Ishtaar Desravines
09/27/2022
Deployment 2 Documentation

Objective: Deploying an application to Elastic Beanstalk through the CI/CD pipeline using Jenkins, Github, and AWS services.

Create an EC2 or use an existing EC2:

Creating a new EC2 with Jenkins:

- Log onto AWS as a root user.
- Search for EC2 in the search bar of Console Home.
- Select “Instances” on the left hand side menu, then select “Launch instances” at top-right hand corner.
- Name the instance (ex. Jenkins Server) and select “ubuntu” under “Application and OS images (Amazon Machine Images”
- Select or create a new key pair
- Use an existing security group or create a new one, making sure ports 22, 80, and 8080 are open.
    - Port 22: open to ssh into the remote server (EC2) from my local ubuntu virtual machine.
    - Port 80: to access the web server of Jenkins.
    - Port 8080: to access the Jenkins webpage to create/log into account from browser.
- Under “Advanced details”, add jenkins server script below to the “User data” section to automatically install Jenkins on the EC2 and create an active agent.

```
#!/bin/bash

sudo apt update
sudo apt -y install openjdk-11-jre
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]
https://pkg.jenkins.io/debian-stable binary/ | sudo tee
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get -y install jenkins
sudo systemctl start jenkins
systemctl status jenkins >> ~/file.txt
```

- Launch instance and ssh into the Jenkins EC2.

- Verify jenkins is running on the EC2 with: `$ systemctl status jenkins`

Installing Virtual Environment on the EC2:

- Python must be installed since the application includes python files.
- Installed a virtual environment on the EC2 so that the application can run without version conflicts from their dependencies.
- Follow the commands below:

```
$ sudo apt install python3-pip
$ sudo apt install python3.10-venv
```

Setting Up Jenkins Account

- Open a web browser and enter http://EC2publicipaddress:8080 to access the jenkins webpage.
- Retrieve the access key for Jenkins using below command and enter it in the Jenkins homepage.

```
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

- Select “Install Suggested Plugins” and wait for all plugins to install.
- Create a First Admin User and then click save and continue.
- You now have access to Jenkins GUI.

Activating Jenkins User on EC2

```
$ sudo passwd jenkins # creates jenkins user on EC2 and sets a password.
$ su - jenkins -s /bin/bash #switches to jenkins user and specifies a bash shell
$ exit  #switches the user back to root on the EC2.
```

Activating Jenkins User on AWS

- Search for “IAM” in the search bar in Console Home on AWS.
- Select “Users” in the menu on the left hand side and then “Add user” on the top right hand corner.
- Enter in a username (ex. EB-user)

- Check “Access Key: Programmatic access” under “Select AWS credential type” and then “Next”
- Select “Attach existing policies directly”, check “Administrator Access” , and then “Next”
- Select “Next” for both “Add Tags” and “Review” page then select “Create User”
- Download/Save the access key and secret access key for the user and “Close”

Installing AWS CLI on the Jenkins EC2

- AWS CLI has to be installed as root user on the EC2, not as Jenkins user.
- Install “unzip” if not already on EC2.

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
$ sudo apt install unzip
$ unzip awscliv2.zip
$ sudo ./aws/install
$ aws -version
$ su - jenkins -s /bin/bash
$ aws configure
Enter Access Key
Enter Secret Access Key
Enter region: us-east-1
Enter Output format: json
```

Installing EB CLI in Jenkins User on EC2

```
$ su - jenkins  -s /bin/bash # to switch back to jenkins user
$ pip install awsebcli --upgrade --user # installs elastic beanstalk command
line interface in jenkins user
```

- Check the home directory to see if the .bashrc file exists. If not, create a .bashrc file and add the path to eb to the PATH environment variable in order for the shell to use the eb command.

```
$ nano .bashrc
PATH=$PATH:/var/lib/jenkins/.local/bin # add this in the .bashrc file. Save and
exit.
$ source .bashrc # makes the change to PATH environment variable
$ echo $PATH #verify that the path to eb is added to the PATH environment
variable
$ eb --version # check the version of eb running on the jenkins user
```

Linking Github to Jenkins EC2

- Fork the github repo: https://github.com/kura-labs-org/kuralabs_deployment_2
- In Github account, follow these steps to generate a new personal access token:
    - Github → Settings → Developer Settings → Personal Access Token → Label token "EB-user" → Select Scopes: "repo" and "admin: repo_hook' → Generate New Token.

Create Multibranch Build on Jenkins GUI

- Select "New Item" on left hand side of Dashboard.
- Enter the name of the application: "url-shortener" → select "Multibranch pipeline" → select "OK"
- Enter Display Name: Build Flask
- Enter Description: CI/CD pipeline deployment 2
- Add Branch source: Github and under Github Credentials: select "Add" → Jenkins
- Under "Username": Enter Github Username
- Under "Password": Enter Github personal access token
- Save Github credentials
- Enter the Github repository URL → select "Validate"
- Check to make sure under Build Configuration → Mode , it says "by Jenkinsfile" and under Script it say "Jenkinsfile"
- Select Apply → Save
- Saving this configuration should trigger an automatic SCM checkout, build, then test in Jenkins. All of these stages of the pipeline should pass.

Deploying the Application to Elastic Beanstalk from CLI

- Make sure you're in the jenkins user on the EC2 and change into the directory of the application to initialize the environment.

```
$ sudo su - jenkins -s /bin/bash
$ source .bashrc
$ cd ~/workspace/url-shortener_main
$ eb init # initializes elastic beanstalk in the directory of the application
on the jenkins user and sets default values.
```

- Select: us-east-1 → Enter
- Press Enter
- Select: Python

- Select: the latest version of Python
- Select: No for CodeCommit
- Run the following command:

```
$ eb create  # creates the Elastic Beanstalk environment
```

- Hit enter to accept the defaults for the next 3 prompts.
- REMEMBER THE ENVIRONMENT NAME.
- For Spot Fleet: enter No
- Wait for the environment to be created

Adding Deployment Stage to CI/CD Pipeline in Jenkinsfile

- Add the following stage to the Jenkins file:

```
stage ('Deploy') {
     steps {
          sh '/var/lib/jenkins/.local/bin/eb deploy url-shortener-main-dev'
          }
     }
```

- From the Jenkins GUI, select “Build Now” on the menu on left hand side of the Dashboard.
- This will start another checkout from the SCM (Github), then a build, test, and deploy will be performed on the application.
- All stages should pass.