

# Setting up the EC2 Server

When starting I had an EC2 already set up with the AWS Ubuntu 22.04 jammy image.

```
richard@Tokyo:~/jenkins $ screenfetch
      .+/+o+-
      yyyyy- -yyyyyy+
      ://+////////-yyyyyyo
      .++ .:/++++++/-+.sss/`
      .:++o: /+++++++/:--:/-
      o:+o+:++. `..``.-/oo+++++/
      .:~o:~o/. `+sssoo+/
      .++/+~:oo+o: ` /sssooo.
      /+++//+:`oo+o /::--:.
      \+/+o+++`o++o ++////.
      .++.o+++oo+: ` /dddhhh.
      .+.o+oo:. ` oddhhhh+
      \+.++o+o`-`~~~~.:ohdhhhhh+
      `:o+++ `ohhhhhhhhyo++os:
      .o: ` .syhhhhhhh/.oo++o`
      /osyyyyyyo++ooo+++/
      ~~~~~ +oo+++o\
      `oo++.
```

```
richard@Tokyo
OS: Ubuntu 22.04 jammy
Kernel: x86_64 Linux 5.15.0-1017-aws
Uptime: 14d 3h 14m
Packages: 681
Shell: bash 5.1.16
Disk: 3.5G / 7.7G (45%)
CPU: Intel Xeon E5-2676 v3 @ 2.4GHz
RAM: 707MiB / 966MiB
```

```
richard@Tokyo:~/jenkins $
```

Then I edited the port rules of the EC2 making sure 22, 443, 8080 and 22 are open.

Inbound rules <a href="#">Info</a>						
Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>	
sgr-03fdf7d94e86ffd47	SSH ▼	TCP	22	Custom ▼ <input type="text" value="0.0.0.0"/>	SSH Traffic	<a href="#">Delete</a>
sgr-097abfbcb953e7936	HTTPS ▼	TCP	443	Custom ▼ <input type="text" value="0.0.0.0"/>	HTTPS Traffic	<a href="#">Delete</a>
sgr-01d394d4ca6afe2e0	Custom TCP ▼	TCP	8080	Custom ▼ <input type="text" value="0.0.0.0"/>	Jenkins Traffic	<a href="#">Delete</a>
sgr-006ae1bda2c51382e	HTTP ▼	TCP	80	Custom ▼ <input type="text" value="0.0.0.0"/>	HTTP Traffic	<a href="#">Delete</a>

# Installing Jenkins on the EC2

After logging on to the EC2 with SSH I ran the script I made below(also on my github as **runinstalljenkins.sh**) that I transferred over to the EC2 earlier. This script **runs a script to install Jenkins(installjenkins.sh)**:

```
1  #!/bin/bash
2
3  #This script is to run the Install Jenkins script
4
5  #Run as admin only check
6  if [ $UID != 0 ]; then
7      echo "Run again with admin permissions"
8      exit 1
9  fi
10
11  echo "Installing Jenkins"
12
13  #Command to run the install script and log everything
14  /bin/bash installjenkins.sh &> installjenkins.log
15
16  #Check if the install had a error
17  if [ $? -ne 0 ]; then
18      echo "Installation error"
19      exit 1
20  else
21      echo "Installation successful"
22      echo "Installation Logs in 'installjenkins.log'"
23  fi
24
25  #Wait 5 seconds then check status
26  sleep 5
27
28  #Check the status of the service
29  systemctl status jenkins --no-pager
30
31  #Print out secret password to login and setup Jenkins
32  echo "Password to Unlock and Setup Jenkins Below"
33  cat /var/lib/jenkins/secrets/initialAdminPassword
34
35  #successful
36  echo "Run successful"
37  exit 0
```

Below is the script(also on my github as **installjenkins.sh**) that **runinstalljenkins.sh** runs. Also transferred to the EC2 earlier. The **output is stored in a log file**.

```
1  #!/bin/bash
2
3  #This script is to install Jenkins and it's dependencies and start the service
4
5  #Run as admin only check
6  if [ $UID != 0 ]; then
7      echo "Run again with admin permissions"
8      exit 1
9  fi
10
11 #Adding the Keyrings without user interaction
12 wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | gpg --batch --yes --dearmor -o /usr/share/keyrings/jenkins.gpg &&
13 echo "Jenkins Keyring Added"
14
15 #Adding the repo to the sources of apt
16 sh -c 'echo deb [signed-by=/usr/share/keyrings/jenkins.gpg] http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/
17 jenkins.list' && echo "Jenkins Repo Added"
18
19 #Update local repo database
20 apt-get update
21
22 #Install java, Jenkins, pip and venv in that order
23 apt-get install default-jre -y && echo "Installed Java Runtime Engine" && apt-get install jenkins -y && echo "Installed Jenkins" &&
24 apt-get install python3-pip -y && echo "Installed Python pip" && apt-get install python3.10-venv -y && echo "Installed Python venv"
25
26 #Start the Jenkins service
27 systemctl start jenkins && echo "Jenkins Started"
28
29 #successful
30 echo "Installation successful"
31 exit 0
```

Below is an example of the output(censored) from the **runinstalljenkins.sh** script.

```
Ubuntu 20.04 on Windows
richard@Tokyo:~$ sudo ./runinstalljenkins.sh
Installing Jenkins
Installation successful
Installation Logs in 'installjenkins.log'
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-08-31 20:59:31 UTC; 16s ago
     Main PID: 137227 (java)
        Tasks: 43 (limit: 1143)
      Memory: 232.8M
         CPU: 20.536s
       CGroup: /system.slice/jenkins.service
               └─137227 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot

Aug 31 20:59:16 Tokyo jenkins[137227]: *****
Aug 31 20:59:16 Tokyo jenkins[137227]: *****
Aug 31 20:59:16 Tokyo jenkins[137227]: WARNING: An illegal reflective access operation has occurred
Aug 31 20:59:16 Tokyo jenkins[137227]: WARNING: Illegal reflective access by org.codehaus.groovy.vmplugin.v4.SwingUtils$ClassPath$1 (module
Aug 31 20:59:16 Tokyo jenkins[137227]: WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy
Aug 31 20:59:16 Tokyo jenkins[137227]: WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
Aug 31 20:59:16 Tokyo jenkins[137227]: WARNING: All illegal access operations will be denied in a future release
Aug 31 20:59:31 Tokyo jenkins[137227]: 2022-08-31 20:59:31.617+0000 [id=28] INFO jenkins.model.Jenkins$2 - Jenkins is starting up
Aug 31 20:59:31 Tokyo jenkins[137227]: 2022-08-31 20:59:31.657+0000 [id=22] INFO hudson.model.Hudson - Hudson is starting up
Aug 31 20:59:31 Tokyo systemd[1]: Started Jenkins Continuous Integration Server.
Hint: Some lines were ellipsized, use -l to show in full.
Password to Unlock and Setup Jenkins Below

Run successful
richard@Tokyo:~$
```

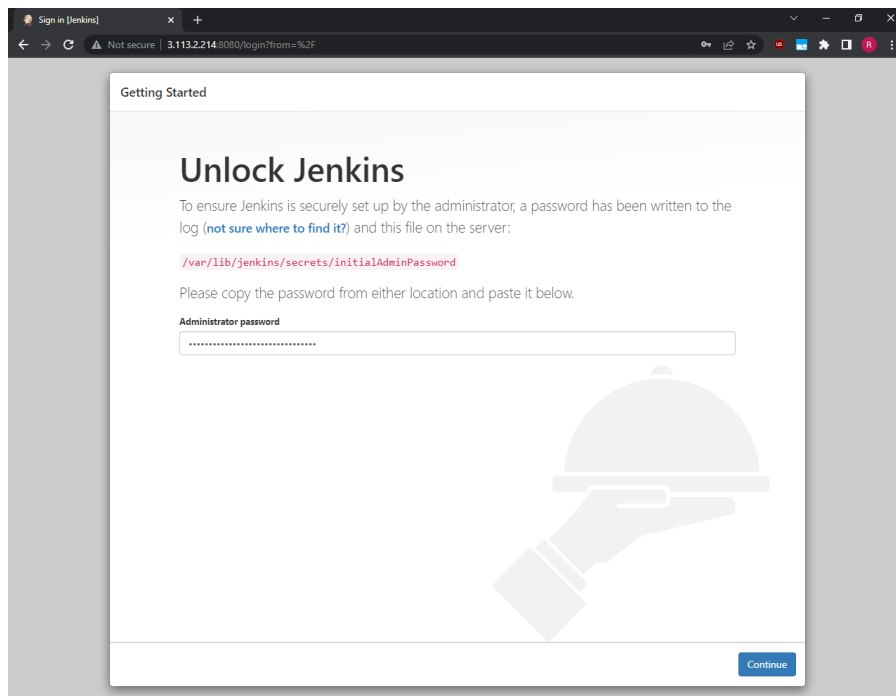
Below is an example of the logs produced from the **installjenkins.sh** script.

```
Ubuntu 20.04 on Windows
GNU nano 6.2
Selecting previously unselected package libpython3.10-dev:amd64.
Preparing to unpack .../06-libpython3.10-dev_3.10.4-3ubuntu0.1_amd64.deb ...
Unpacking libpython3.10-dev:amd64 (3.10.4-3ubuntu0.1) ...
Selecting previously unselected package libpython3-dev:amd64.
Preparing to unpack .../07-libpython3-dev_3.10.4-0ubuntu2_amd64.deb ...
Unpacking libpython3-dev:amd64 (3.10.4-0ubuntu2) ...
Selecting previously unselected package python3.10-dev.
Preparing to unpack .../08-python3.10-dev_3.10.4-3ubuntu0.1_amd64.deb ...
Unpacking python3.10-dev (3.10.4-3ubuntu0.1) ...
Selecting previously unselected package python3-dev.
Preparing to unpack .../09-python3-dev_3.10.4-0ubuntu2_amd64.deb ...
Unpacking python3-dev (3.10.4-0ubuntu2) ...
Selecting previously unselected package python3-wheel.
Preparing to unpack .../10-python3-wheel_0.37.1-2_all.deb ...
Unpacking python3-wheel (0.37.1-2) ...
Selecting previously unselected package python3-pip.
Preparing to unpack .../11-python3-pip_22.0.2+dfsg-1_all.deb ...
Unpacking python3-pip (22.0.2+dfsg-1) ...
Setting up javascript-common (11+nmu1) ...
Setting up python3-wheel (0.37.1-2) ...
Setting up libexpat1-dev:amd64 (2.4.7-1) ...
Setting up python3-pip (22.0.2+dfsg-1) ...
Setting up zlib1g-dev:amd64 (1:1.2.11.dfsg-2ubuntu9) ...
Setting up libjs-jquery (3.6.0+dfsg~3.5.13-1) ...
Setting up libjs-underscore (1.13.2~dfsg-2) ...
Setting up libpython3.10-dev:amd64 (3.10.4-3ubuntu0.1) ...
Setting up libjs-sphinxdoc (4.3.2-1) ...
Setting up python3.10-dev (3.10.4-3ubuntu0.1) ...
Setting up libpython3-dev:amd64 (3.10.4-0ubuntu2) ...
Setting up python3-dev (3.10.4-0ubuntu2) ...
Processing triggers for man-db (2.10.2-1) ...
Installed Python pip
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  python3-pip-whl python3-setuptools-whl
The following NEW packages will be installed:
  python3-pip-whl python3-setuptools-whl python3.10-venv
0 upgraded, 3 newly installed, 0 to remove and 6 not upgraded.
Need to get 0 B/2473 kB of archives.
After this operation, 2882 kB of additional disk space will be used.
Selecting previously unselected package python3-pip-whl.
(Reading database ... ^M(Reading database ... 5%^M(Reading database ... 10%^M(Reading
Preparing to unpack .../python3-pip-whl_22.0.2+dfsg-1_all.deb ...
Unpacking python3-pip-whl (22.0.2+dfsg-1) ...
Selecting previously unselected package python3-setuptools-whl.
Preparing to unpack .../python3-setuptools-whl_59.6.0-1.2_all.deb ...
Unpacking python3-setuptools-whl (59.6.0-1.2) ...
Selecting previously unselected package python3.10-venv.
Preparing to unpack .../python3.10-venv_3.10.4-3ubuntu0.1_amd64.deb ...
Unpacking python3.10-venv (3.10.4-3ubuntu0.1) ...
Setting up python3-setuptools-whl (59.6.0-1.2) ...
Setting up python3-pip-whl (22.0.2+dfsg-1) ...
Setting up python3.10-venv (3.10.4-3ubuntu0.1) ...
Installed Python venv
Jenkins Started
Installation successful

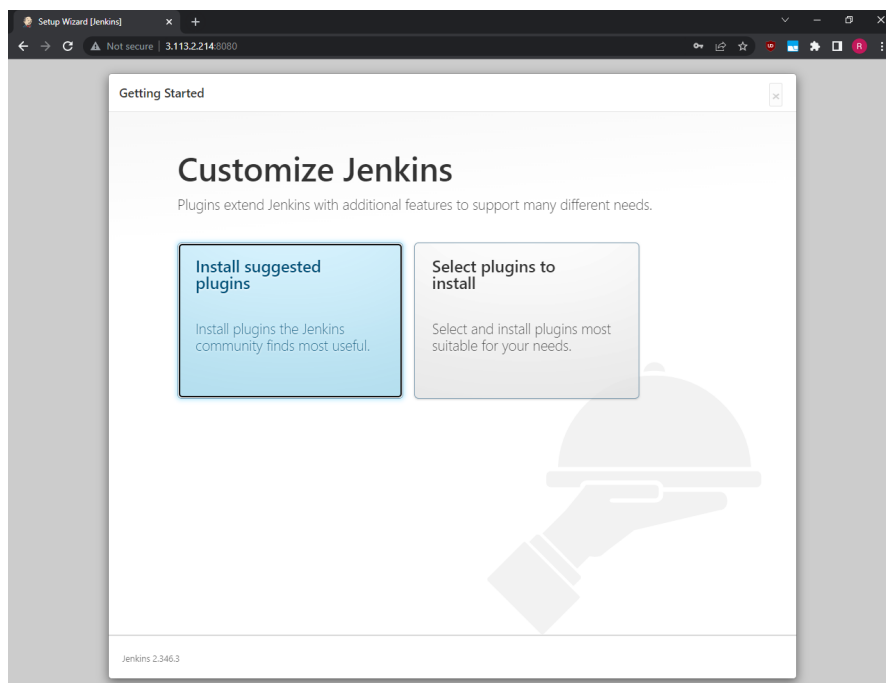
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^N Replace    ^U Paste      ^J Justify
```

# Configuring Jenkins on the EC2

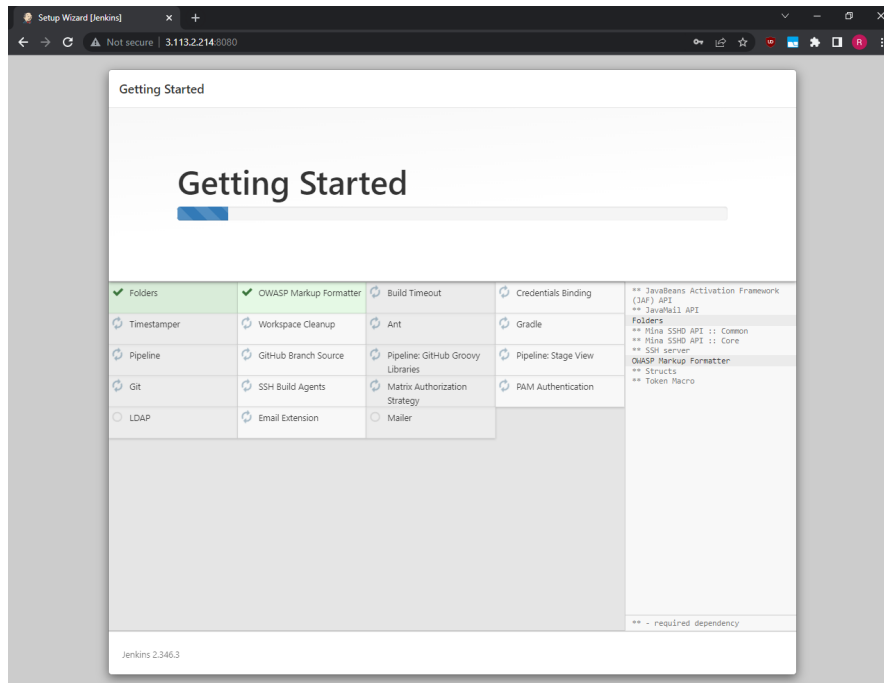
I navigated to the webpage of Jenkins and port 8080 such as **http://3.113.2.214:8080/** and entered the **secret password** located in the file **/var/lib/jenkins/secrets/initialAdminPassword** and clicked **Continue**.



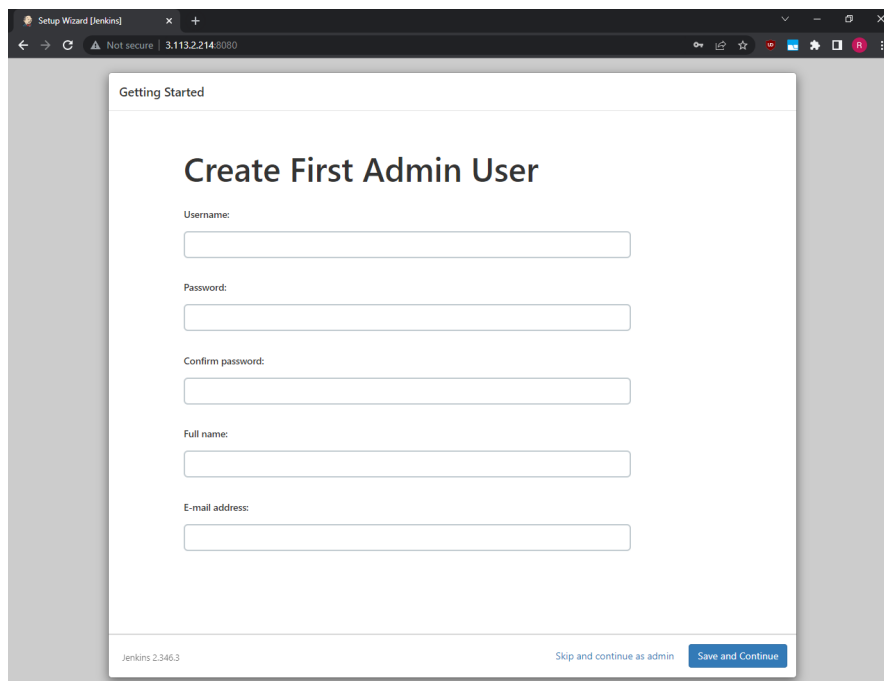
Then I needed to install plugins so I selected **install suggested plugins**.



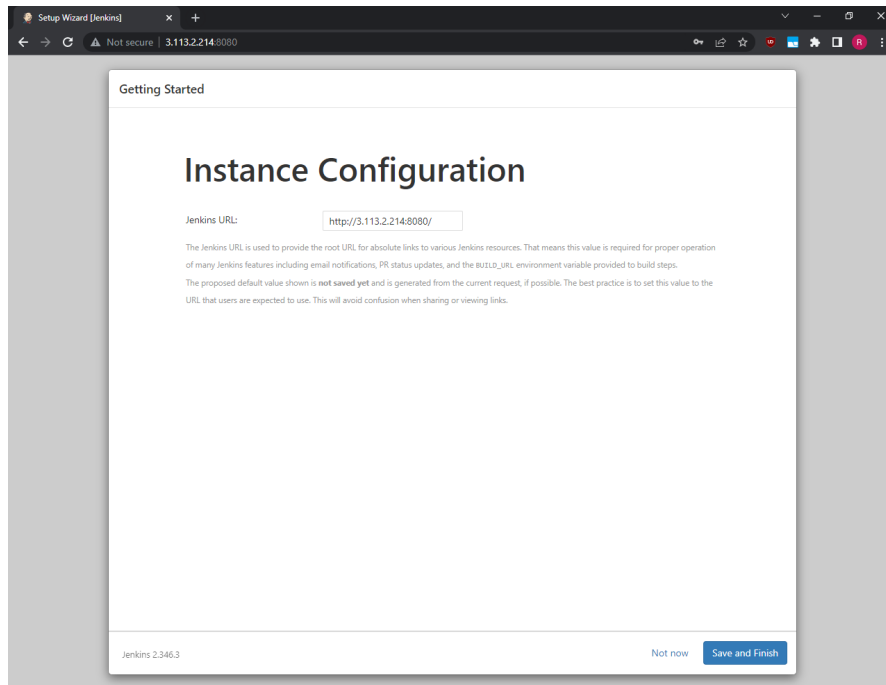
I had to **wait** for the plugins to download and install themselves. **No action is needed.**



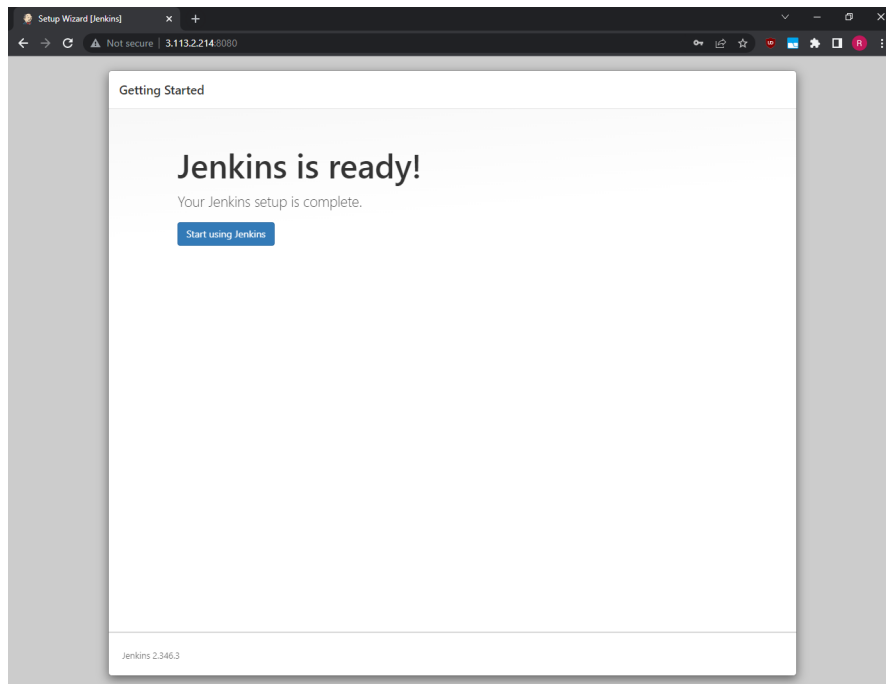
After that I created the **first admin user** by inputting my information and clicking **Save and Continue**.



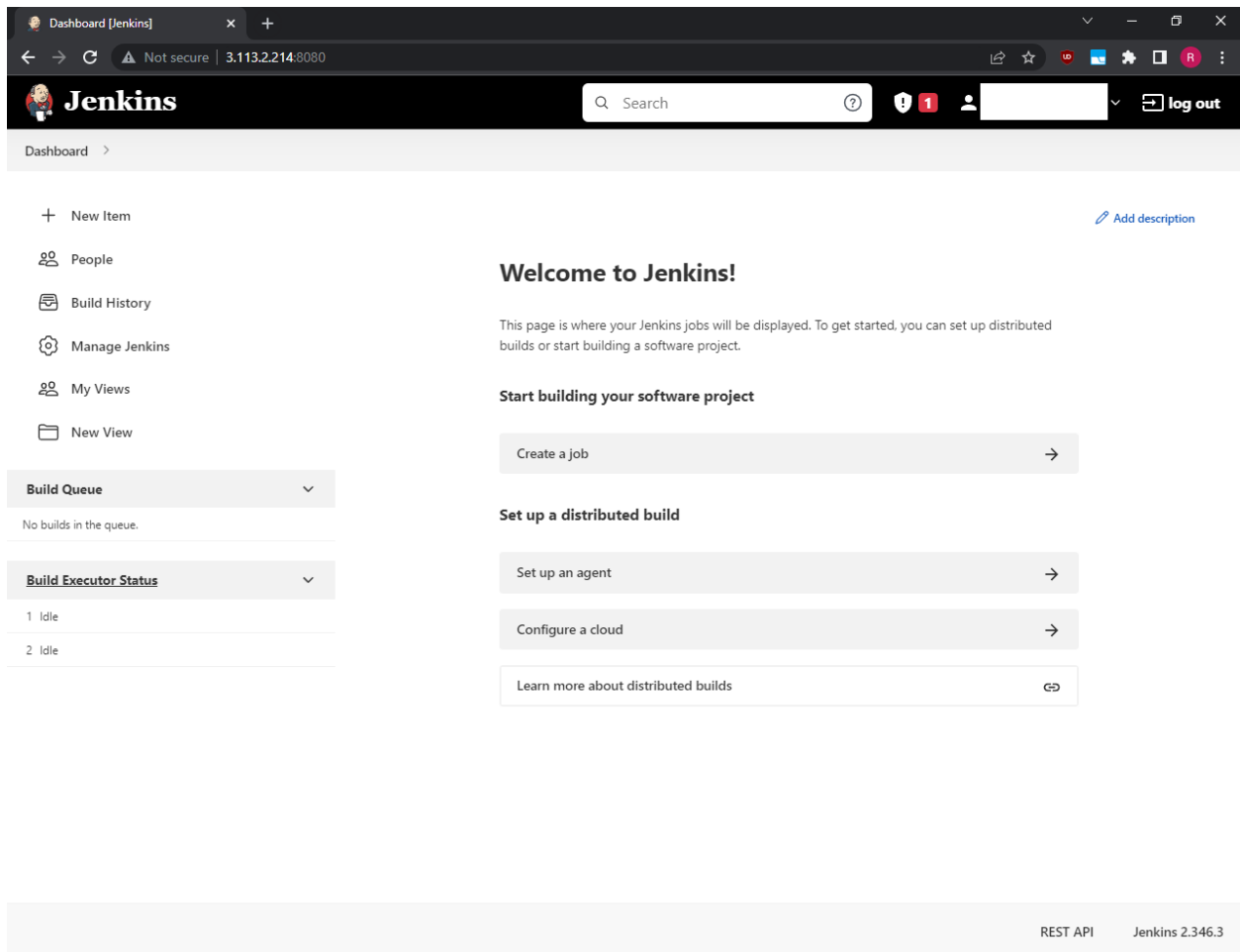
When the user is created then I had to configure the instance specifically the **Jenkins URL** which is the root URL. The default is good so I left it alone and clicked **Save and Finish**.



This marks the **end of configuring** Jenkins. I clicked **Start using Jenkins**.



After that is this page which is the **main screen of Jenkins**. This part is completed for now. **Optionally** I could have set up the **Amazon EC2 plugin** and **Configure a Cloud** but I felt it was **unnecessary**.



The screenshot shows the Jenkins Dashboard in a web browser. The browser's address bar displays 'Not secure | 3.113.2.214:8080'. The Jenkins header includes the logo, a search bar, a notification bell with a red '1', a user profile icon, and a 'log out' button. The left sidebar contains navigation links: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'New View'. Below these are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two 'Idle' executors). The main content area features a 'Welcome to Jenkins!' message, a brief description of the dashboard's purpose, and a 'Start building your software project' section with a 'Create a job' button. Below this is a 'Set up a distributed build' section with buttons for 'Set up an agent', 'Configure a cloud', and a link to 'Learn more about distributed builds'. The footer at the bottom right indicates 'REST API' and 'Jenkins 2.346.3'.

Dashboard [Jenkins] x +

← → ↻ ⚠ Not secure | 3.113.2.214:8080

Jenkins 🔍 Search ? ! 1 👤 [User] ⌵ log out

Dashboard >

+ New Item [Add description](#)

👤 People

📅 Build History

⚙️ Manage Jenkins

👤 My Views

📁 New View

**Build Queue** ▾

No builds in the queue.

**Build Executor Status** ▾

1 Idle

2 Idle

## Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

### Start building your software project

Create a job →

### Set up a distributed build

Set up an agent →

Configure a cloud →

Learn more about distributed builds ↗

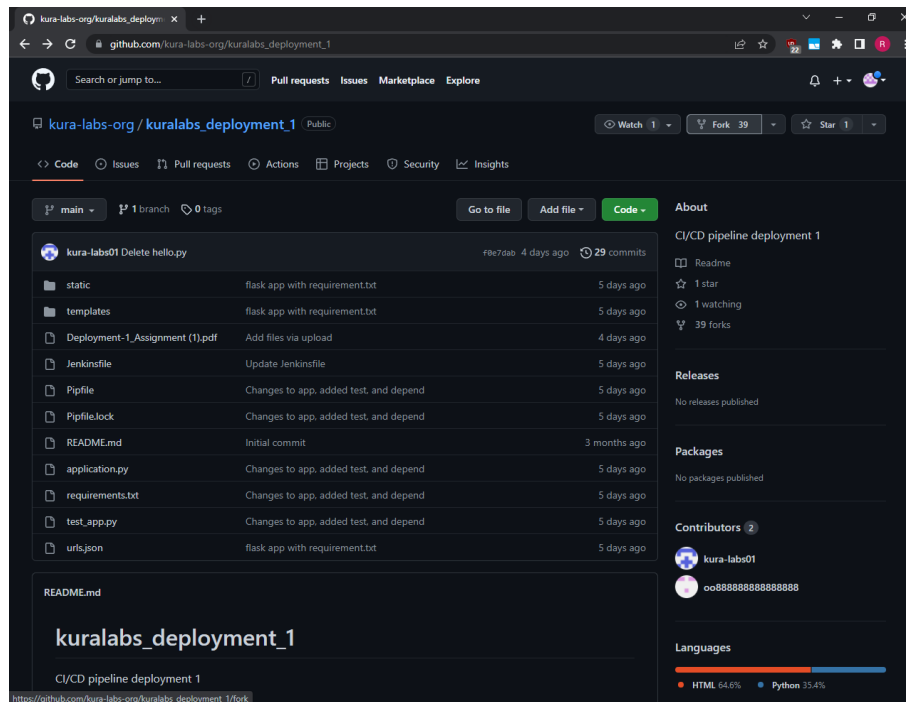
REST API Jenkins 2.346.3



# Forking the Deployment Repository

I navigated to the repository page

([https://github.com/kura-labs-org/kuralabs\\_deployment\\_1](https://github.com/kura-labs-org/kuralabs_deployment_1)) and clicked **Fork** on the right.



Then after all the information was correct I clicked **Create fork**.

## Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner \*

Repository name \*

/ kuralabs\_deployment\_1

By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

CI/CD pipeline deployment 1

☒ Copy the `main` branch only

Contribute back to kura-labs-org/kuralabs\_deployment\_1 by adding your own branch. [Learn more.](#)

You are creating a fork in your personal account.

Create fork

When the fork was created I had **my own copy of the repository**. This is the end of this part.

The screenshot shows a GitHub repository page for 'kuralabs\_deployment\_1', which is a fork of 'kura-labs-org/kuralabs\_deployment\_1'. The repository is public and has 39 forks and 0 stars. The main branch is 'main', and there are 2 branches and 0 tags. A warning message states 'Your main branch isn't protected' and suggests protecting the branch. The repository is described as 'CI/CD pipeline deployment 1'. The commit history shows a series of commits by 'kura-labs01', including 'Delete hello.py', 'static', 'templates', 'Deployment-1\_Assignment (1).pdf', 'Jenkinsfile', 'Pipfile', 'Pipfile.lock', 'README.md', 'application.py', 'requirements.txt', and 'test\_app.py'. The file list on the right shows the repository's structure, including 'static', 'templates', 'Deployment-1\_Assignment (1).pdf', 'Jenkinsfile', 'Pipfile', 'Pipfile.lock', 'README.md', 'application.py', 'requirements.txt', and 'test\_app.py'. The 'Languages' section shows that the repository is primarily HTML (64.6%) and Python (35.4%).

Search or jump to...

Pull requests Issues Marketplace Explore

/kuralabs\_deployment\_1 Public

forked from kura-labs-org/kuralabs\_deployment\_1

Code Pull requests Actions Projects Wiki Security Insights Settings

main 2 branches 0 tags

Go to file Add file Code

About

CI/CD pipeline deployment 1

Readme

0 stars

0 watching

39 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

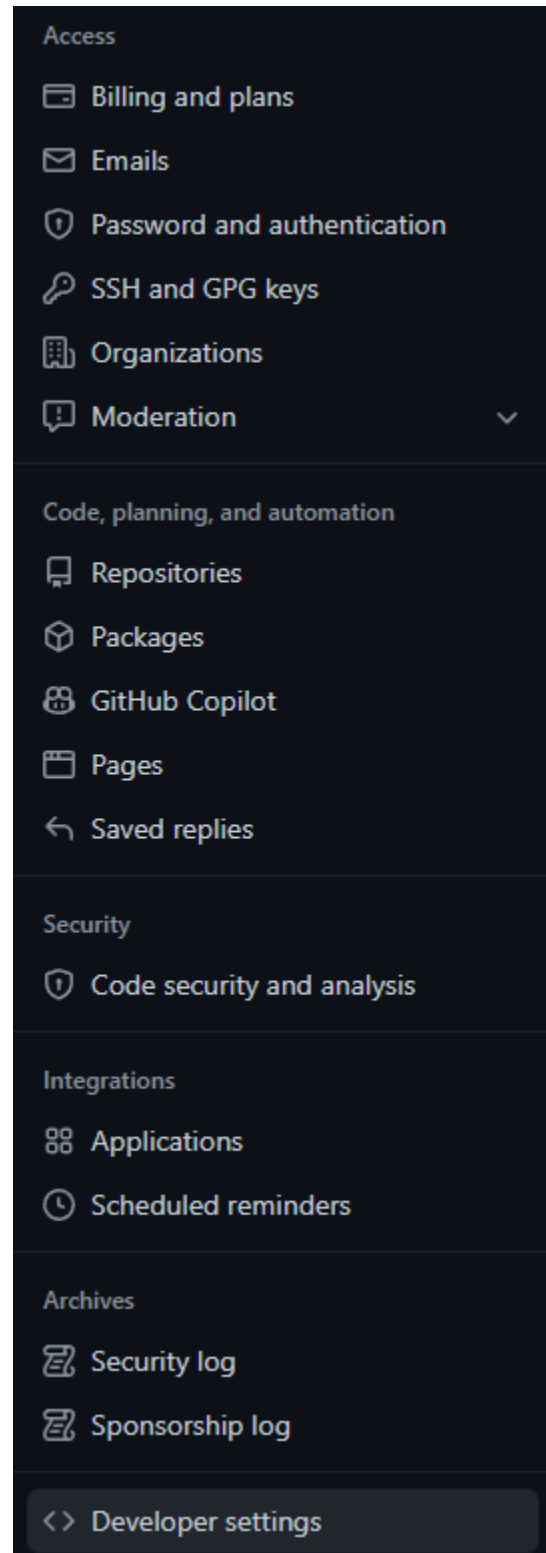
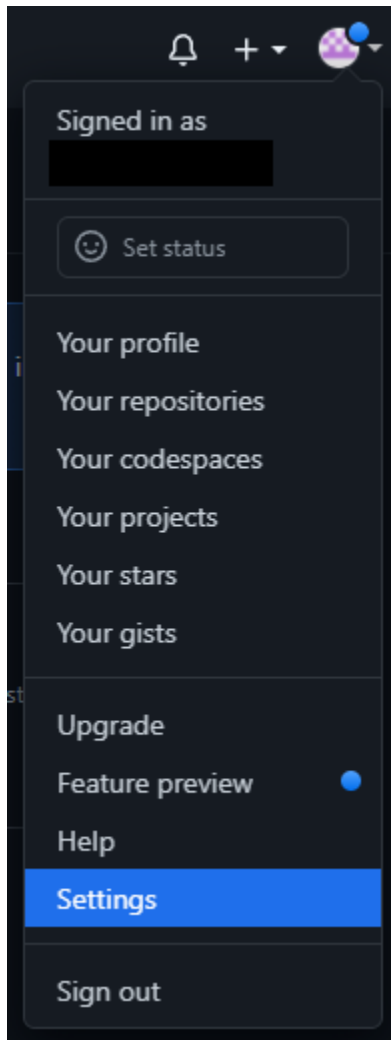
HTML 64.6% Python 35.4%

Commit history:

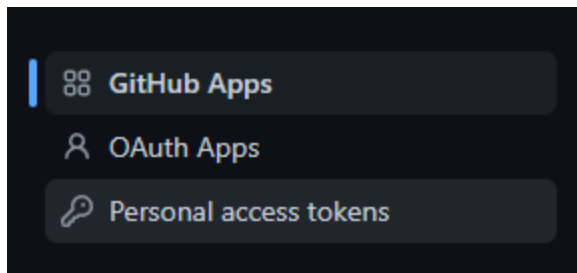
Commit	Message	Time
kura-labs01	Delete hello.py	f0e7dab 4 days ago 29 commits
static	flask app with requirement.txt	5 days ago
templates	flask app with requirement.txt	5 days ago
Deployment-1_Assignment (1).pdf	Add files via upload	4 days ago
Jenkinsfile	Update Jenkinsfile	5 days ago
Pipfile	Changes to app, added test, and depend	5 days ago
Pipfile.lock	Changes to app, added test, and depend	5 days ago
README.md	Initial commit	3 months ago
application.py	Changes to app, added test, and depend	5 days ago
requirements.txt	Changes to app, added test, and depend	5 days ago
test_app.py	Changes to app, added test, and depend	5 days ago

# Github Personal Access Token

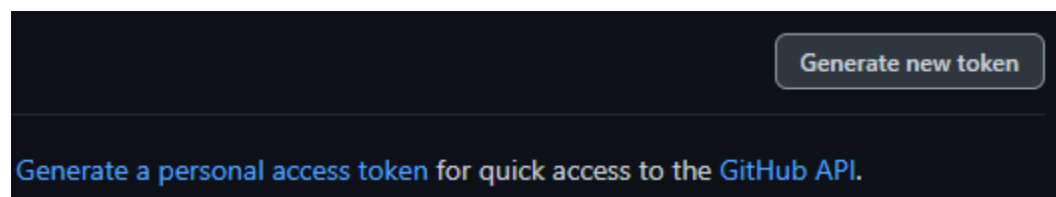
I went to **Github** and then clicked my **Account Icon** on the right for the options. Then click **Setting** then on the new page click **Developer settings** on the left.



On the left of the new page click **Personal access tokens**.



In the middle of the new page click **Generate new token**.



Write a **Note** on what the token is for then click **repo** for the scope.

## New personal access token

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

**Note**

Jenkins

What's this token for?

**Expiration \***

30 days

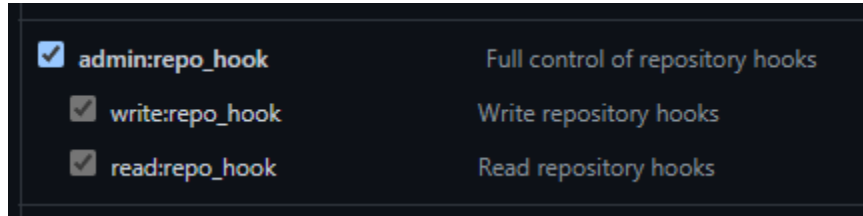
The token will expire on Fri, Sep 30 2022

**Select scopes**

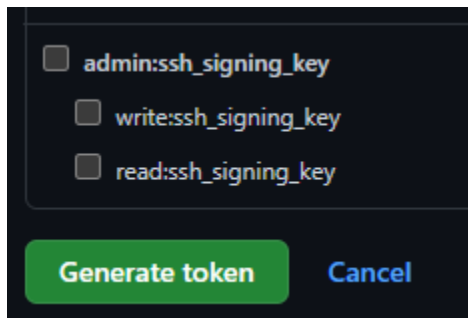
Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> <b>repo</b>	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

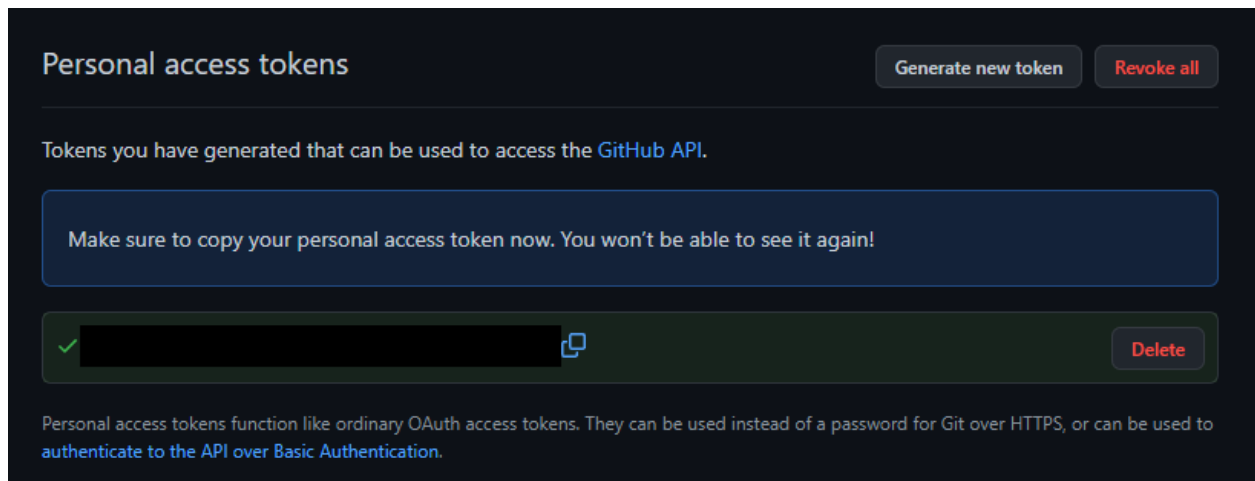
Further down click **admin:repo\_hook** to add to the scope.



After that click **Generate token** to generate the Personal access token.

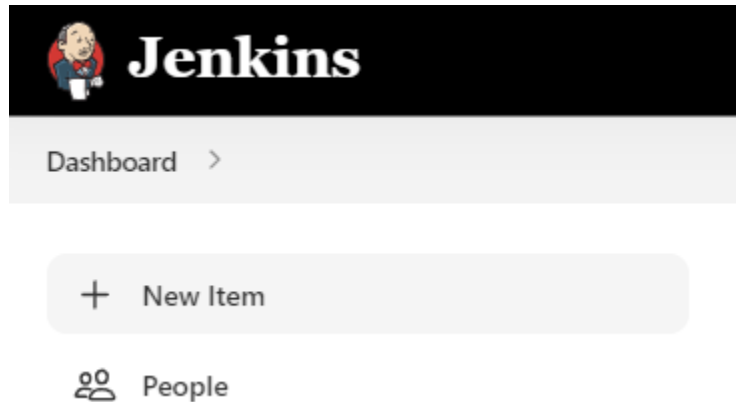


When it was generated I had the following(censored) **Personal access token**.

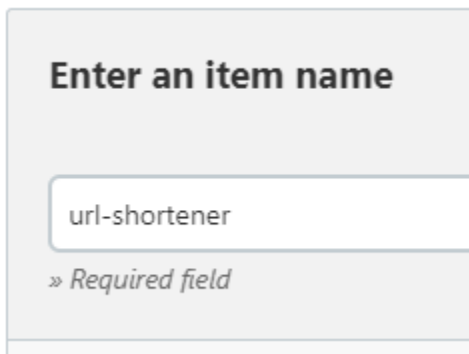


# Create a Multibranch Build on Jenkins

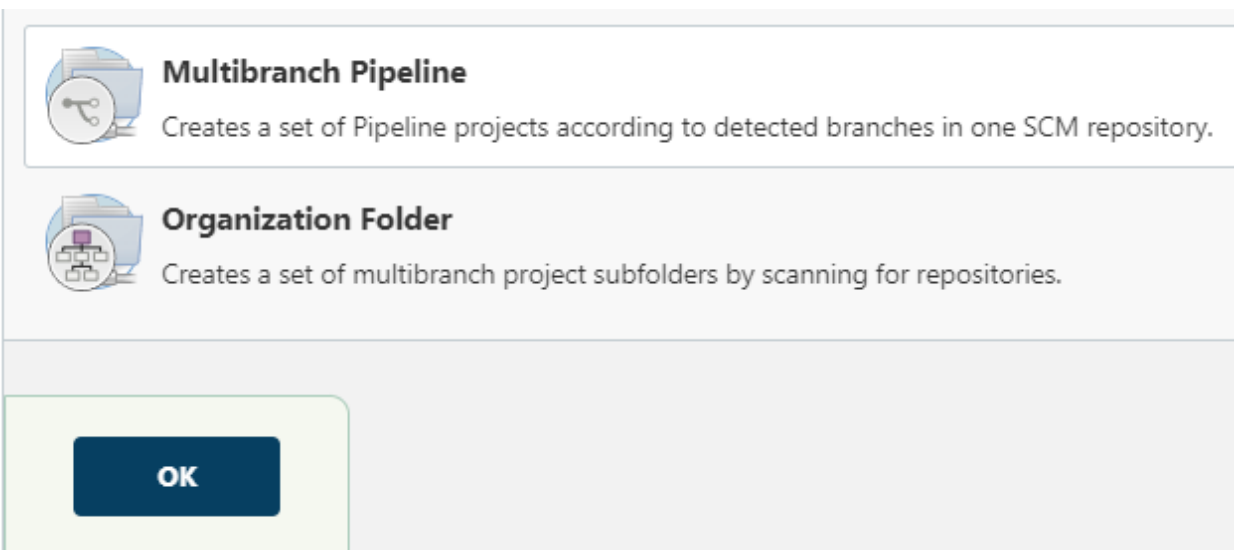
I logged into Jenkins and clicked **New Item** on the left.



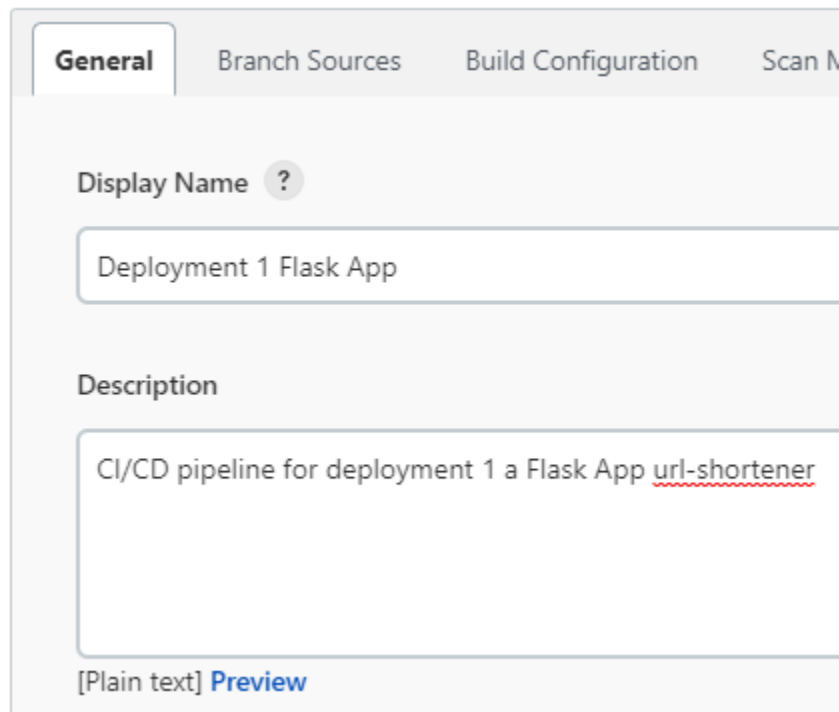
Then I entered a **name(url-shortener)**.

A screenshot of the 'Enter an item name' form in Jenkins. It features a text input field containing 'url-shortener'. Below the field is a label '» Required field'.

After that I selected the **Multibranch Pipeline** option and clicked **OK**.

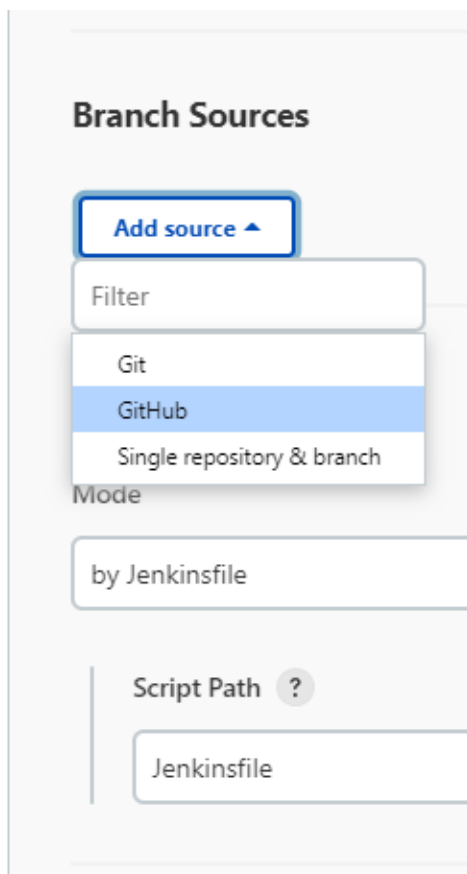


Now I had to select a **Display Name** and **Description**.



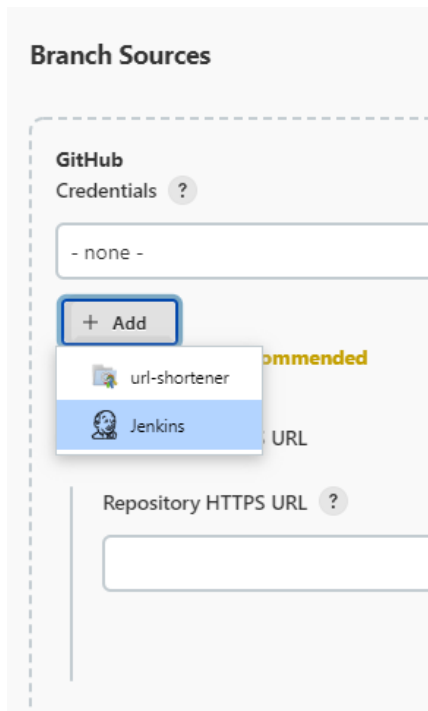
The screenshot shows the 'General' tab of a Jenkins configuration page. It features two input fields: 'Display Name' with a help icon and a text box containing 'Deployment 1 Flask App'; and 'Description' with a text box containing 'CI/CD pipeline for deployment 1 a Flask App url-shortener'. Below the description field, there is a '[Plain text] Preview' link.

With that done I went to the **Branch Sources** section and clicked **Add source** then **GitHub** to add the deployment GitHub Repository([https://github.com/kura-labs-org/kuralabs\\_deployment\\_1](https://github.com/kura-labs-org/kuralabs_deployment_1))

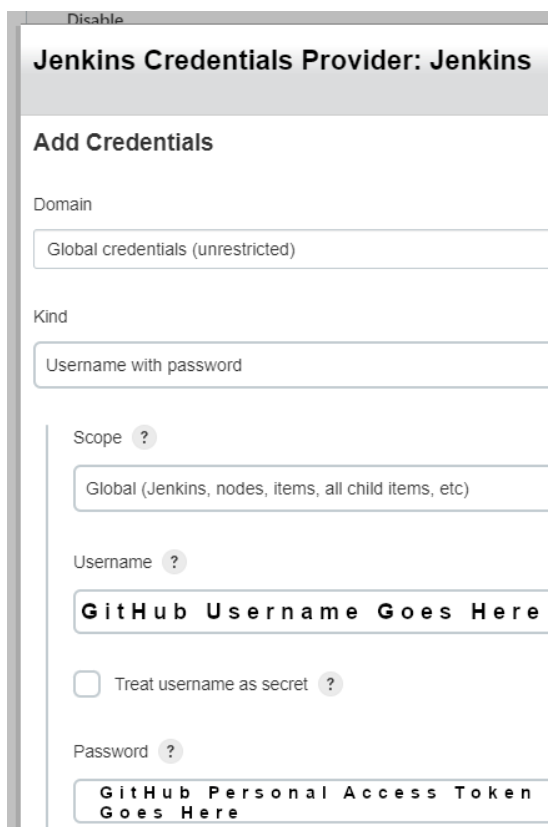


The screenshot shows the 'Branch Sources' section of the Jenkins configuration page. It includes an 'Add source' button with a dropdown arrow, a 'Filter' input field, and a dropdown menu with options: 'Git', 'GitHub' (highlighted), and 'Single repository & branch'. Below this is a 'Mode' input field containing 'by Jenkinsfile'. At the bottom, there is a 'Script Path' field with a help icon and a text box containing 'Jenkinsfile'.

This next step seems optional because I don't believe I need **credentials** to grab a **public project** from Github. The original and my fork are **both public** but I set the credentials anyway as **practice**. I clicked **Add** then **Jenkins**.

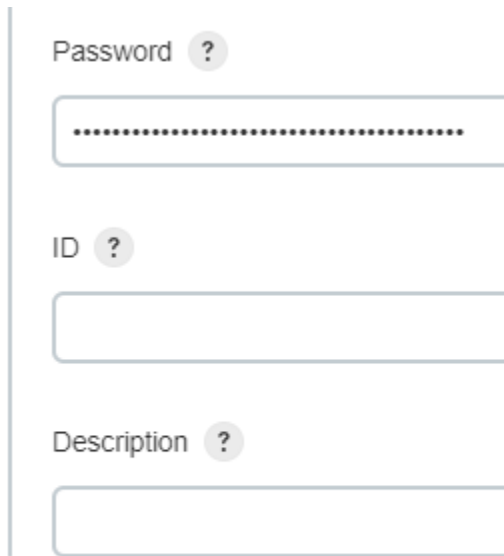


The Jenkins Credentials Provider popped up and I left it on **Username and password for Kind** and set the **Username** as my **GitHub username** and **Password** as the **GitHub Personal access token** I previously generated.



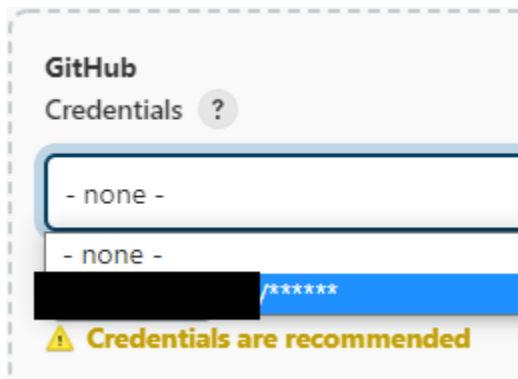


When all the information was entered I clicked **Add** to add the credentials.



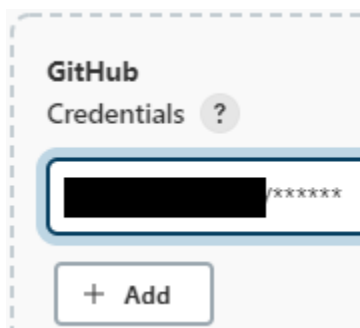
A form for adding credentials. It has three input fields: "Password" with a question mark icon, "ID" with a question mark icon, and "Description" with a question mark icon. The "Password" field contains a series of dots representing masked text. Below the fields are two buttons: "Add" (dark blue) and "Cancel" (light blue).

Then once it was added I saw it as an **option to pick which credentials to use**.



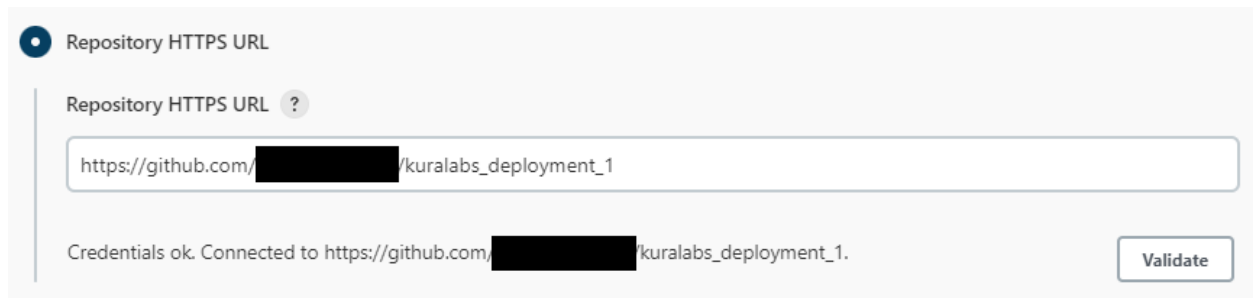
A dropdown menu for selecting credentials. The title is "GitHub Credentials" with a question mark icon. The dropdown list shows three options: "- none -", "- none -", and a selected option with a black box and "/\*\*\*\*\*". Below the dropdown is a yellow warning message: "⚠ Credentials are recommended".

I **selected** the credentials I just added.



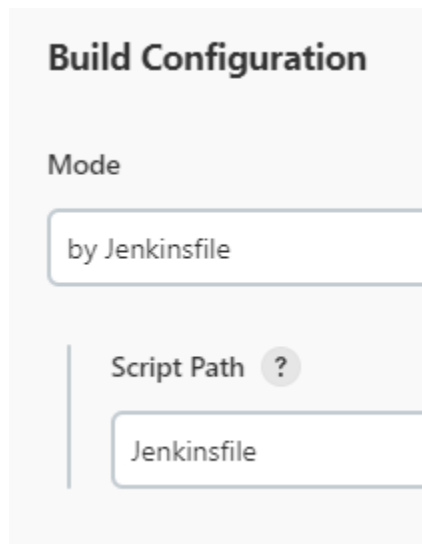
A dropdown menu for selecting credentials. The title is "GitHub Credentials" with a question mark icon. The dropdown list shows one option with a black box and "/\*\*\*\*\*". Below the dropdown is a button with a plus icon and the text "Add".

After that I entered the **GitHub url of the repository to build**. I could have used [https://github.com/kura-labs-org/kuralabs\\_deployment\\_1](https://github.com/kura-labs-org/kuralabs_deployment_1) or my fork of that, I used **my fork** of it then clicked **Validate**.



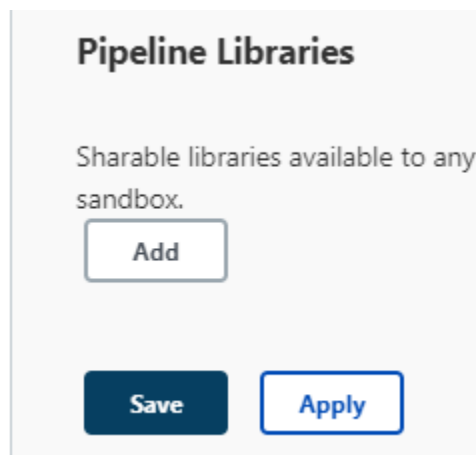
The screenshot shows a form titled "Repository HTTPS URL" with a sub-label "Repository HTTPS URL ?". A text input field contains the URL "https://github.com/[redacted]/kuralabs\_deployment\_1". Below the input field, a status message reads "Credentials ok. Connected to https://github.com/[redacted]/kuralabs\_deployment\_1." A "Validate" button is located at the bottom right of the form.

I scrolled down and made sure the **Build Configuration** was left at the default(**Jenkinsfile**) below.



The screenshot shows the "Build Configuration" section. Under the "Mode" label, a dropdown menu is set to "by Jenkinsfile". Below this, under the "Script Path ?" label, a text input field contains "Jenkinsfile".



Next I scrolled to the bottom and clicked **Apply** and **Save** to save everything inputted.







The screenshot shows the "Pipeline Libraries" section. It includes the text "Sharable libraries available to any sandbox." and an "Add" button. At the bottom, there are two buttons: "Save" and "Apply".

# The Build on Jenkins

When I finished creating the multibranch build it started **building**(I had two branches).

Build Executor Status		
1	Idle	
2	Idle	
<hr/>		
Deployment 1 Flask App » main	#1	
<hr/>		
Deployment 1 Flask App » original	#1	
<hr/>		

When it was done building it. It **passed the test** and I got the green check mark.

S	W	Name ↓
		main
		original

The **url-shortener** built successfully and passed the test stage. This part is done.

## Branch main

Full project name: url-shortener/main



## Stage View

		Declarative: Checkout SCM	Build	test
Average stage times: (Average <u>full</u> run time: ~36s)		2s	20s	2s
<hr/>				
#1	Sep 01 01:08 No Changes	2s	20s	2s



# Deploy to Elastic Beanstalk

The first step I did was to **prepare the zip file** with the code by cloning the repository and zipping the files myself.

```
richard@Executor:/mnt/f/Desktop/Test $ git clone git@github.com:[redacted]:kuralabs_deployment_1.git
Cloning into 'kuralabs_deployment_1'...
remote: Enumerating objects: 115, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 115 (delta 12), reused 35 (delta 12), pack-reused 79
Receiving objects: 100% (115/115), 9.19 MiB | 3.40 MiB/s, done.
Resolving deltas: 100% (59/59), done.
Updating files: 100% (30/30), done.
richard@Executor:/mnt/f/Desktop/Test $ ls
kuralabs_deployment_1
richard@Executor:/mnt/f/Desktop/Test $ cd kuralabs_deployment_1/
richard@Executor:/mnt/f/Desktop/Test/kuralabs_deployment_1 main $ ls
'Deployment-1 Assignment (1).pdf' Jenkinsfile Pipfile Pipfile.lock README.md application.py requirements.txt
test.py test_app.py urls.json
richard@Executor:/mnt/f/Desktop/Test/kuralabs_deployment_1 main $ zip -q ../kuralabs_deployment_1 -r * .[^.]*
richard@Executor:/mnt/f/Desktop/Test/kuralabs_deployment_1 main $ cd ..
richard@Executor:/mnt/f/Desktop/Test $ ls
kuralabs_deployment_1 kuralabs_deployment_1.zip
richard@Executor:/mnt/f/Desktop/Test $
```

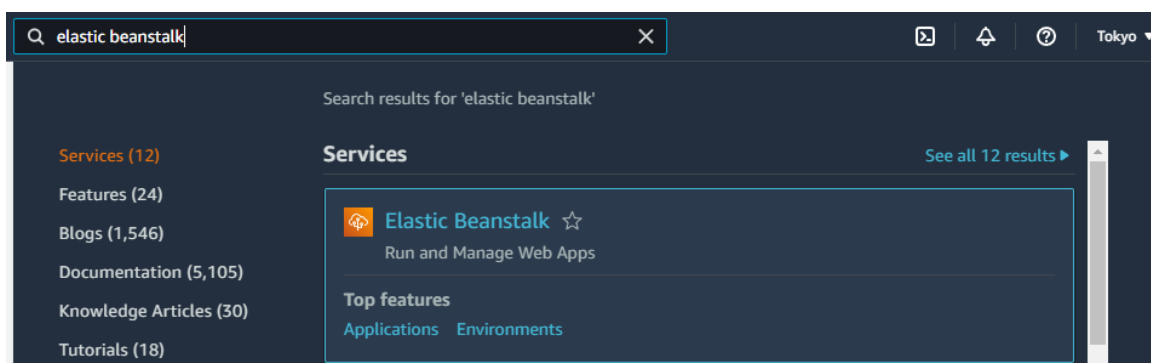
I also wrote a script to do the **same thing** (also on my github as **ziprepo.sh**).

```
1  #!/bin/bash
2
3  #This script is to clone the deployment repository and zip it for the elastic beanstalk
4
5  #The Url of the GitHub repository
6  GitGubUrl="https://github.com/kura-labs-org/kuralabs_deployment_1.git"
7
8  #The name of the repository folder
9  RepoFolderName="kuralabs_deployment_1"
10
11 #The Name of the Zip file we create
12 ZipFileName="kuralabs_deployment_1"
13
14 #Command to clone the GitHub repository
15 GitCloneCMD="git clone $GitGubUrl"
16
17 #Command to zip the cloned repository
18 ZipCMD="zip -q ../"$ZipFileName" -r * .[^.]*"
19
20 #Check if the zip command is installed
21 which zip > /dev/null 2>&1
22
23 #If it's not installed exit
24 if [ $? -ne 0 ]; then
25     echo "Zip command is not installed"
26     exit 1
27 fi
28
29 #Clone the repository and zip it
30 $GitCloneCMD && cd $RepoFolderName && $ZipCMD && cd ..
31
32 #Check if the commands worked
33 if [ $? -ne 0 ]; then
34     echo "Clone and Zip Error"
35     exit 1
36 fi
37
38 #successful
39 echo "Zip successful"
40 exit 0
41 |
```

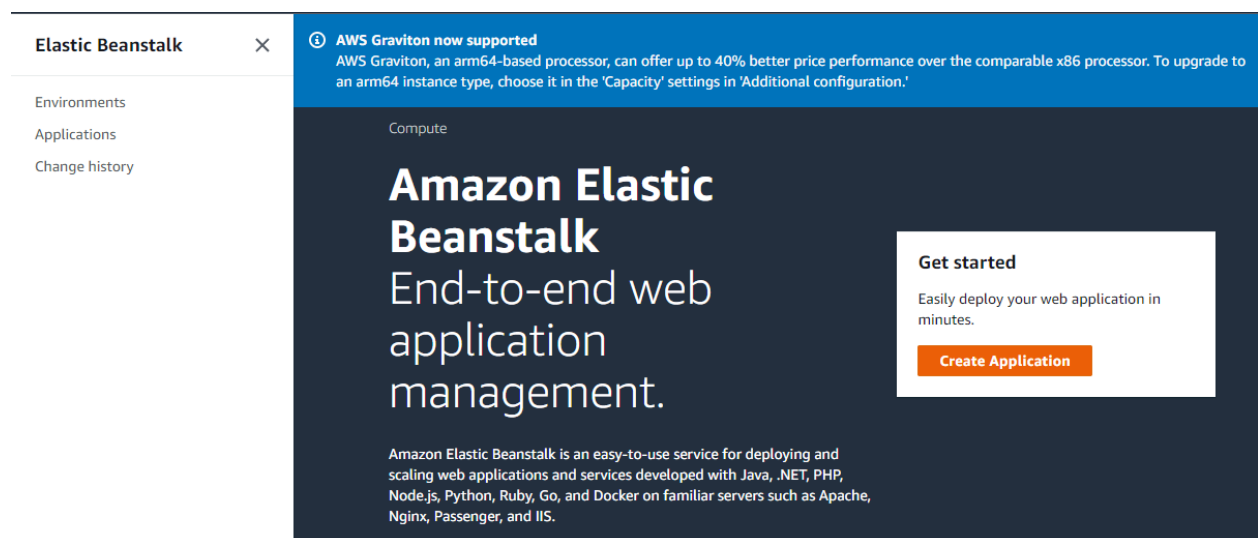
Below is an example of the **output** from the above script.

```
Ubuntu 20.04 on Windows
richard@Tokyo:~/Jenkins-Setup main $ ls
README.md  installjenkins.log  installjenkins.sh  runinstalljenkins.sh  ziprepo.sh
richard@Tokyo:~/Jenkins-Setup main $ ./ziprepo.sh
Cloning into 'kuralabs_deployment_1'...
remote: Enumerating objects: 115, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 115 (delta 12), reused 35 (delta 12), pack-reused 79
Receiving objects: 100% (115/115), 9.19 MiB | 20.10 MiB/s, done.
Resolving deltas: 100% (59/59), done.
Zip successful
richard@Tokyo:~/Jenkins-Setup main $ ls
README.md          installjenkins.sh    kuralabs_deployment_1.zip  ziprepo.sh
installjenkins.log  kuralabs_deployment_1  runinstalljenkins.sh
richard@Tokyo:~/Jenkins-Setup main $
```

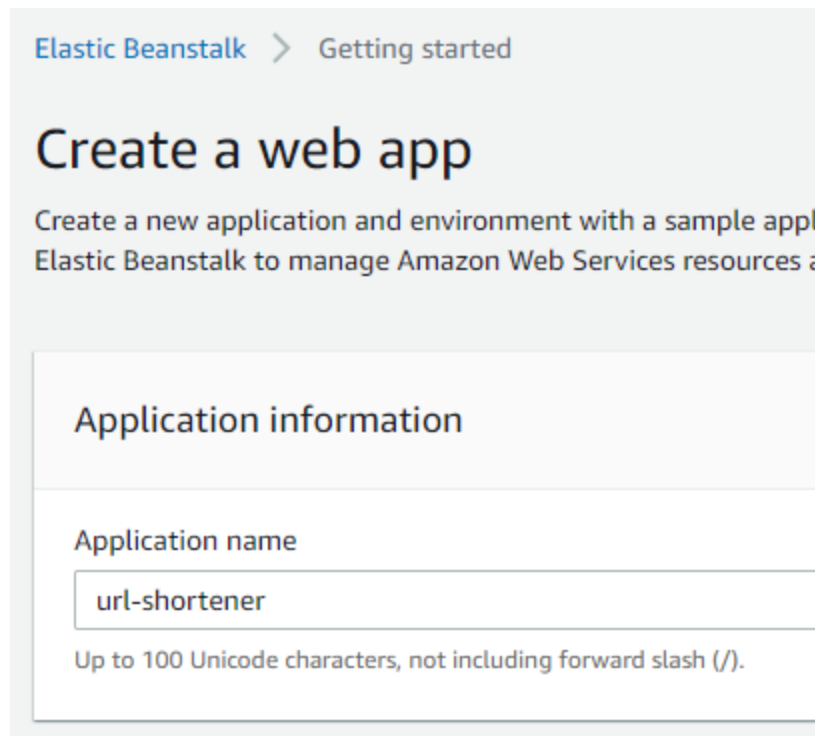
Then I went to the **AWS website**, logged in and searched for **Elastic Beanstalk** and clicked on **Elastic Beanstalk** on the **Services** list.



After the page loaded I clicked on **Create Application** on the right of the page.



On the new page it loaded I entered an **Application name**(url-shortener).



Elastic Beanstalk > Getting started

## Create a web app

Create a new application and environment with a sample application. Elastic Beanstalk to manage Amazon Web Services resources and infrastructure.

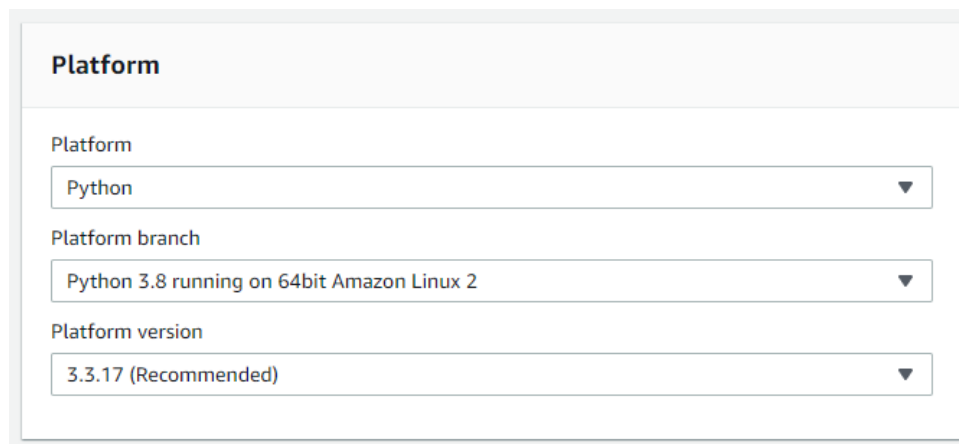
### Application information

Application name

url-shortener

Up to 100 Unicode characters, not including forward slash (/).

Then I scrolled down to **Platform** and selected **Python** leaving the default options it fills.



### Platform

Platform

Python

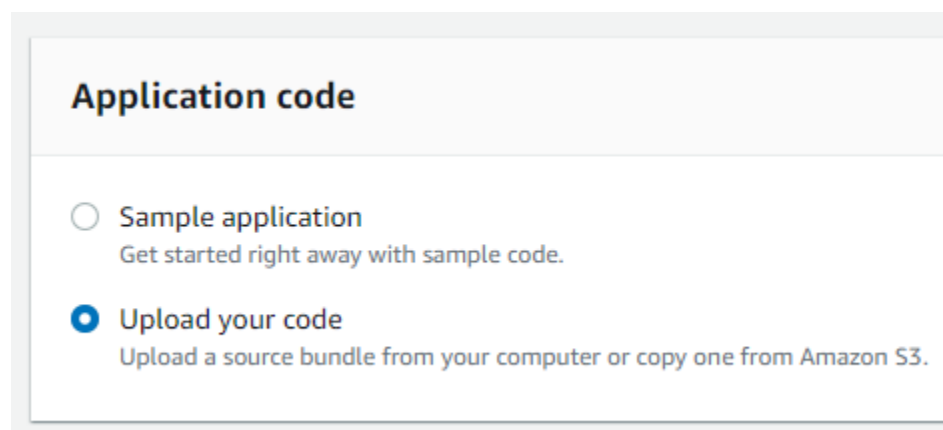
Platform branch

Python 3.8 running on 64bit Amazon Linux 2

Platform version

3.3.17 (Recommended)

Scrolling further down to **Application code** I selected **Upload your code**.

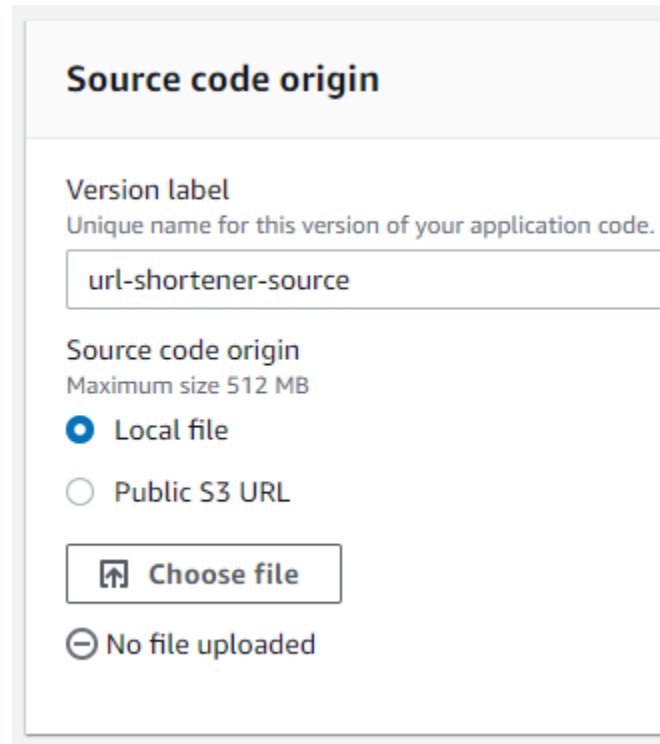


### Application code

☐ Sample application  
Get started right away with sample code.

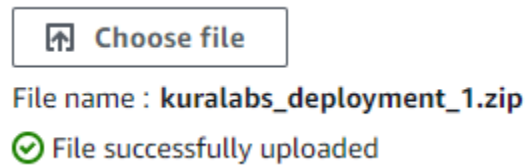
☒ Upload your code  
Upload a source bundle from your computer or copy one from Amazon S3.

When selected it showed some options to upload the code I selected **Local file** and clicked **Choose file** then selected the correct zip file from the pop up and waited for it to upload the zip file.



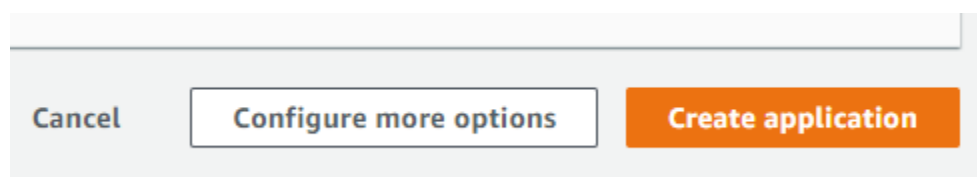
The screenshot shows a configuration window titled "Source code origin". It contains a "Version label" field with the text "url-shortener-source" and a description "Unique name for this version of your application code.". Below this is the "Source code origin" section, which has a "Maximum size 512 MB" limit. Two radio buttons are present: "Local file" (which is selected) and "Public S3 URL". A "Choose file" button with a folder icon is located below the radio buttons. At the bottom, there is a "No file uploaded" option with a minus icon.

After it finished uploading it showed the **filename with a green checkmark**.



This screenshot shows the confirmation message after a file upload. It features a "Choose file" button with a folder icon. Below the button, the text "File name : kuralabs\_deployment\_1.zip" is displayed. At the bottom, there is a green checkmark icon followed by the text "File successfully uploaded".


With everything correct I clicked **Create application** at the bottom of the page.



The screenshot shows the bottom navigation bar of the application. It contains three buttons: "Cancel", "Configure more options", and "Create application". The "Create application" button is highlighted in orange, while the other two are in a light gray color.

When the application was created it made an **environment** for it and showed the progress so I waited for it to complete.

Elastic Beanstalk > Environments > Urlshortener-env

 **Creating Urlshortener-env**  
This will take a few minutes.


6:37pm Using elasticbeanstalk-ap-northeast-1-498463483397 as Amazon S3 storage bucket for environment data.


6:37pm createEnvironment is starting.

When everything was built it showed the status of **Health** as **Ok**.

Elastic Beanstalk > Environments > Urlshortener-env


### Urlshortener-env

[Urlshortener-env.eba-auutvhqg.ap-northeast-1.elasticbeanstalk.com](https://urlshortener-env.eba-auutvhqg.ap-northeast-1.elasticbeanstalk.com)  (e-csgdjzmr6w)  
Application name: url-shortener

 Refresh

Actions ▼

#### Health



Ok


Causes

#### Running version

url-shortener-source

Upload and deploy

#### Platform



Python 3.8 running on 64bit  
Amazon Linux 2/3.3.17

Change

#### Recent events

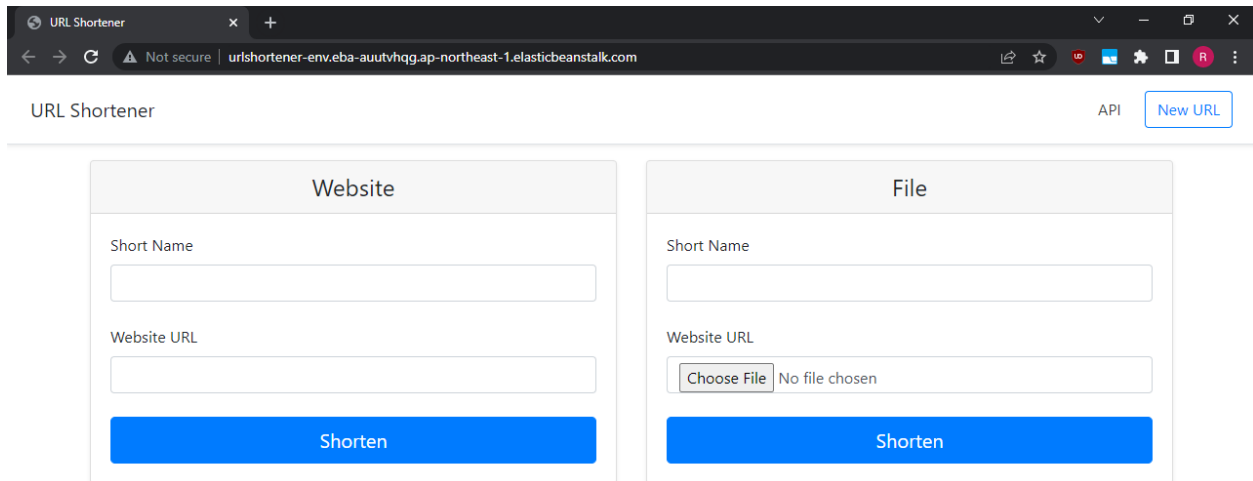
Show all

< 1 >

Time	Type	Details
2022-09-01 18:41:14 UTC-0400	INFO	Successfully launched environment: Urlshortener-env
2022-09-01 18:41:13 UTC-0400	INFO	Application available at Urlshortener-env.eba-auutvhqg.ap-northeast-1.elasticbeanstalk.com.
2022-09-01 18:40:56 UTC-0400	INFO	Instance deployment completed successfully.

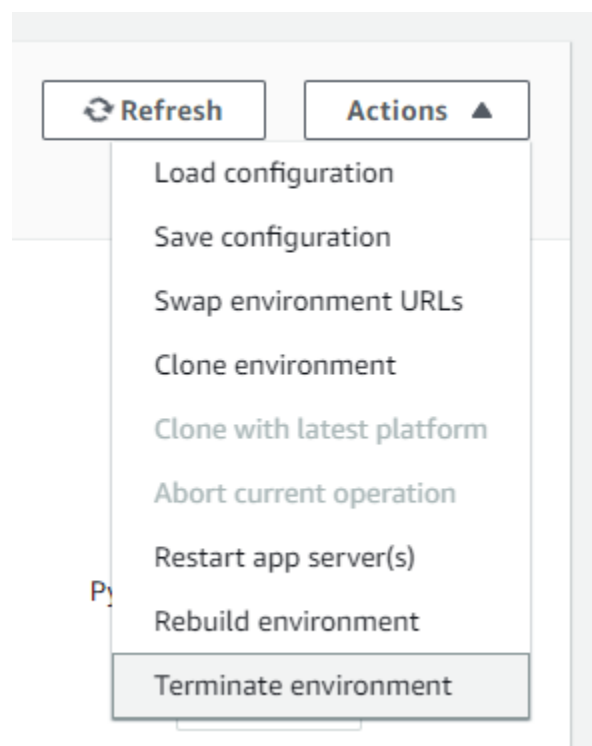


When I opened the **link to the url-shortener app** it created, it worked.



The screenshot shows a web browser window with the title "URL Shortener". The address bar displays "urlshortener-env.eba-auutvhqg.ap-northeast-1.elasticbeanstalk.com". The page has a header with "URL Shortener" on the left, "API" in the middle, and a "New URL" button on the right. The main content area is divided into two panels: "Website" and "File". Each panel contains a "Short Name" input field, a "Website URL" input field, and a blue "Shorten" button. The "File" panel also includes a "Choose File" button and the text "No file chosen".

When I was done I **terminated** the environment.



It asked to **confirm** by typing the name of the environment.

### Confirm environment termination

Permanently terminate **Urlshortener-env**? This action can't be undone.

- Tier: Web Server
- Platform: Python 3.8 running on 64bit Amazon Linux 2/3.3.17
- Version: url-shortener-source
- Last modified: 2022-09-01 18:41:14 UTC-0400

**⚠ Terminating this environment will also terminate its associated resources.**

- URL** - *Urlshortener-env.eba-auutvhqg.ap-northeast-1.elasticbeanstalk.com* will be released.
- Additional resources** - any resources associated with this environment will also be terminated.

Enter the name of the environment to confirm:

This is case sensitive

Cancel **Terminate**

Once that was done it started to terminate it and **eventually it was terminated**. The deployment was **successfully completed**.

# All environments

Actions

Create a new environment

Filter results matching the display values

<

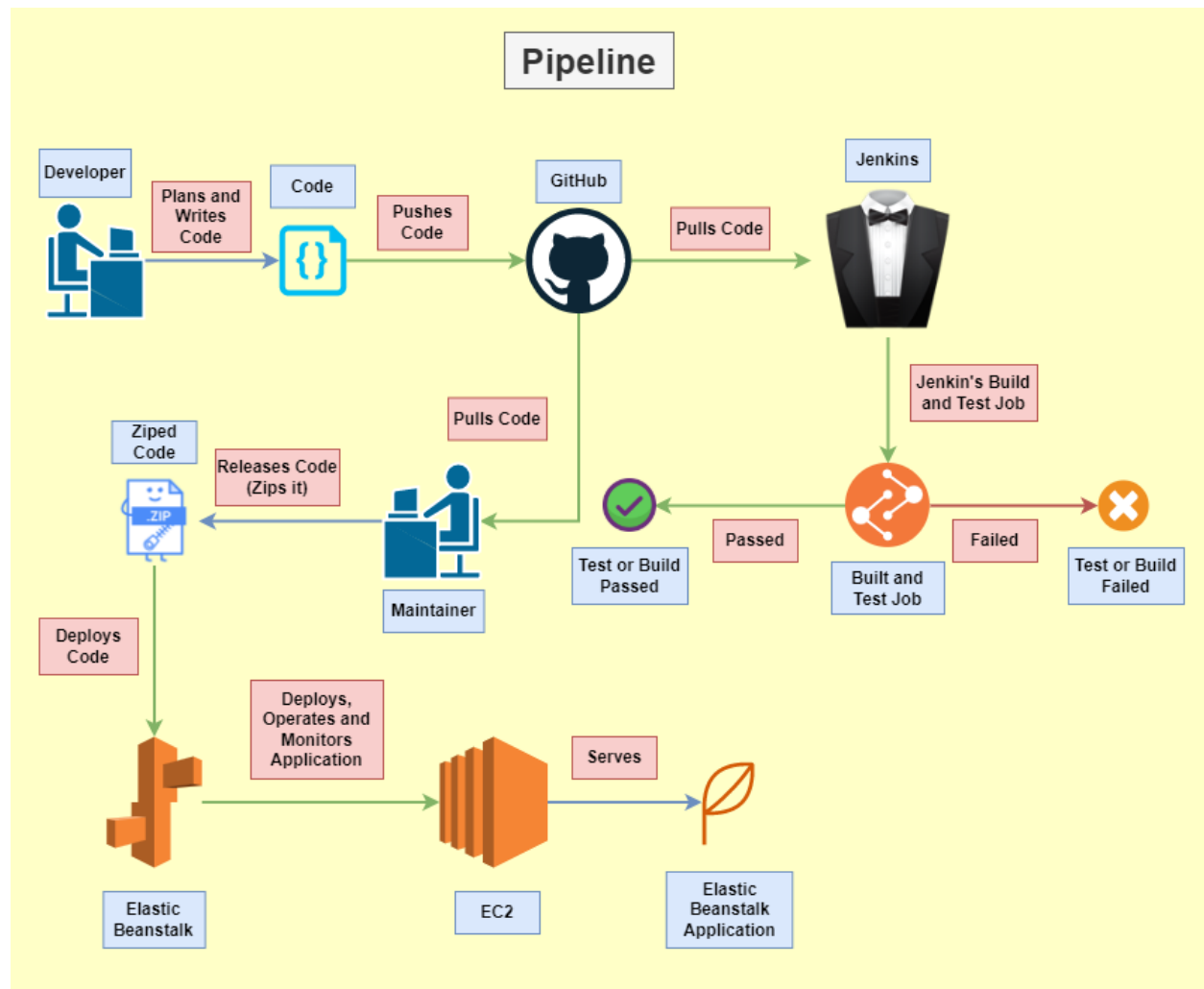
1

>

	Environment name	Health	Application name	Date created	Last modified	URL	Run version
<div></div>	Urlshortener-env (terminated)	-	url-shortener	2022-09-01 18:37:49 UTC-0400	2022-09-01 19:14:12 UTC-0400	Urlshortener-env.eba-auutvhqg.ap-northeast-1.elasticbeanstalk.com	url-shortener-source

# Pipeline Diagram

This is how I understand the process I **experienced** would look as a pipeline.



# What Could I Improve

If I could learn the **AWS CLI** I assume I could automate creating an EC2 for the Jenkins server to run.

Another part that could be automated is the **configuration of the Jenkins server**, there probably exists a way to do it from the command line instead of using the web page.

There is probably a way to set it up so that Jenkins automatically creates or updates the Elastic Beanstalk application and environment **after a build passes**. I think this might be achieved by adding another stage for deployment or release and using a script that clones the repo, zips it and uses AWS CLI to create or update the Elastic Beanstalk.