**DEPLOY**

**APPLICATION ON CLOUD**

**PROJECT**

Prepared by: PALLAVI THOKALA

**Project Objective :**

As a developer, deploy your Spring Boot application on Cloud.

**Use The Following :**

* Cloud: AWS

You have been assigned a new task during the sprint planning. You have been asked to deploy your Spring Boot application on the Cloud. You need to create and launch an instance, provide all the required privileges, and run your application.

**Following Requirements Should Be Met :**

* A few of the source codes should be tracked on GitHub repositories. You need to document the tracked files which are ignored during the final push to the GitHub repository.
* The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository in the document.
* The step-by-step process involved in completing this task should be documented

To configure AWS CodePipeline to deploy a spring boot application from GitHub to an AWS Elastic Beanstalk instance. First of all, let’s get to know what is the automation of code deployment. It is simply called as CI / CD i.e. Continuous Integration and Continuous Delivery and Deployment. As nowadays the Agile approach is used, features have to be delivered in every 2 to 3 weeks. That will make the client happy as well as everything would get sorted In those instances you will have to deploy your code to the server every now and then. It is a difficult task. Then it comes the advantage of using ci/cd . You simply have to commit your changes and the ci/cd configuration will automatically test your code and deploy it to the server .

Now let’s move on to the example I am going to describe. In this example, source code is in GitHub and once we commit a change, it will trigger a hook to AWS infrastructure. Then the build is going to be done using AWS CodeBuild. Build is defined using buildspec.yml which I will show later. After source code is built, artifacts are uploaded to an S3 bucket.

There is a feature called AWS CodeDeploy which deploys the artifacts in the S3 bucket to the Elastic BeanStalk instance. This whole process is controlled using a service called AWS CodePipelines. Now I am going to use my initial project, which I mentioned earlier. In addition to that, I am going to create a GitHub repository and commit and push my code into it.

I had not added the builtspec.yml which I mentioned earlier. Thus right now I am going to add that to the root of my project. This file will be the build configuration for AWS CodeBuild.

In this example, I am going to deploy an exploded WAR file. Thus Exploded WAR plugin needs to be added to the pom.xml file.

What is done by this plugin is that the artifacts will be exploded into a folder called deploy. Simply the WAR file will be exploded into the deploy folder.version represents the build spec standard being used in the configuration. phases represent the stages that we want to instruct CodeBuild to run commands. phases can be installed, pre\_build, build, and post\_build. Here I have only used pre\_build, build, and post\_build. In the pre\_build phase, I have only added a simple log.

**mvn clean prepare-package war:exploded**

This will explode the WAR file into the deploy package as I have added the plugin previously. In my post\_build phase, I have added a command to move the deploy folder from target directory to root.

# target/deploy ./

artifacts represent a set of build output artifacts that CodeBuild uploads to the output bucket. In this case, artifacts will be in the deploy folder thus I state here. Ok, now it’s done. Now let’s go to the AWS console and link this GitHub repo to CodePipeline.

Search for CodePipeline.