Kura Labs Deployment Documentation

Author: Mike Major

Kura Labs - Cohort #3

9/27/22

URL Shortener Application Deployment #2

**Installation of Jenkins on Amazon Web Services**

1. First, installing Jenkins on Amazon Web Services was the first step in creating my deployment for URLShortener within Ubuntu. (Note:Although Ubuntu was used in my deployment, other Linux distributions could be used as well.)
   1. Configuring your ports will ensure that you are able to SSH into your instance. It is recommended to use port 22.
      1. (Note:It is recommended that you set your SSH source IP to your local machine or a specific IP to minimize vulnerability in your environment. Having an EC2 instance with your source open in 0.0.0.0\0 is not recommended for production environments due to creating a vulnerability.)
   2. Next, launch your instance and then begin the process of connecting to it via your Linux environment.
   3. When you first connect to your environment, make sure that you update your existing packages and install the update java runtime. (Note:Jenkins requires Java to run properly, so installing the latest version would be very helpful for a successful install.)
   4. Next, pull Jenkins file from the repository via the wget command.
   5. Finally, run the apt update and install Jenkins. After installing Jenkins, ensure the service is started and check the status of the request.

**Configuration of Jenkins**

1. Next, configure the Jenkins application to assist with building and testing out your application. If you are not familiar with [Jenkins](https://www.jenkins.io/) please see the link below.
   1. To connect to Jenkins, use your public DNS alongside port 8080 to connect.
   2. This will take you to the initial screen where you’ll need to unlock Jenkins. (Note: Don’t worry, you can connect, but you’ll need the password. The password location will be in your instance per the file path on the Jenkins screen.)
   3. An admin user will need to be created first after logging into the environment.
   4. To ensure your EC2 instance that was created can connect in this environment you’ll need to install the AWS EC2 plugins. From there connect your AWS credentials to the environment..
   5. After your AWS EC2 credentials have been added, you are able to use EC2 as Jenkins agents.
      1. Jenkins agents allow your controller to distribute workloads as it relates to building and testing your application code.
   6. Once you are done, ensure you have stopped or terminated your instance. This will greatly reduce your chances of incurring additional charges.

**Activate Jenkins User**

1. During the Jenkins installation, the “Jenkins user” is created as a service account. (Note: This allows you to operate the Jenkins functions within your environment. It is recommended that you switch to the user complete.)
   1. In your terminal, sign into the jenkins user and modify the existing password. switch to the Jenkins user by using the “sudo su -s /bin/bash” after modifying the password.

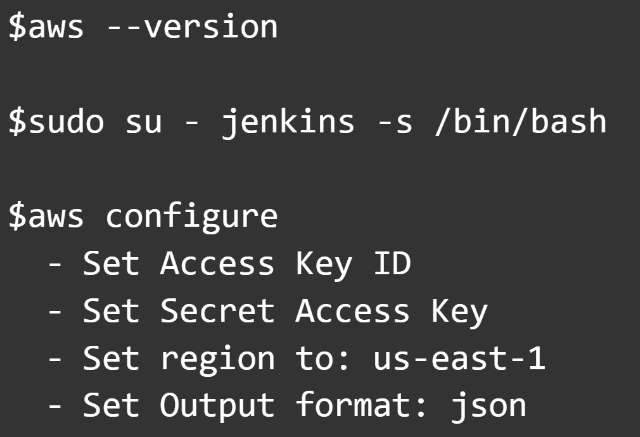
**Create Jenkins User in Amazon Web Services**

1. Next, you’ll create a “Jenkins” user within your AWS console via IAM. IAM will allow you to create users, groups, and roles to interact with your AWS resources in a safe and secure way.
   1. Navigate to IAM (Identity Access Management) within the console.
   2. Select “add user” and assign the username = (EB-user)
   3. Select programmatic access. This option will create an access key ID and a secret access key to be used later. **DO NOT LOSE YOUR ACCESS KEYS. This grants access to your environment via your user.**
   4. Add administrator access to the user and click “attach existing policies directly”.
   5. Next, create the user -- make sure to copy and save both access keys.

**Install AWS CLI in Jenkins EC2**

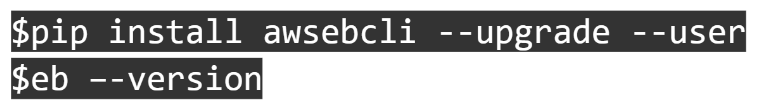
1. Next, the AWS CLI will need to be installed. This will allow the Jenkins user to interact with the AWS resources to be used later in the deployment.
   1. First, use the curl command to download and rename the zip file as you can see below.
   2. Next, unzip the file and install. (Trouble)
   3. Next, check the version to ensure the installation is completed successfully.
   4. Then, after completing the install, switch to your Jenkins user and configure your access with the EB-user credentials from earlier (access key ID and secret access keys)



****

**Install EB CLI in Jenkins EC2**

1. To interact with Elastic Beanstalk within the CLI, you will need to install the EB CLI
   1. Use the pip command below to install the AWS EB CLI. Then, check the version to confirm that it is installed.

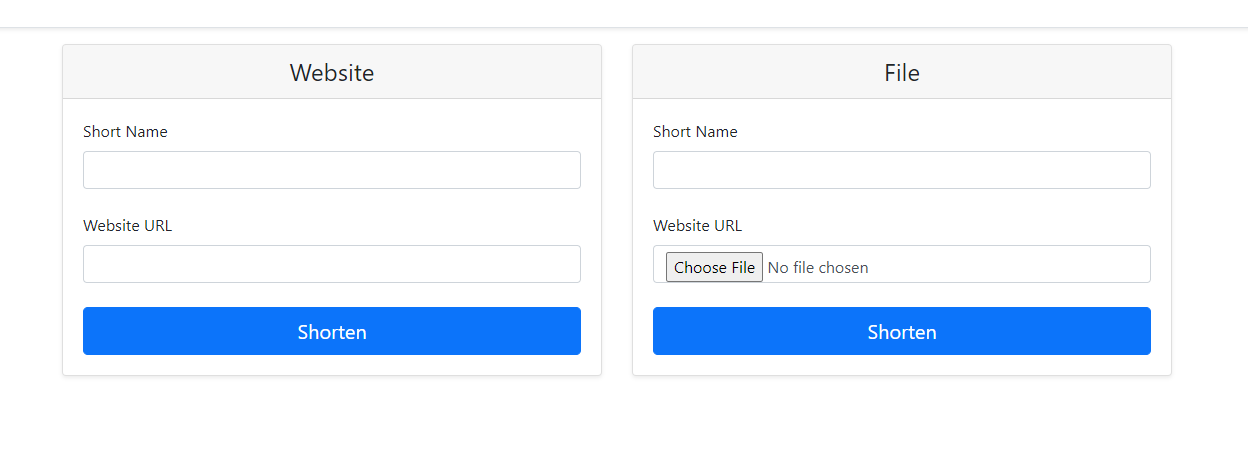
****

**Connecting GitHub to Jenkins**

1. One of the great benefits of using Jenkins is that it can take your code from your repository (in this example, GitHub) to assist in building and testing your application code. This is very helpful in minimizing time in your pipeline within a CI/CD deployment structure.
   1. To connect your GitHub account to Jenkins, you’ll need to establish an access token.
   2. You’ll then need to configure a multibranch build, with the source branch linked to your GitHub repo.
   3. After you’ve linked your repo to the Jenkins application, a build should begin.
   4. Next, your code will be tested and provided with a green indicator if your code passes its checks.

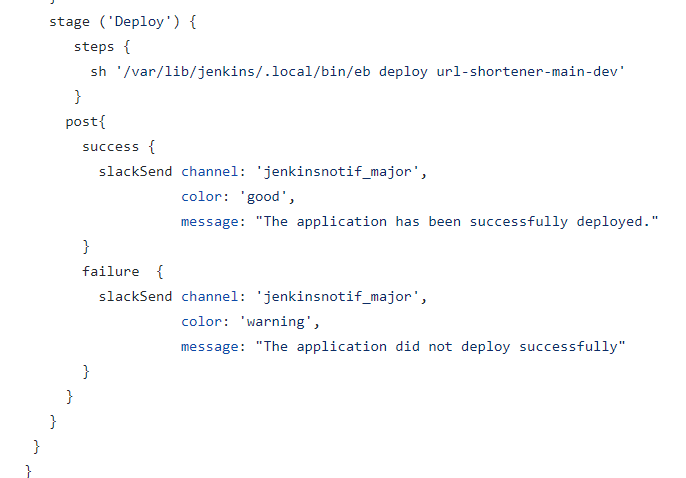
**Deploy Application in Elastic Beanstalk**

1. After completing your build in Jenkins, you are ready to deploy your application within Elastic Beanstalk.
   1. Switch to the Jenkins user.
   2. CD to the workspace location in jenkins. The Jenkins location can be found after clicking on “last build” then the Workspaces option on the left side of the page. This will be your workspace path.
   3. Next, run eb init to configure your EB CLI environment with the specific parameters of your environment.
   4. Lastly, run the eb create command to begin the process of creating your environment and the application.
   5. Once this is complete, you can login into the AWS console, go to the Elastic Beanstalk page and confirm your application has been deployed.



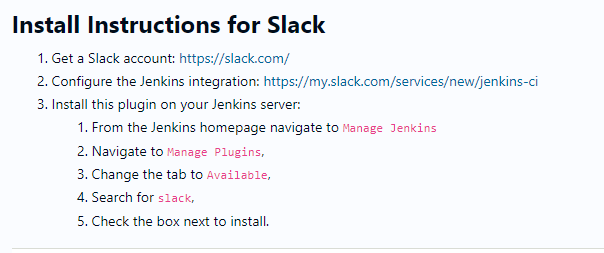
**Modify Jenkins File to Add Deployment Stage**

1. Within your Github repo, go to your Jenkinsfile.
2. Edit your Jenkinsfile to include the Deploy stage of the build process. The environment name (which was previously disclosed -- is all you will need to include in the step to finalize the build.



**Pipeline Modification**

1. For my pipeline modification I chose to use a webhook and notification plugin within Jenkins called Slack Notification. This plugin provides notification integration with Slack or Slack compatible applications.



Areas of Opportunity

1. Create script to automate creation of EC2 instances either with an AMI templates or a script to spin up an EC2 instance with the exact configurations necessary.
2. Utilizing git archive to ensure that the proper files were zipped was an important step in creating my E Beanstalk environment
3. Connecting Elastic Beanstalk and Git via CLI
   1. <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb3-cli-git.html>

Troubleshooting

1. When attempting to install the AWS Elastic Beanstalk CLI, I noticed that I was unable to install the CLI due to the $PATH variable not including the path to of where my files were properly located. The /usr/.local/bin path needed to be appended to the current file by modifying the .bashrc file. After modifying the .bashrc file and logging out and back in as the jenkins user the correct path to run the AWS ELB CLI was correctly set and allowed me to complete the install.
2. After configuring my Jenkins environment and creating my admin username and password, I discovered that I had misplaced my password and could not log in. A solution I was able to find was to disable the security file located in the configuration .xml file. After setting the configuration file from “true” to “false”, then restarting the jenkins -- I was able to login and change my password.