VERITAS UNIVERSITY, ABUJA

FACULTY OF NATURAL & APPLIED SCIENCES

Computer and Information Technology Department

Rainfall, Second Semester
2022/2023 Academic session

Mr. Enoch O. Daniel
Email: daniele@veritas.edu.ng

CSC 314 – CLOUD COMPUTING

LAB 3: CAPSTONE PROJECT

CONTINUOUS INTEGRATION AND DEPLOYMENT PIPELINE WITH AWS

SUBMISSION DUE DATE: MONDAY 28th May, 2024

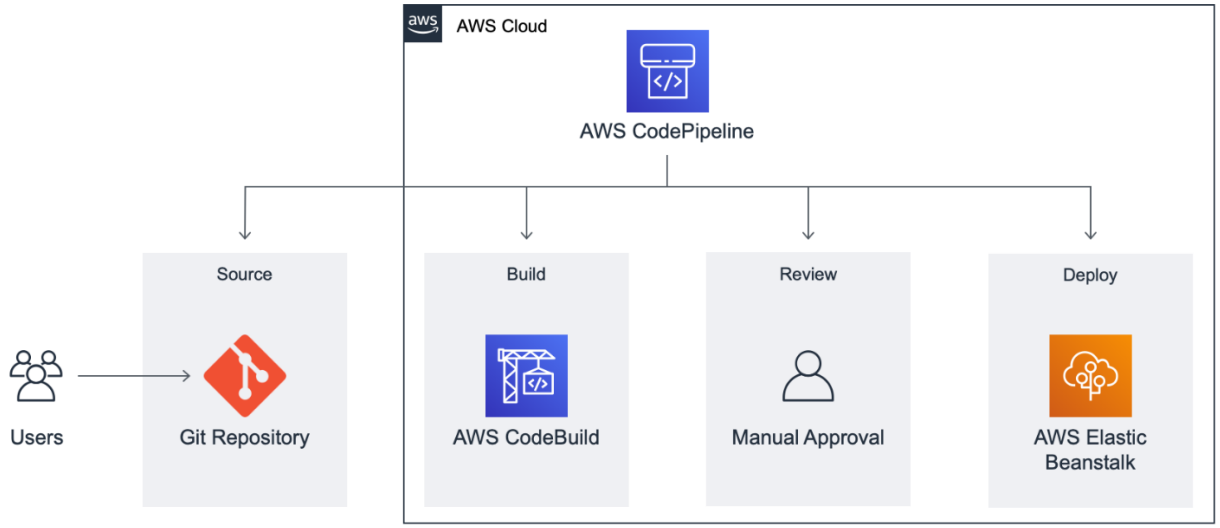
PS: skip to STEP 3 if you were in class for the initial STEPs (confirm to see if you have success note from the server configuration)

CI-CD PROJECT AWS CLOUD ARCHITECTURE

The following diagram provides a visual representation of the services used in this lab and how they are connected. This application uses

GitHub, AWS Elastic Beanstalk, AWS CodeBuild, and AWS CodePipeline.

As you go through this lab, these services will be discussed in detail and point to resources that will help you get up to speed with them.



STEP 1

In this step, you will set up a repository for your code so it can be easily accessed over the Internet. In this example, we will be using **GitHub**, but there are other Git-compatible options you can use, including **AWS CodeCommit**. In one of the following steps, you will connect this hosted repo to our pipeline, so every time you push a new commit to it your build process is started.

What you will accomplish

In this step, you will:

- Fork a GitHub repository to create a new one
- Store code and metadata in GitHub
- Interact with a code repository using Git

Key concepts

1. **Version control**—A system for storing source code and tracking any changes to it. Changes are stored as versions, so a developer can easily compare versions or choose to revert to an older version.
2. **Git—**An open-source version control tool for managing changes to source code.
3. **Git repository (repo**)—All the files and directories that will be tracked by the version control system, including metadata. Each user will interact with a complete copy locally and push files to the hosted version to share changes.
4. **Git commit**—The method to add changes to a Git repository.
5. **Pushing to a repository**—Copying any changes stored via a commit from a local repository to a hosted one.
6. **Forking a repository**—Creating a copy of an existing repository.

Time to complete

Step prerequisites

- **GitHub account**
- **Git installed on your computer**
- **A text editor.**
    - Visual Studio Code

STEP 1 A: FORK THE REPOSITORY

This lab assumes you have an existing **GitHub account and Git cli** installed on your computer. If you don't have either of these two installed, you can follow these step-by-step instructions.

1. Open a folder in your Vscode named CSC314-CICD
2. In a new browser tab, navigate to GitHub and make sure you are logged into your account.
3. In that same tab, open **https://github.com/veritas-uni-abj/VUNA-CSC-CLOUD-CICD-PIPELINE.git** repo.
4. Choose the white Fork button on the top right corner of the screen. Next, you will see a small window asking you where you would like to fork the repo. Name it **CSC314-CLOUD-CICD-YOURNAME**
5. Verify it is showing your account and choose Create a fork. After a few seconds, your browser will display a copy of the repo in your account under Repositories.

STEP 2

Deploy Web App

In this step, you will create and deploy a web application using AWS Elastic Beanstalk

Overview

In this step, you will use the **AWS Elastic Beanstalk (EBS)** console to create and deploy a web application. ***AWS Elastic Beanstalk is a compute service that makes it easy to deploy and manage applications on AWS without having to worry about the infrastructure that runs them.*** You will use the Create web app wizard to create an application and launch an environment with the AWS resources needed to run your application. In subsequent steps, you will be using this environment and your continuous delivery pipeline to deploy the Hello World! web app created in step 1.

Objectives

you will:

i. Configure and create an AWS Elastic Beanstalk environment

ii. Deploy a sample web app to AWS Elastic Beanstalk

iii. Test the sample web app

Key concepts

i. **AWS Elastic Beanstalk** - A service that makes it easy to deploy your application on AWS. You simply upload your code and Elastic Beanstalk deploys, manages, and scales your application.
ii. **Environment** - Collection of AWS resources provisioned by Elastic Beanstalk that are used to run your application.
iii. **EC2 instance** - Virtual server in the cloud. Elastic Beanstalk will provision one or more Amazon EC2 instances when creating an environment.
iv. **Web server** - Software that uses the HTTP protocol to serve content over the Internet. It is used to store, process, and deliver web pages.
v. **Platform**—Combination of operating system, programming language runtime, web server, application server, and Elastic Beanstalk components. Your application runs using the components provided by a platform.

STEP 2 A; CONFIGURING EBS

1. In the AWS CONSOLE search and open the **AWS Elastic Beanstalk console.**
2. Choose the orange Create Application button.
3. Choose Web server environment under the Configure environment heading.
4. In the text box under the heading Application name, enter **CSC314-CLOUD-SERVER-CICD-YOURNAME.**
5. In the Platform dropdown menu, under the Platform heading, select **Node.js.** Platform branch and Platform version will automatically populate with default selections.
6. Confirm that the radio button next to Sample application under the Application code heading is selected.
7. Confirm that the radio button next to Single instance (free tier eligible) under the Presets heading is selected.
8. Select **Next.**

9. On **the Configure service** access screen, choose Use an existing service role for Service Role.
10. For EC2 instance profile dropdown list, the values displayed in this dropdown list may vary, depending on whether you account has previously created a new environment.

11. Choose one of the following, based on the values displayed in your list.

12. If ***aws-elasticbeanstalk-ec2-role*** displays in the dropdown list, select it from the EC2 instance profile dropdown list.

13. If another value displays in the list, and it's the default EC2 instance profile intended for your environments, select it from the **EC2 instance profile dropdown list.**

14. If the EC2 instance profile dropdown list doesn't list any values to choose from, expand the procedure that follows, ***Create IAM Role for EC2 instance profile*** **on the IAM page**

    - Complete the steps in **Create IAM Role for EC2 instance profile** to create an IAM Role that you can subsequently select for the EC2 instance profile attach these policies
    - . Then, return back to this step.
    - Now that you've created an **IAM Role, and refreshed the list, i**t displays as a choice in the dropdown list. **Select the IAM Role you just created from the EC2 instance profile dropdown list.**

- Keep pressing NEXT or Choose Skip to Review on the Configure service access page.

STEP 3

Create Build Project

In this step, you will configure and execute the application build process using AWS CodeBuild

Overview

In this step, you will use AWS CodeBuild to build the source code previously stored in your GitHub repository. ***AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy.***

Objectives

In this step, you will:

i. Create a build project with AWS CodeBuild
ii. Set up GitHub as the source provider for a build project
iii. Run a build on AWS CodeBuild

Key concepts

i. **Build process**—Process that converts source code files into an executable software artifact. It may include the following steps: compiling source code, running tests, and packaging software for deployment.
ii. **Continuous integration**—Software development practice of regularly pushing changes to a hosted repository, after which automated builds and tests are run.
iii. **Build environment**—Represents a combination of the operating system, programming language runtime, and tools that CodeBuild uses to run a build.
iv. **Buildspec**—Collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build.
v. **Build Project**—Includes information about how to run a build, including where to get the source code, which build environment to use, which build commands to run, and where to store the build output.
vi. **OAuth**—Open protocol for secure authorization. OAuth enables you to connect your GitHub account to third-party applications, including AWS CodeBuild.

STEP 3 A: CONFIGURING AWS BUILDCODE PROJECT

1. In your AWS console search and, open the AWS Code Build console in a new tab.
2. Choose the orange Create project button.
3. In the Project name field, enter **CSC314-CLOUD-BUILD-CICD-YOURNAME**
4. Select **GitHub** from the Source provider dropdown menu.
5. **Confirm that the Connect using OAuth radio button is selected**.
6. Choose the white Connect to GitHub button. A new browser tab will open asking you to give AWS CodeBuild access to your GitHub repo.
7. Choose the green Authorize **aws-codesuite button.**
8. Enter your GitHub password or use the github App on your phone to authenticate
9. Choose the orange Confirm button.
10. Select Repository in my GitHub account.
11. Enter enter the **EBS project name** we created in step 1 above in the search field.
12. Select the **repo you forked** in Step 1. After selecting your repo, your screen should look like this:
13. Confirm that Managed Image is selected.
14. Select Amazon Linux 2 from the Operating system dropdown menu.
15. Select Standard from the Runtime(s) dropdown menu.
16. Select **aws/codebuild/amazonlinux2-x86_64-standard:3.0** from the Image dropdown menu.
17. Confirm that Always use the latest image for this runtime version is selected for Image version.
18. Confirm that Linux is selected for Environment type.
19. Confirm that New service role is selected

STEP 3 B: CREATING A BUILDSPEC

1. Select Insert build commands.
2. Choose Switch to editor.
3. Replace the Buildspec in the editor with the code below (i.e. COPY AND PASTE)

```
version: 0.2
phases:
    build:
        commands:
            - npm i --save
artifacts:
    files:
```

STEP 3 C: TESTING THE CODEBUILD PROJECT

1. Choose the orange Start build button. This will load a page to configure the build process.
2. Confirm that the loaded page references the correct GitHub repo.
3. Choose the orange Start build button.
4. Wait for the build to complete. As you are waiting you should see a green bar at the top of the page with the message *Build started,* the progress for your build under Build log, and, after a couple minutes, a green checkmark and a *Succeeded* message confirming the build worked.

STEP 4

Create Delivery Pipeline

In this step you will use AWS CodePipeline to set up a continuous delivery pipeline with source, build, and deploy stages

Overview

In this step, you will use **AWS CodePipeline** to set up a continuous delivery pipeline with source, build, and deploy stages. The pipeline will detect changes in the code stored in your GitHub repository, build the source code using AWS CodeBuild, and then deploy your application to AWS Elastic Beanstalk.

Objectives

In this step, you will:

i. Set up a continuous delivery pipeline on AWS CodePipeline
ii. Configure a source stage using your GitHub repo
iii. Configure a build stage using AWS CodeBuild
iv. Configure a deploy stage using your AWS ElasticBeanstalk application
v. Deploy the application hosted on GitHub to Elastic Beanstalk through a pipeline

Key concepts

i. **Continuous delivery**—Software development practice that allows developers to release software more quickly by automating the build, test, and deploy processes.
ii. **Pipeline**—Workflow model that describes how software changes go through the release process. Each pipeline is made up of a series of stages.
iii. **Stage**—Logical division of a pipeline, where actions are performed. A stage might be a build stage, where the source code is built and tests are run. It can also be a deployment stage, where code is deployed to runtime environments.
iv. **Action**—Set of tasks performed in a stage of the pipeline. For example, a source action can start a pipeline when source code is updated, and a deploy action can deploy code to a compute service like AWS Elastic Beanstalk.

STEP 4 A; CREATE A NEW PIPELINE

1. In a browser tab, open the **AWS CodePipeline** console from AWS search console.
2. Choose the orange Create pipeline button. A new screen will open up so you can set up the pipeline.
3. In the Pipeline name field, enter **CSC314-CLOUD-PIPELINE-CICD-YOURNAME**.
4. Confirm that **New service role** is selected.
5. Choose the orange Next button

STEP 4 B: CONNECTING TO VCS

1. Select **GitHub version 1** from the Source provider dropdown menu.
2. Choose the **white Connect to GitHub button**. A new browser tab will open asking you to give AWS CodePipeline access to your GitHub repo.
3. Choose the green **Authorize aws-codesuite button**. Next, you will see a green box with the message *You have successfully configured the action with the provider*.
4. From the Repository dropdown, **select the repo you created in STEP 1**.
5. Select main from the branch dropdown menu.
6. Confirm that **GitHub webhooks is selected**.
7. Choose the orange Next button.

STEP 4 C: CONFIGURE BUILD STAGE

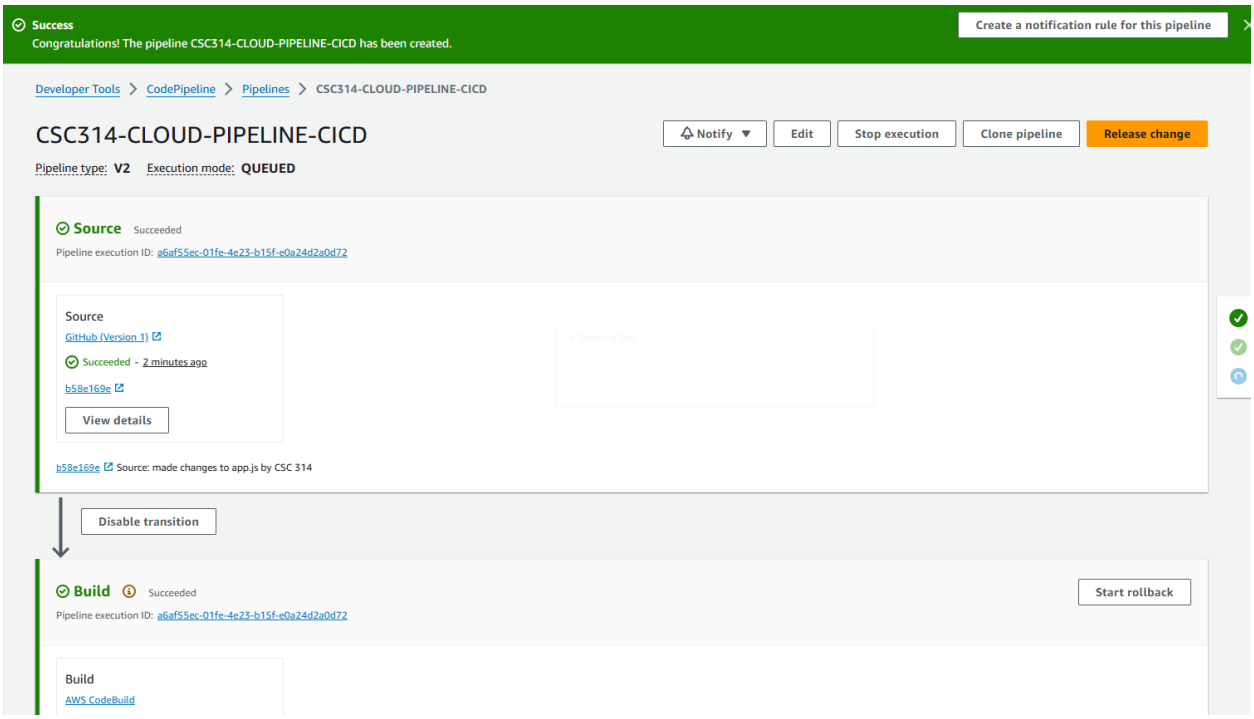
1. From the Build provider dropdown menu, select **AWS CodeBuild.**
2. Under Region confirm that the **US North Virgina** Region is selected.
3. **Select Build from the previous STEP 3** under Project name.
4. Choose the orange Next button.

STEP 4 D: CONFIGURE DEPLOY

1. Select **AWS Elastic Beanstalk** from the Deploy provider dropdown menu.
2. Under Region, confirm that the **US -North virginia** (ensure this your region otherwise use yours) Region is selected.
3. Select the field under Application name and confirm you can see the app created in Step 2.
4. Select the **-env** environment also form step 1 from the Environment name textbox.

5. Choose the orange Next button. You will now see a page where you can review the pipeline configuration.
6. Choose the orange Create pipeline button.

EXPECTED SUCCESS PAGES DURING THE BUILD PROCESS



STEP 5: DEPLOYED CICD PIPELINE FINALIZE PIPELINE AND TEST

In this step, you will add a review stage to your continuous delivery pipeline using **AWS CodePipeline**

Overview

In this step, you will use **AWS CodePipeline** to add a review stage to your continuous delivery pipeline. As part of this process, you can add an approval action to a stage at the point where you want the pipeline execution to stop so someone can manually approve or reject the action. Manual approvals are useful to have someone else review a change before deployment. If the action is approved, the pipeline execution resumes. If the action is rejected—or if no one approves or rejects the action within seven days—the result is the same as the action failing, and the pipeline execution does not continue.

Objectives

In this step, you will:

1. Add a review stage to your pipeline
2. Manually approve a change before it is deployed

Key concepts

1. **Approval action**—Type of pipeline action that stops the pipeline execution until someone approves or rejects it.
2. **Pipeline execution**—Set of changes, such as a merged commit, released by a pipeline. Pipeline executions traverse the pipeline stages in order. Each pipeline stage can only process one execution at a time. To do this, a stage is locked while it processes an execution.
3. **Failed execution**—If an execution fails, it stops and does not completely traverse the pipeline. The pipeline status changes to *Failed* and the stage that was processing the execution is unlocked. A failed execution can be retried or replaced by a more recent execution.

STEP 5 A: CREATE REVIEW STAGE IN PIPELINE

1. Open the AWS CodePipeline console.
2. You should see the pipeline we created in Step 4, which was called **CSC314-CLOUD-PIPELINE-CICD-YOURNAME**.
3. . Select this pipeline.
4. Choose the **white Edit button near the top of the page**.
5. Choose the white Add stage button between the **Build and Deploy stages.**
6. In the Stage name field, enter **CSC314-CLOUD-MANUALREV-CICD.**
7. Choose the orange Add stage button.
8. In the Review stage, choose the white Add action group button.
9. Under Action name, enter **CSC314-CLOUD-MANREVIWER-CICD**
10. From the Action provider dropdown, **select Manual approval**.
11. Confirm that the optional fields have been left blank.
12. Choose the orange Done button.
13. Choose the orange Save button at the top of the page.
14. Choose the orange Save button to confirm the changes. You will now see your pipeline with four stages: **Source, Build, Review, and Deploy.**

STEP 5 A: PUSH A NEW COMMIT

1. In your favorite VScode editor, open the app.js file from step 1.
2. Change the message in your app.js Node JS application. With this

   THIS IS CSC 314 CLOUD COMPUTING DEVOPS; CI-CD PIPELINE CREATED WITH AWS. THIS PAGE INDICATES SUCCESS.

   YOURNAME ----------

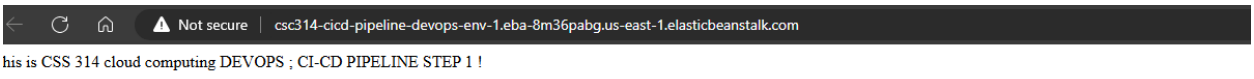
   MATRIC NUMBER ----------

3. Save the file.
4. Open your terminal in the VScode.
5. Navigate to the folder created in STEP 1.
6. Commit the change with the following commands
7. finally, **git push**

STEP 5B : MONITORING

1. Navigate to the AWS CodePipeline console.
2. Select the pipeline named **CSC314-CLOUD-CICD-YOURNAME** You should see the Source and Build stages switch from blue to green.
3. When the Review stage switches to blue, choose the white Review button.
4. Write an approval comment in the Comments textbox.
5. Choose the orange Approve button.
6. Wait for the Review and Deploy stages to switch to green.
7. Select the AWS Elastic Beanstalk link in the Deploy stage. A new tab listing your Elastic Beanstalk environments will open.
8. Select the URL in the **CSC314-CLOUD-MANREVIWER-CICD -env** row. You should see a webpage with a white background and the text you had in your most recent GitHub commit.

Congratulations! You have a fully functional continuous delivery pipeline hosted on AWS.. You should see the image below



SUMMARY

We have created a continuous delivery pipeline on AWS CodePipeline with three stages: source, build, and deploy. The source code from the GitHub repo created in Step 1 is part of the source stage. That source code is then built by AWS CodeBuild in the build stage. Finally, the built code is deployed to the AWS Elastic Beanstalk environment created in Step 3