# Deployment 1 - Documentation

## Creating EC2 Instance

* Created an EC2 instance using Ubuntu as the image
* For Key Pair, I generated a new RSA token named kl-deployment1 and downloaded it in my VM
* Under Network Setting, I added two more security group rules to have ports 80 (HTTP), 8080 (custom TCP) on top of the default port 22

Graphical user interface, application

Description automatically generated

* To connect to the EC2, I first changed the .pem file permission to 400 (read only)
* Then I ran this command to connect: **ssh -I kl-deployment1.pem ubuntu@3.84.52.108**
* To connect to the EC2 using my own SSH key (and not the .pem file generated by AWS):
  + Run **ssh-keygen** to generate a pair of private and public key (kl-deployment1)
  + Copy the content of the public key and paste it inside the EC2 file: **/home/ubuntu/.ssh/authorized\_keys**
  + Now, I only need to run **ssh ubuntu@3.84.52.108** to connect to EC2
* To make life even easier:
  + Make a **config** file in **~/.ssh** and inside:Text

    Description automatically generated
  + Now, I only need to run **ssh kl-deployment1** to connect to EC2

## Installing Jenkins on EC2 (kl-deployment1)

* I ran the provided commands to install Jenkins on my newly created EC2 instance (kl-deployment1)
* One trouble I had was running this command: Text

  Description automatically generated

The issue was that I did not put a space between – and https

* Once resolved, everything ran accordingly and was able to check status of Jenkins running on my EC2:Text

  Description automatically generated

## Configuring Jenkins

* Connected to <http://ec2-3-84-52-108.compute-1.amazonaws.com:8080>
* Entered password found in **/var/lib/Jenkins/secrets/initialAdminPassword**
* Installed suggested plugins:

Graphical user interface, text, application

Description automatically generated

* Created an admin account, then went into **Manage Jenkins -> Manage Plugins** -> downloaded **Amazon EC2** plugin without restart
* Once done, I went back to the dashboard, then Configure Cloud > Amazon EC2
  + For Amazon EC2 Credentials:
    - AWS > IAM > Your Security Credentials > Access Keys > Create New Access Key
  + For EC2 Key Pair’s Private Key:
    - AWS > EC2 > Network & Security > Key Pairs > Create key pair
    - Tested connection which returned Success:

Graphical user interface, text, application, email, Teams

Description automatically generated

## Install Virtual Environment

* I was able to run **sudo apt install python3-pip** command just fine, but I had issues with the second package (python3-10-venv)
* Resolved this by changing the first dash into a period: **sudo apt install python3.10-venv**

## Connect GitHub to Jenkins Server

* Forked the Deployment repo to my personal GitHub account
* Created my access token from developer settings page with provided token permissions
* Entered my GitHub username and the generated token as password, then validated connection:

Graphical user interface, text, application

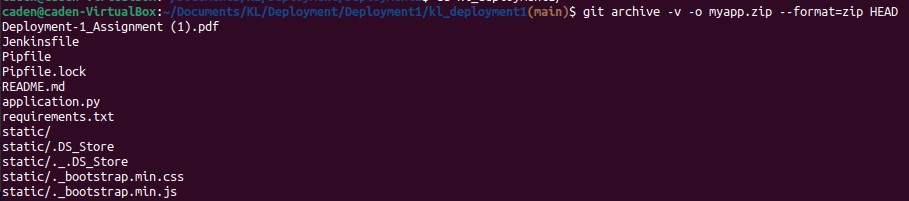
Description automatically generated

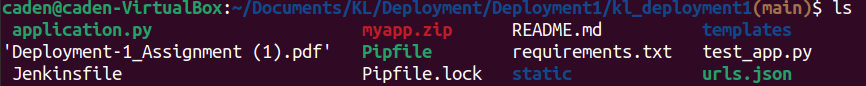
* After applying and saving, clicked into the main branch to confirm that build has occurred successfully:Graphical user interface, application, Teams

  Description automatically generated

## Deploy to Elastic Beanstalk

* Created the myapp.zip file for source bundle: **git archive -v -o myapp.zip --format=zip HEAD**





* Entered all the provided info and uploaded url-shortener-source zipped file, then created the application

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated

* Accessed the application using the URL generated by Elastic Beanstalk:

<http://urlshortener-env.eba-vm7z6jyk.us-east-1.elasticbeanstalk.com/>

Graphical user interface, application

Description automatically generated

## Terminating Elastic Beanstalk and Stopping EC2

* Terminated the environment in Elastic Beanstalk after making sure that it worked:

Graphical user interface, text, application, email

Description automatically generated

* Stopped the EC2 instance created for Jenkins pipeline (kl-deployment1):A screenshot of a computer

  Description automatically generated

## Other Observations

* I did not have an option to create a new environment in Elastic Beanstalk because I did not have any application to begin with – the only option I was able to choose was to create a new application (and the environment was created automatically as a result), so this part of the PDF was a bit confusing
* When I went to stop the EC2 instance created for the Jenkins pipeline, I noticed another EC2 that was created automatically by Elastic Beanstalk (urlshortener-env) – I did not realize that creating a web application using Elastic Beanstalk would prompt it to create an environment inside a brand new EC2 instance!A screenshot of a computer

  Description automatically generated
* To be completely honest – at first, I did not fully understand the difference between what the task was for Jenkins and Elastic Beanstalk. I initially thought we were doing the same thing in two different places, but as I reread my documentation and helped others with their deployment, I realized how wrong my assumptions were:
  + Jenkins was used to build and test the code from our deployment repository - Jenkins needed to connect to GitHub via the token so it can automatically trigger the build and test anytime there was a change in the main branch
  + Elastic Beanstalk was used to actually run the application by creating an environment that the application would live in (i.e. a brand new EC2 instance to host the application) – once the environment setup was completed, AWS provided a URL that can be used to access the fully built app! If I gave that link to other people, they would be able to access that same application on their own machines without a problem!

## Points of Improvements

* I noticed that setting up Jenkins with an EC2 required a lot of manual steps involving providing IAM Access Key, SSH Key Pair, and Access Token – I see this as a definite point of improvement where the security credential configuration step can be shortened or done differently to minimize going back and forth between Jenkins, GitHub, and AWS.
* One observation I made was that there were two separate instances created for this deployment: one for Jenkins and one for Elastic Beanstalk. If we can utilize just one EC2 instance to run both, that would be a lot more efficient and there would be less things to worry about (i.e. terminating/stopping EC2 after deployment is over).

## Pipeline Diagram

* See GitHub repository: <https://github.com/cadenhong/kl_deployment1>