kura labs

Deployment #1 – Creation of CI/CD Pipeline

Author: Ricardo Bonnet

Date: 09-02-2022

Project Description: Two stage pipeline (build & test) creation utilizing Jenkins, GitHub repository deployment utilizing AWS Elastic Beanstalk.

Prerequisites:

- CI/CD Environment (Jenkins)
- AWS Account (EC2 and Elastic Beanstalk)
- GitHub Repository to be deployed

Creation of EC2 Instance

1) Create an Ubuntu EC2 instance and save the .pem file associated with your EC2 to your local machine. This file will be used later to SSH into your EC2 instance.
    - **How to Create an EC2:** https://www.cloudbooklet.com/create-an-ec2-instance-on-aws-with-ubuntu-18-04/
2) Edit inbound rules and open ports 80, 8080, and 22 on your EC2.
3) Create EC2 and wait for instance to initialize and launch.
4) Navigate to .pem file directory in terminal and run the following command:
    - sudo chmod 600 pemfilename.pem
5) SSH into your EC2 instance using the following command:
    - ssh -i pemfilename.pem ubuntu@EC2_Public_IPV4_Address

Installing Jenkins on an EC2

1) After connecting to your EC2 instance, run the following commands in order:
    - sudo apt update && sudo apt upgrade -y
    - sudo apt install default-jre
    - wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key |sudo gpg --dearmor -o/usr/share/keyrings/jenkins.gpg
    - sudo 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'
    - sudo apt update && sudo apt install Jenkins -y
    - sudo systemctl start Jenkins
    - sudo systemctl status Jenkins
        - **How to Install Jenkins Guides:**
            - https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/
            - https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04

Installing Virtual Environment

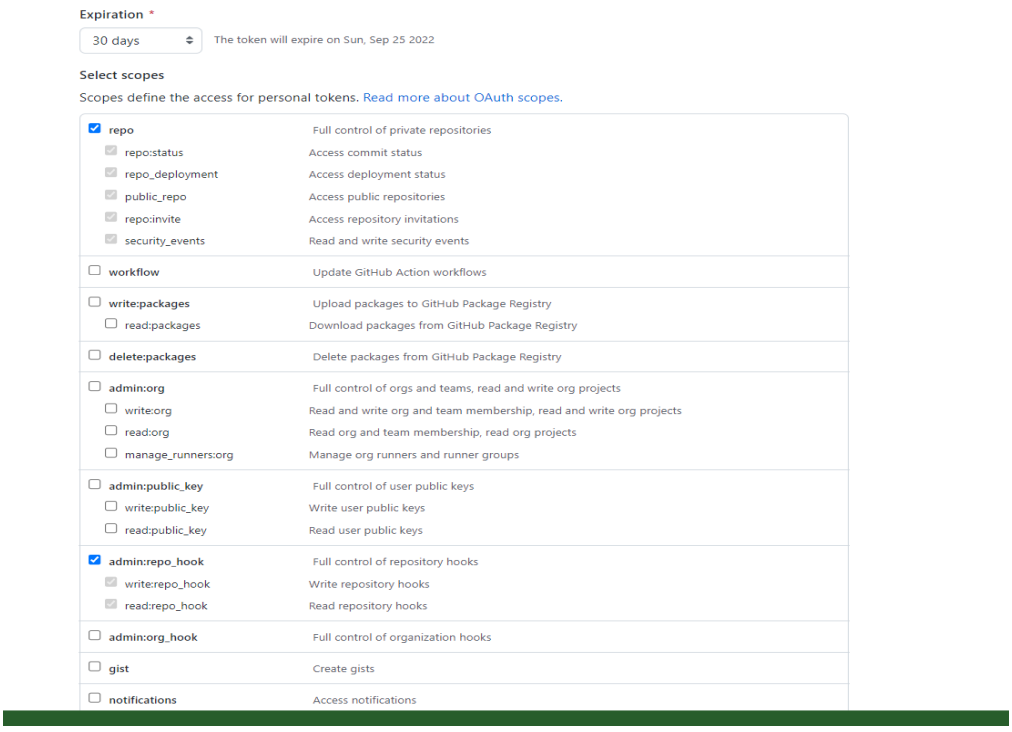
1) First, check if python3 is installed on your EC2 via the following command:
    - Python3 --version
2) If not installed, run the following commands, in order, to install your virtual environment and upgrade to the latest version of python3 and pip:
    - sudo apt install python3-pip
    - sudo pip3 install --upgrade pip
    - sudo apt install python3.10-venv
        - **Guides on Installing Python3, Pip, and Venv:**

- https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/
- https://installati.one/ubuntu/22.04/python3.10-venv/

Connecting GitHub to Jenkins Server

1) Fork the designated deployment repository:

   - https://github.com/rbonnet9/Kura_Labs_Deployment_1
   - Create personalized access token from your GitHub:
     - Navigate to your GitHub Settings → Developer Settings → Personal Access Token → Generate New Token
     - Select settings below when generating token:



- Once token is generated, copy the unique ID that is generated and store for later use.

Setting up Jenkins

1) Connect to http://"EC2_public_IPV4_Address":8080 without the quotes from browser. This directs you to the Jenkins interface.
2) Go into your EC2 terminal and run the following command to reveal your Jenkins password required to unlock the interface:
    - sudo cat ~/var/lib/jenkins/secrets/initialAdminPassword
    - Copy the password displayed and paste into Jenkins interface to continue with your setup.
    - Once unlocked, select "install recommended plugins" and let your environment finish its setup.
3) Create your admin user (username, password, full name, email) when prompted
4) The Jenkins URL provided after creating your user can be used to access your Jenkins interface in the future. It's your EC2 public IPV4 with port 8080.

Creating a Multibranch Build in Jenkins

1) In Jenkins, select "New Item", and select "Multibranch Pipeline"
2) Enter a display name, "URL Shortener" for this deployment.
3) Add a branch source, select "GitHub".
    - Under GitHub credentials, select "Add" and select "Jenkins"
        - Under username, input your GitHub username.
        - Under password, enter your GitHub Token we generated earlier.

- Enter the repository URL into the designated area.
- Make sure the “mode” is set to “Jenkins file”.

4) Once all settings are verified as listed, hit apply and save at the bottom of the page.
5) A build will begin happening, if not select “Scan Repository” on the lefthand side of the page.
    - Two stages will be created, your “Build” and “Test” stages, as designated in the jenkinsfile within your repository.
    - A green box indicates a successful deployment of the designated stages. A red box indicates a failed stage, and any errors can be accessed by hover overing the build stage time and looking at the generated log file for the failed stage.

Downloading Files from GitHub

1) Use git clone (or download directly from GitHub repository page) command to copy repo files of our flask application to your local machine:
    - git clone git@github.com:repositorylink
2) Once on local machine, compress the file into a ZIP. To do so, run the following commands:
    - sudo apt install zip
    - zip file_name_here
3) Now you have a zipped file with all the required files that can be uploaded into an Elastic Beanstalk environment.

---

kura labs

Creation and Deployment of Elastic Beanstalk Env.

1) Head to the AWS Elastic Beanstalk website.
2) Create a new web server environment/app (sometimes it'll prompt one but not the other), and enter the following parameters for your environment:
    - Application name: url-shortener
    - Environment name: Urlshortener-env
    - Platform: Python
    - Platform branch: Python 3.8
    - Platform version: 3.3.16 (or the recommended version)
    - Application code: Select "upload your code" → "local file" → upload the ZIP file we compressed
3) Finish by clicking "Create environment". It can take anywhere from 5-15 minutes for your environment to be created depending on the contents of the ZIP file and its size.
4) Refresh the page, once your environment is finished creating, select the URL provided on the "application health" page of your newly created environment to access the url-shortener from the flask app.

Issues Encountered & Solutions Found

1) Issue #1: Failed to download ZIP file via SCP from EC2

- Initially I had an issue transferring the ZIP file from the EC2. It was nearly an hour of encountering "Permission denied (publickey)" and trying a variety of methods to no avail.
    - **Solution:** I opted for WinSCP and connected my local machine to the EC2 via the EC2s IP, port 22, and login credentials for the EC2. After that it was

as simple as dragging and dropping the ZIP file onto my local machine within WinSCP.

2) Issue #2: Elastic Beanstalk HTTP 502 Error

- The health of the Elastic Beanstalk environment showed as "Severe" after the environment was finalized.
    - **Solution:** When zipping the file, you must only ZIP the files within the parent folder and not the parent folder itself. A simple fix.

Potential Deployment Improvements

1) Improvement #1: Quicker Deployment Utilizing Jenkins

- https://www.serverkaka.com/2019/04/deploy-java-web-app-elastic-beanstalk-jenkins.html
- Combining Elastic Beanstalk with Jenkins will allow for a quicker deployment, without having to download a zip file and upload it to your Elastic Beanstalk environment.
- No need to worry about compatibility and/or reliability issues.
- Ability to configure how you want the deployment to be performed, but everything else is automated.
- https://www.ibexlabs.com/blog/ci-cd-processes-and-tools-for-aws-elastic-beanstalk

2) Improvement #2: Automation of Initial Jenkins Install

- Instead of manually entering in the commands to install Jenkins, we could instead input a script in the initial EC2 setup to automatically run when the EC2 is launched. All that would need to be done after the EC2 is launched is run a "sudo

apt update && sudo apt upgrade -y" command along with "python3 --version" to make sure everything is installed and updated.

- Script would be as follows:

```
#!/bin/bash

# Download JRE
sudo apt update && sudo apt install default-jre -y

# Get the key to access Jenkins package
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo gpg --dearmor -
o/usr/share/keyrings/jenkins.gpg

# Using the key to authenticate the package
sudo 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'

# Update and install Jenkins
sudo apt update && sudo apt install jenkins -y

# Start Jenkins and check the status to make sure it is running
sudo systemctl start jenkins
sudo systemctl status jenkins >> ~/file.txt
```