Kura Labs Deployment Documentation
Author: Mike Major
Kura Labs - Cohort #3
9/1/22
URL Shortener Application Deployment

Installation of Jenkins on Amazon Web Services

1. The installation of Jenkins on Amazon Web Services was the first step in creating my deployment for URLShortener. The environment was the Linux distribution, Ubuntu, known for its ease of use, open source and low cost.
    a. Although Ubuntu was used in my deployment, other Linux distributions could be used as well.
    b. Configuring the appropriate ports will ensure that you are able to SSH into your instance. 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.
    c. After the correct ports and distribution of your choice has been configured, launch your instance and then begin the process of connecting to it via your Linux environment. In this example, I used Ubuntu (as previously stated.) as my Linux environment to install Jenkins.
    d. When you first connect to your environment you want to make sure that you update your existing packages and install the update java runtime. Jenkins requires Java to run properly, so installing the latest version would be very helpful for a successful install.
    e. After updating and installing Java to your Ubuntu environment, you'll need to pull the Jenkins file from the repository via the wget command.
    f. Next, 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. After setting up your Ubuntu environment and installing Jenkins, you'll now need to configure the Jenkins application to assist with building and testing out your application. If you are not familiar with Jenkins please see the link below.
    a. To connect to Jenkins, use your public DNS alongside port 8080 to connect. This will take you to the initial screen where you'll need to unlock Jenkins. 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.
        i. The public DNS or the Public IP4 can also be used to connect to your Jenkins applications.

b. An admin user will need to be created first after logging into the environment.
c. 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..
d. After your AWS EC2 credentials have been added, you are able to use EC2 as Jenkins agents.
    i. Jenkins agents allow your controller to distribute workloads as it relates to building and testing your application code.
e. Once you are done, ensure you have stopped or terminated your instance. This will greatly reduce your chances of incurring additional charges.

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.
    a. To connect your GitHub account to Jenkins, you'll need to establish an access token.
    b. You'll then need to configure a multibranch build, with the source branch linked to your GitHub repo.
    c. After you've linked your repo to the Jenkins application, a build should begin.
    d. Next, your code will be tested and provided with a green indicator if your codes passes its checks.

GitHub to Elastic Beanstalk

1. a Elastic Beanstalk will allow you to deploy, manage and scale your web applications and in this case, connecting it to your exported code can make uploading the file very simple.
    a. Make sure to zip all of the necessary files (including the hidden ones) to ensure it is fully zipped.
    b. I utilized the git archive command as it worked within the git environment and ensures that all the necessary files are zipped. If you do not properly zip your files, when you go to upload them to Elastic Beanstalk, it'll give a status of severe for your application.
    c. After zipping the files, log into Elastic Beanstalk, then configure your environment for Python. After that is complete, then upload your file and then upload.
        i. Figure 1 - This is a healthy and successful upload of your application code in Elastic Beanstalk.
        ii. Figure 2 - URL Shortener application fully deployed
        iii. Figure 3 - Severe status for

Figure 1

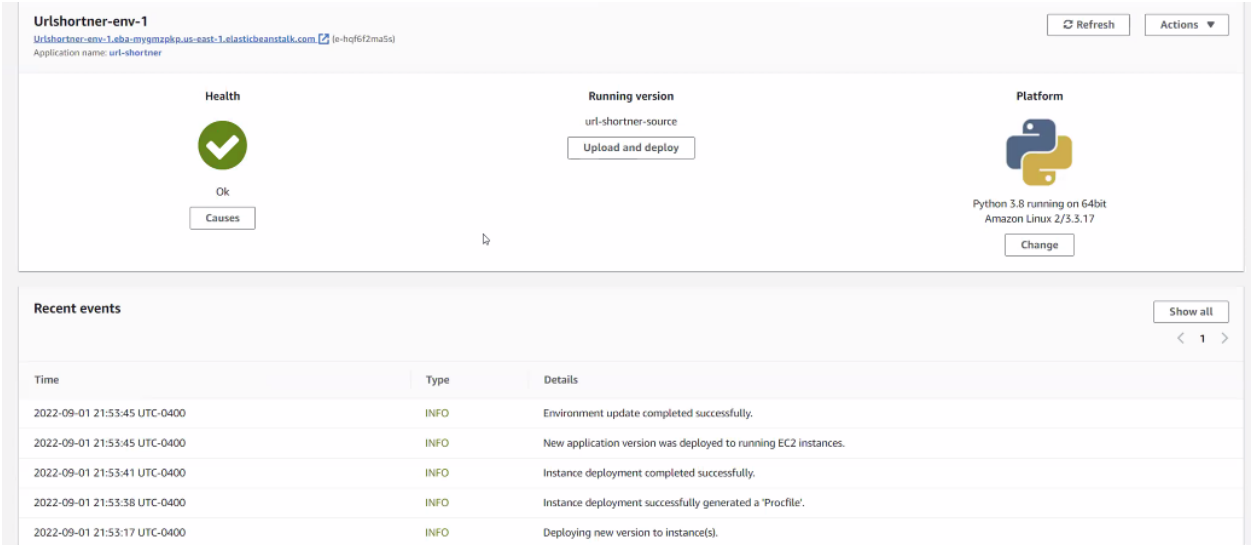


Figure 2

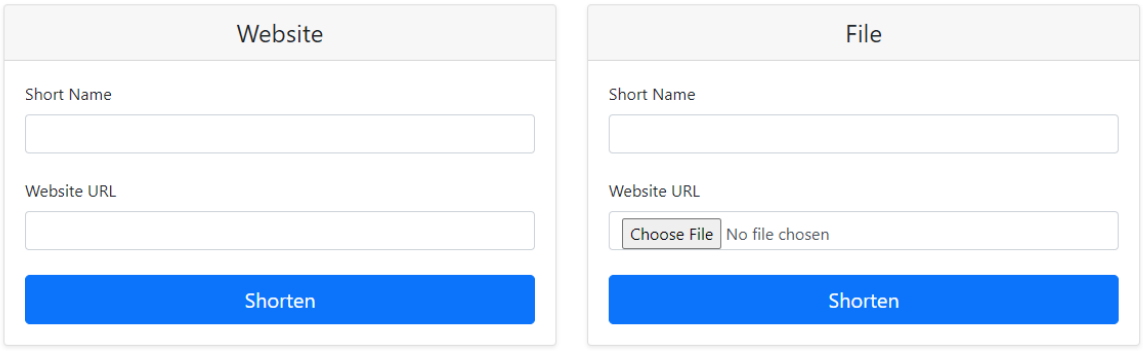
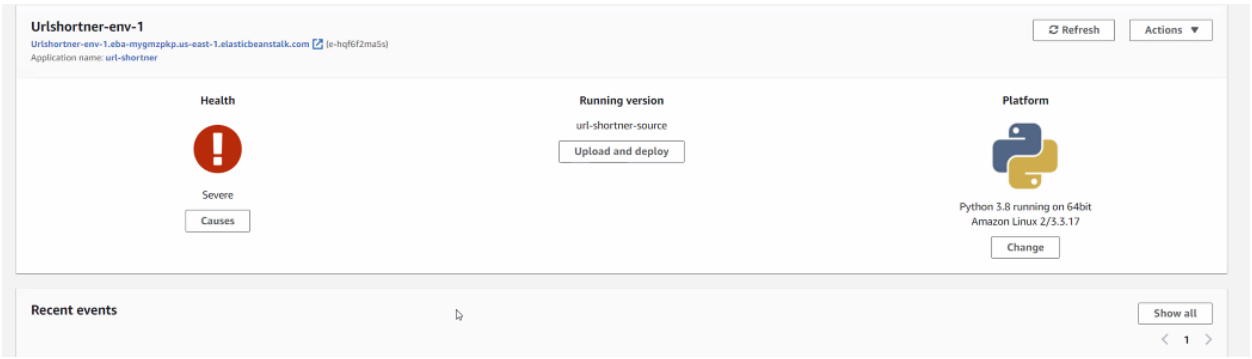


Figure 3



Areas of Opportunity

1. Create cript 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
    a. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb3-cli-git.html