Overview

A DevOps code pipeline helps to automate continuous delivery processes for software development organizations. Developers, testers, and operations staff depend on functioning pipelines to transform code changes into deployable software releases. While software developers are the primary users of a pipeline, IT operations professionals oftentimes have to maintain it.

In this week's assignment, you will build a code deployment pipeline using Jenkins, a continuous integration tool. The pipeline will test, build, and deploy a very simple Java application.

Requirements

You need to have a personal AWS account and GitHub account for this assignment.

The assignment

Let's get that code pipeline flowing!

Fork the Java application

The first step in the assignment is to fork a GitHub repository containing a Java application. Forking the repository will setup a copy of the repository in your personal GitHub account. You can then make changes to the application files in the repository without affecting the original (parent) repository.

Go to GitHub and fork the following repository by clicking on the **Fork** button:

https://github.com/UST-SEIS665/java-project

After a short period of time you should see a forked copy of this repository in your personal GitHub account.

If you already forked this repository during the class lecture, I recommend deleting this repository in GitHub and forking it again to get a clean copy.

Launch the Jenkins stack

You will use a CloudFormation template to launch a stack containing a Jenkins server. The stack will create the following AWS resources:

- An AWS Virtual Private Cloud (VPC), including all the necessary routing tables and routes, an Internet gateway, and security groups for EC2 instances to be launched into.
- An Amazon EC2 instance that hosts a Jenkins server, installed and configured for you with all the necessary plugins.

- An IAM instance role which allows the Jenkins server to access the S3 bucket.

Launch the stack by clicking on the following URL and then clicking the **Next** link:

Launch stack

On the stack **Specify Details** page, enter the following:

1. In **Stack name**, use `jenkins` as the name for the stack.
2. In **JenkinsPassword**, provide an administrative password for this system (choose a hard password).
3. In **JenkinsUsername**, you can leave this set to `admin` or choose a different admin username.
4. In **Key Name**, choose the name of your Amazon EC2 key pair.
5. In **YourIP**, type the public IP address (appending /32) of the computer system which you will access the resources created by this template. If you do not know your current public IP address, you can open up a new web browser window and go to the URL: http://checkip.amazonaws.com/

Click the **Next** button.

On the **Options** page you can tag the resources created by the template. Feel free to create whatever tag you would like and click **Next**.

On the **Review** page, select the **I acknowledge that this template might cause AWS CloudFormation to create IAM resources check box**. (It will.) Review the other settings, and then choose **Create**.

It will take several minutes for CloudFormation to create the resources on your behalf. You can watch the progress messages on the **Events** tab in the console. When the stack has been created, you will see a CREATE_COMPLETE message in the **Status** column of the console and on the **Overview** tab.

You will find the IP address of the Jenkins server in the stack outputs (select the stack name and look at the outputs tab). Access the Jenkins server with a web browser on port 8080:

```
http://<Jenkins IP address>:8080
```

Log into Jenkins using the username and password you provided during the stack creation process. Note that it might take a couple minutes for the Jenkins service to start up and display a login page.

Congratulations! You've created the infrastructure required to build a code pipeline.

Setup GitHub integration

Before you start working on a build pipeline, you should configure Jenkins to integrate with your GitHub account. This will allow Jenkins to automatically pull source code from your forked Java repository when the build pipeline runs.

Here is a summary of the steps required to setup this integration. You can review the class lecture notes or online how-to guides to complete this setup.

- Create a new personal access token in your GitHub account. The token should have `admin:repo_hook`, `repo`, and `repo:status` permissions.
- Add a new GitHub server in the Jenkins system configuration settings.
- Add credentials for the GitHub server configuration.
    - Kind: Secret Text
    - Copy personal GitHub token into secret value field.
- Remember to select the GitHub token credential in the GitHub server settings after adding it!
- Click the Test connection button to verify that the credentials work and save the Jenkins system configuration.

Now Jenkins is setup to communicate with GitHub, but committing changes to your forked repository won't automatically trigger Jenkins builds. Let's fix that.

Go to your Github forked java-project repository and click on the repository settings link. Next, select the **Integrations & services** settings. Add a new integration service called **Jenkins (Github Plugin)**. You will need to configure a Jenkins hook URL for this service. The URL is:

```
http://<your Jenkins server IP>:8080/github-webhook/
```

Okay, now your Jenkins-GitHub integration should be complete. You will be able to verify this integration in the next section when you start to push changes to your java-project repository.

Create a Pipeline

You will create a new build pipeline using the Jenkins Pipeline DSL. Jenkins supports two different types of Pipeline DSL formats: declarative and scripted. We will use the declarative format for this assignment because it's a little easier to work with.

Begin by creating a new pipeline project in Jenkins called **java-pipeline**. Configure the project to reference the Java GitHub repository you forked earlier in the assignment.

Setup a build trigger for this project using the `GitHub hook trigger for GITScm polling` configuration setting. The project should create a pipeline (`Pipeline script from SCM`) based on a file called **Jenkinsfile** located at the root (top) directory of the Java project.

The project pipeline should include the following named stages (highlighted in bold):

- **Unit Tests**

    - The pipeline will initiate unit tests using `ant` and create a junit report.
    - The shell command to run ant tests is `ant -f test.xml -v`
    - The junit report source data is located in `reports/result.xml`
- **Build**
    - The pipeline will compile the Java application using `ant`.
    - The shell command to compile the application is `ant -f build.xml -v`
- **Deploy**
    - The pipeline will copy the build output jar file into an S3 bucket (use an existing S3 bucket or create a new one for this assignment).
    - The name of the output jar file will look something like `rectangle-2.jar`, where the number represents the current Jenkins build number.
    - You can use the AWS CLI to copy files from Jenkins to the S3 bucket. Don't worry about access credentials for this step because the Jenkins server has a proper role attached which allows it to access the S3 bucket.
- **Report**
    - The pipeline will generate a report of the CloudFormation stack resources created in your environment using the command: `aws cloudformation describe-stack-resources --region us-east-1 --stack-name jenkins`
    - Note: you will need to setup proper IAM and Jenkins access credentials to run this command.

Hints

I recommend starting the pipeline by first creating a very basic Jenkinsfile and committing it to your forked repository. Then verify that the pipeline project automatically triggers and tries to create a build. Don't worry if the build initially fails.

Next, begin adding build stages to the Jenkinsfile, committing changes to the GitHub repository as you add each new stage. You should monitor each build in Jenkins by reviewing the console output for each job. Once you are confident that a stage is working, move on to the next stage.

It will likely take a handful of iterations and updates before you complete the pipeline. This is how we build and test pipelines in practice.

Save your work

Create a new GitHub Classroom repository by clicking on this link:
https://classroom.github.com/a/bZFs-qsA

Copy the Jenkinsfile you created for this assignment into this new repository and commit the code. Also, copy the console output from a successful build job into a file called `console.txt`

Check your work

Here is what the contents of your git repository should look like before final submission:

```
├ Jenkinsfile
└ console.txt
```

Terminate application environment

The last step in the assignment is to delete all the AWS resources created by the stack. You don't want to keep this stack running for a long time because the costs will accumulate.

Go to the CloudFormation dashboard, select your running stack, and choose the delete option. Watch as CloudFormation deletes all the resources previously created.

Submitting your assignment

I will review your published work on GitHub after the homework due date.