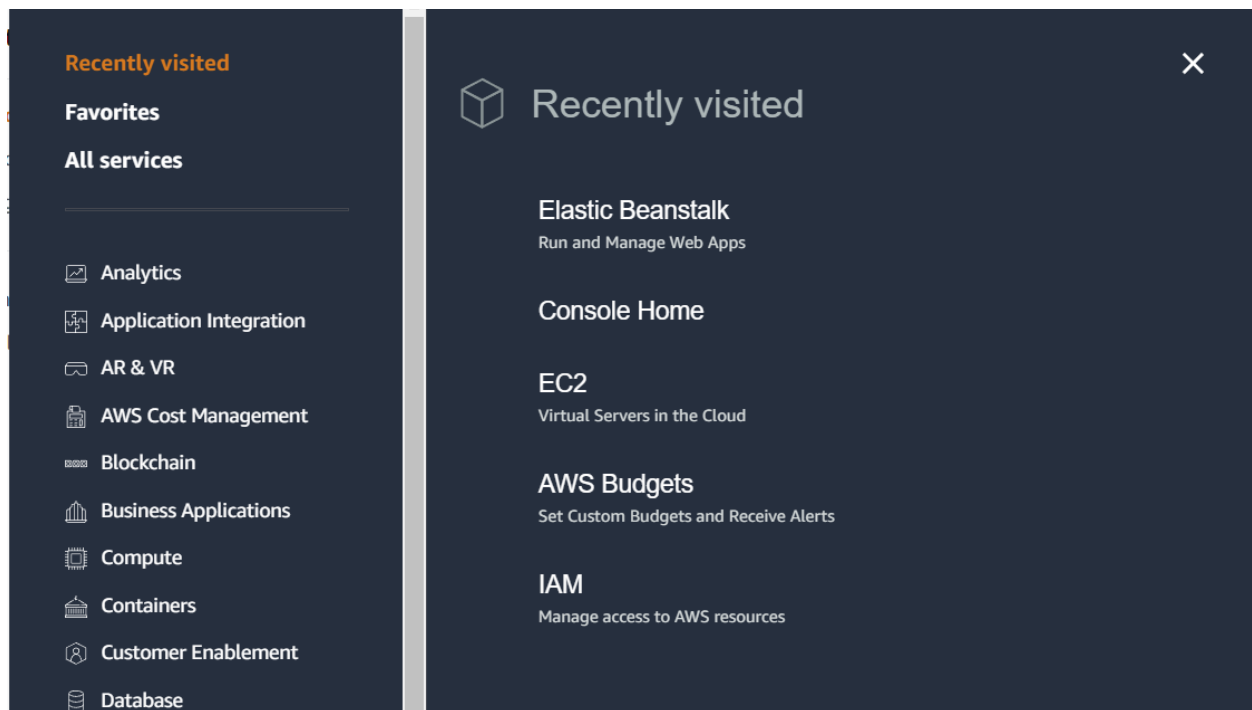
*This part was a little tricky because when I zipped the files directly from github and tried to use them to run elastic beanstalk it did not work out and gave me a state of severe on the health. To resolve this issue, I had to zip only the files inside into a new compressed file and use those instead.*



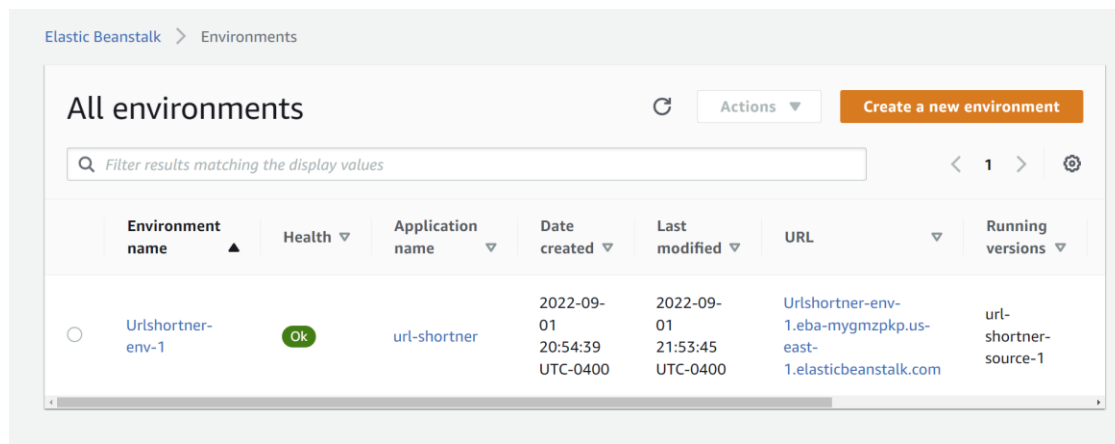
The new file that was created that I called “deployment1” is the one I used to run it with elastic beanstalk where I did not get any of the issues previously discussed.

To start working with elastic beanstalk you can head to the AWS website and sign in.

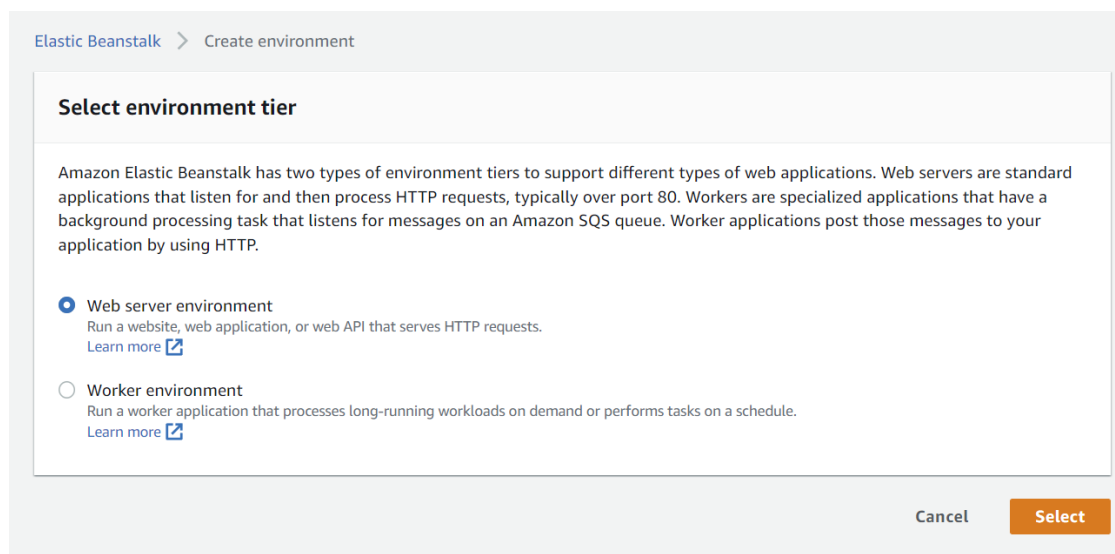
Once you have signed in you can look for the services section on the top left corner or type the name directly in the search box. “Elastic Beanstalk”.



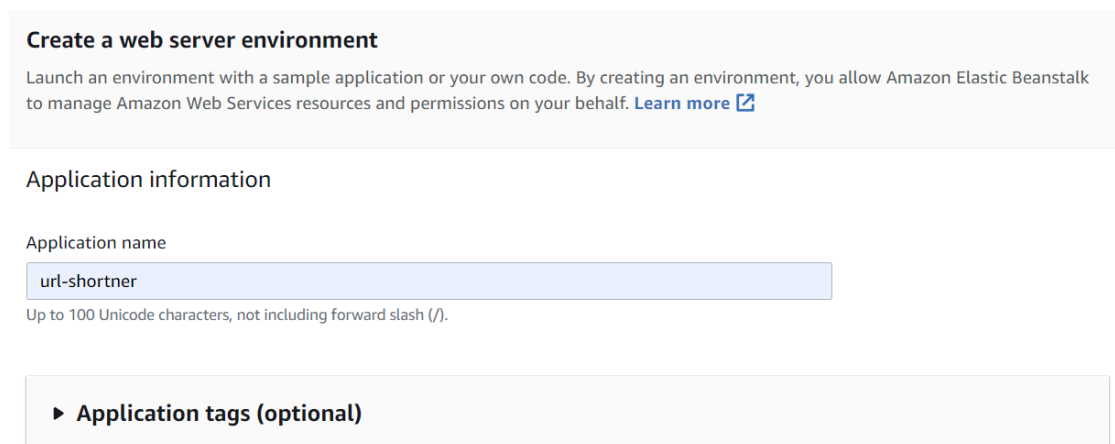
Once you are in the Elastic Beanstalk page select the option “create new environment”.



Followed by choosing to create a “Web Server Environment”



Add the application name “url-shortner”



Select python as the preferred platform to work with:

### Platform

☒ **Managed platform**  
Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#)

☐ **Custom platform**  
Platforms created and owned by you.

**Platform**  
Python

**Platform branch**  
Python 3.8 running on 64bit Amazon Linux 2

**Platform version**  
3.3.17 (Recommended)

To be able to use the repository we clone we used the “upload your code” option and then I selected the zipped repository using the choose file option.

### Application code

☐ **Sample application**  
Get started right away with sample code.

☐ **Existing version**  
Application versions that you have uploaded for `url-shortner`.  
-- Choose a version --

☒ **Upload your code**  
Upload a source bundle from your computer or copy one from Amazon S3.  
**Version label**  
Unique name for this version of your application code.  
url-shortner-source  
**Source code origin**  
Maximum size 512 MB  
☒ **Local file**  
☐ **Public S3 URL**  
Choose file  
☐ No file uploaded  
▶ Application code tags

Cancel

Configure more options

Create environment

After all the steps are done you can select “Create environment”.

It will take a few minutes for the whole building process to finish. The environment should have a health “Ok” distinction so know that everything is running properly.

**Elastic Beanstalk**

Environments  
Applications  
Change history

▼ url-shortner  
Application versions  
Saved configurations

▼ **Urlshortner-env-1**  
Go to environment [🔗](#)  
Configuration  
Logs  
Health  
Monitoring  
Alarms  
Managed updates  
Events  
Tags

▼ Recent environments  
**Urlshortner-env-1**

**Urlshortner-env-1**  
Urlshortner-env-1.elba-mygmzpkp.us-east-1.elasticbeanstalk.com [🔗](#) (e-hqf6f2ma5a)  
Application name: url-shortner

Refresh Actions

**Health**  
Ok  
Causes

**Running version**  
url-shortner-source-1  
Upload and deploy

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

**Recent events** Show all

| Time                         | Type | Details   |
|------------------------------|------|---|
| 2022-09-01 21:55:08 UTC-0400 | INFO | Environment health has transitioned from Severe to Ok. Application update completed 83 seconds ago and took 28 seconds. |
| 2022-09-01 21:53:45 UTC-0400 | INFO | Environment update completed successfully.  |
| 2022-09-01 21:53:45 UTC-0400 | INFO | New application version was deployed to running EC2 instances.  |
| 2022-09-01 21:53:41 UTC-0400 | INFO | Instance deployment completed successfully.   |
| 2022-09-01 21:53:38 UTC-0400 | INFO | Instance deployment successfully generated a 'Profile'.   |

To see the final product you can click on the blue text on top of the check mark and it will direct you to the web page of the application.

URL Shortener

API

New URL

Website

Short Name

Website URL

Shorten

File

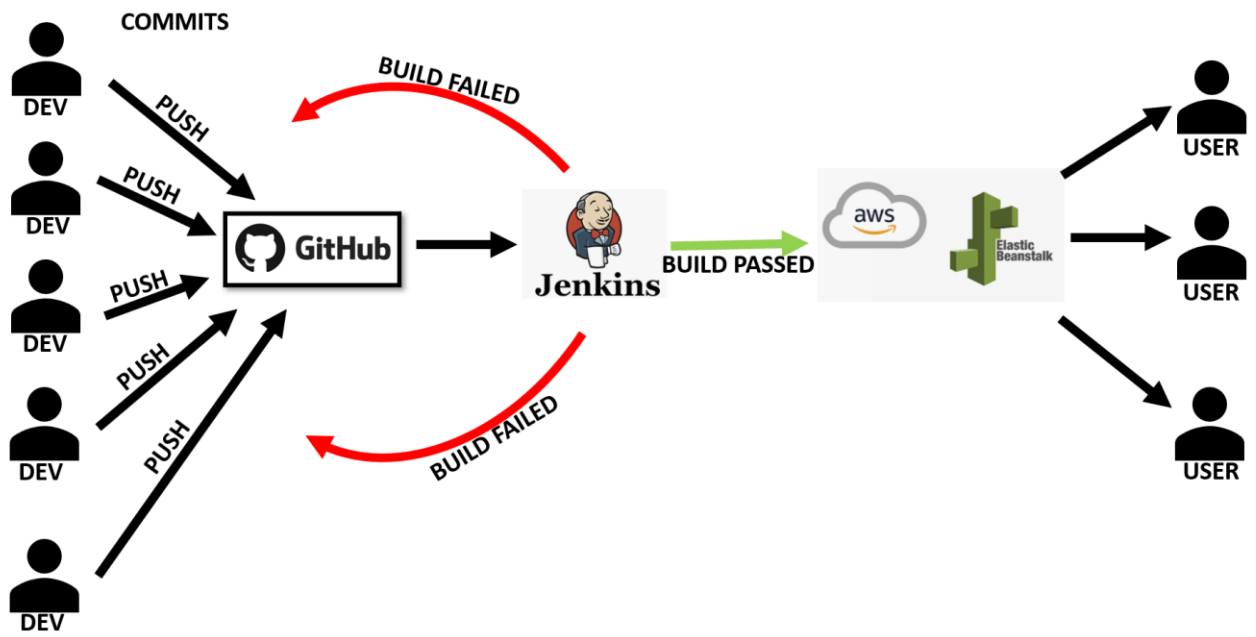
Short Name

Website URL

Choose File No file chosen

Shorten

## Diagram:





**Improvements:**

The overall project was not very smooth, I got multiple road blocks working on the deployment. Doing the process manually definitely taught me about the essential configuration of the application.

One way to improve the process of setting up a new EC2 is to have a script ready with all the command we used to set up the environment.

If I can ssh into the computer, I can also ssh and copy files directly to the EC2 to automate the process of running all the commands I needed to run.