St. Francis Institute of Technology, Mumbai-400 103
Department of Information Technology

A.Y. 2023-2024
Class: TE-ITA/B, Semester: V

Subject: **Advanced DevOps Lab**

Experiment – 4: To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

1. **Aim:** To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.
2. **Objectives:** After study of this experiment, the student will be able to
   - Build an application using AWS CodeBuild and Deploy on S3 using CodePipeline.
   - Deploy an application on EC2 using AWS CodeDeploy.
3. **Outcomes:** After study of this experiment, the student will be able to
   - Build an application using AWS CodeBuild and Deploy on S3 using CodePipeline.
   - Deploy an application on EC2 using AWS CodeDeploy.
4. **Prerequisite:** Fundamentals of cloud computing
5. **Requirements:** PC and Internet
6. **Pre-Experiment Exercise:**

Brief Theory:

AWS CodePipeline:

AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously.

CICD:

odePipeline is a *continuous delivery* service that automates the building, testing, and deployment of your software into production.

Continuous delivery is a software development methodology where the release process is automated. Every software change is automatically built, tested, and deployed to production. Before the final push to production, a person, an automated test, or a business rule decides when the final push should occur. Although every successful software change can be immediately released to production with continuous delivery, not all changes need to be released right away.

Continuous integration is a software development practice where members of a team use a version control system and frequently integrate their work to the same location, such as a main branch. Each change is built and verified to detect integration errors as quickly as possible. Continuous integration is focused on automatically building and testing code, as compared to *continuous delivery*, which automates the entire software release process up to production.

Features:
You can use CodePipeline to help you automatically build, test, and deploy your applications in the cloud. Specifically, you can:

1. **Automate your release processes**: CodePipeline fully automates your release process from end to end, starting from your source repository through build, test, and deployment. You can prevent changes from moving through a pipeline by including a manual approval action in any stage except a Source stage. You can release when you want, in the way you want, on the systems of your choice, across one instance or multiple instances.
2. **Establish a consistent release process**: Define a consistent set of steps for every code change. CodePipeline runs each stage of your release according to your criteria.
3. **Speed up delivery while improving quality**: You can automate your release process to allow your developers to test and release code incrementally and speed up the release of new features to your customers.
4. **Use your favorite tools**: You can incorporate your existing source, build, and deployment tools into your pipeline. For a full list of AWS services and third-party tools currently supported by CodePipeline..
5. **View progress at a glance**: You can review real-time status of your pipelines, check the details of any alerts, retry failed actions, view details about the source revisions used in the latest pipeline execution in each stage, and manually rerun any pipeline.
6. **View pipeline history details**: You can view details about executions of a pipeline, including start and end times, run duration, and execution IDs.

You will create the pipeline using AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change. You will use your GitHub account, an Amazon

Simple Storage Service (S3) bucket, or an AWS CodeCommit repository as the source location for the sample app's code. You will also use AWS Elastic Beanstalk as the deployment target for the sample app. Your completed pipeline will be able to detect changes made to the source repository containing the sample app and then automatically update your live sample app.

7. Laboratory Exercise

1) To simplify the process of setting up and configuring EC2 instances, you will spin up a sample environment using AWS Elastic Beanstalk. Elastic Beanstalk lets you easily host web applications without needing to launch, configure, or operate virtual servers on your own. It automatically provisions and operates the infrastructure (e.g. virtual servers, load balancers, etc.) and provides the application stack (e.g. OS, language and framework, web and application server, etc.) for you. Your continuous deployment pipeline will need a target environment containing virtual servers, or Amazon EC2 instances, where it will deploy sample code. You will prepare this environment before

creating the pipeline.

Step1: Create a deployment environment

Step 2: Get the copy of Sample Code

In this step you will retrieve the sample app's code and choose a source to host the code.

Pipeline takes the code and performs actions on it.

You can use three options.

- GitHub Repository
- Amazon S3 Bucket
- AWS CodeCommit Repository

Open Amazon S3 console and Create your S3 Bucket:

Create Bucket

Upload the code to Bucket

Select Bucket and Click on Upload (Right Corner)

Add Files --□ upload zip file from downloads of your computer. Click on Upload Button

Step 3: Create Pipeline

In this step, you will create and configure a simple pipeline with two actions: source and deploy. You will provide CodePipeline with the locations of your source repository and deployment environment.

Create new pipeline: Source Provider = Amazon S3, Bucket = VaishaliBucket1, S3 Object Key = Copy S3 Uri from Amazon S3 bucket.

Add Deploy Stage:

Review the settings and create the pipeline.

Add deploy stage and Give details : Deployement provider : AWS Elastic Beanstalk, Region,Application Name and Environment Name…(Auto suggested.)

Pipeline created successfully.

After your pipeline is created, the pipeline status page appears and the pipeline automatically starts to run. You can view progress as well as success and failure messages as the pipeline perform each action.

To verify your pipeline ran successfully, monitor the progress of the pipeline as it moves through each stage. The status of each stage will change from No executions yet to In Progress, and then to either Succeeded or Failed. The pipeline should complete the first run within a few minutes.

Now go to your EBS environment and click on the URL to view the sample website you deployed.

Step 5: Commit a change and then update your app

In this step, you will revise the sample code and commit the change to your repository.

CodePipeline will detect your updated sample code and then automatically initiate deploying it to

1.Visit your own copy of the repository that you forked in GitHub.

- Open index.html
- Select edit icon

2.Update the webpage by copying and pasting the following text on line 30.

3.Commit the change to your repository.

4.Return to your pipeline in the CodePipeline console. In a few minutes, you should see the

Source change to blue, indicating that the pipeline has detected the changes you made to your source repository. Once this occurs, it will automatically move the updated code to Elastic Beanstalk.

- After the pipeline status displays Succeeded, in the status area for the Beta stage, click AWS Elastic Beanstalk.

5. The AWS Elastic Beanstalk console opens with the details of the deployment. Select the environment you created earlier. And click the URL again from EBS environment again.

Step 6: Clean up the resources

To avoid future charges, you will delete all the resources you launched which includes the pipeline, the Elastic Beanstalk application, and the source you set up to host the code.

First, you will delete your pipeline:

- In the pipeline view, click Edit.
- Click Delete.
- Type in the name of your pipeline and click Delete.

Second, delete your Elastic Beanstalk application:

- Visit the Elastic Beanstalk console.
- Click Actions.
- Then click Terminate Environment.
- We have successfully created an automated software release pipeline using AWS CodePipeline! Using CodePipeline, We created a pipeline that uses GitHub, Amazon S3, or AWS CodeCommit as the source location for application code and then deploys the code to an Amazon EC2 instance managed by AWS Elastic Beanstalk. Pipeline will automatically deploy your code every time there is a code change.

8. Post-Experiments Exercise

A. Extended Theory:

- Deploy Web Application using Elastic BeanStalk
- AWS Storage Service: S3

B. Questions:

Q.1 Does AWS Elastic Beanstalk store anything in Amazon S3?

Q. 2 What database solutions can we use with AWS Elastic Beanstalk?

C. Conclusion:

Write the significance of the topic studied in the experiment.

D. References:

https://aws.amazon.com/elasticbeanstalk/faqs/#

https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingBucket.html

---