**Lesson 02 Demo 04**

**Implementing Blue/Green Deployment Using CodeDeploy**

**Objective:** To implement a blue/green deployment strategy using AWS CodeDeploy and gain proficiency in setting up a continuous integration pipeline using AWS CodePipeline and CodeBuild for automating Maven-based build and test processes

**Tools required:** AWS CodeBuild, AWS CodePipeline, and AWS Elastic Beanstalk

**Prerequisites:** Deployment of a sample application on AWS Elastic Beanstalk

Steps to be followed:

1. Import the repository from GitHub
2. Configure CodeBuild and CodePipeline to perform build and test automation

**Step 1: Import the repository from GitHub**

1. Go to the following URL and sign in to your GitHub account:

**https://github.com/**

A screenshot of a chat

Description automatically generated

1. Click on the **New** button to create a new repository

A screenshot of a chat

Description automatically generated

1. Name the repository as **Maven**, scroll down, and click on **Create repository**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

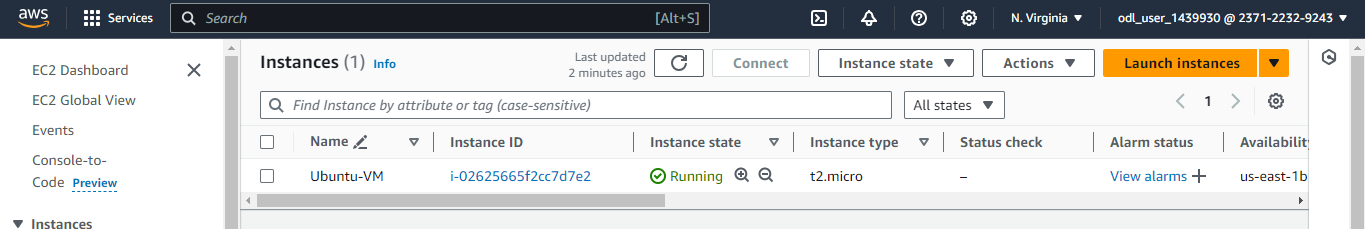
The repository is created successfully.

1. Sign in to the AWS Console

A screenshot of a computer

Description automatically generated

1. Create an EC2 instance and connect to it



1. Create a directory using the following command:

**mkdir demo**



1. Navigate to the created directory using the following command:

**cd demo**



1. Initialize Git using the following command:

**git init**

**A screenshot of a computer screen

Description automatically generated**

1. Clone the GitHub repository using the following command:

**git clone --mirror https://github.com/anujdevopslearn/MavenBuild MavenBuild**

A computer screen with white text

Description automatically generated

1. Switch to the cloned repository using the following command:

**cd MavenBuild/**



1. List all files present in the repository using the following command:

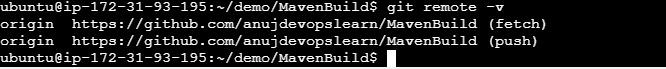
**ls -lart**

A screen shot of a computer

Description automatically generated

1. List all remote repositories using the following command:

**git remote -v**



1. Remove the existing remote configuration using the following command:

**git remote remove origin**

**A black screen with white text

Description automatically generated**

1. Add a new remote repository using the following command:

**git remote add origin Repository-link**



|  |
| --- |
| **Note**: Replace **Repository-link** with the link to the repository you created in Step 1.3 |

1. Verify if the new remote is correctly configured using the following command:

**git remote -v**



1. Upload your local repository content to the remote repository using the following command:

**git push --all origin**

A computer screen with white text

Description automatically generated

|  |
| --- |
| **Note**: When prompted, enter your GitHub credentials |

**Step 2: Configure CodeBuild and CodePipeline to perform build and test**

**automation**

1. Click on the **Create pipeline** button  
     
   **A screenshot of a computer

   Description automatically generated**
2. Add a namefor the pipeline and click on the **Next** button  
     
   A screenshot of a computer

   Description automatically generated

1. Under **Source provider**, select **AWS CodeCommit**

A screenshot of a computer

Description automatically generated

1. Select **MavenBuild** as the **Repository name**  
    **A screenshot of a computer

   Description automatically generated**

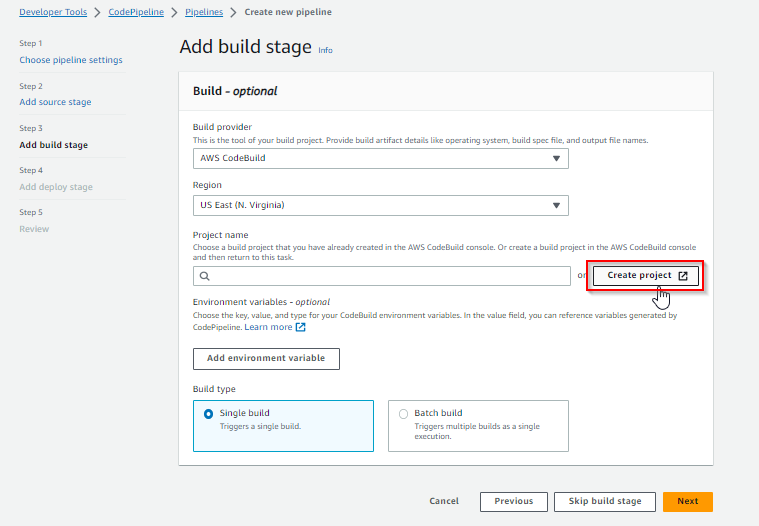
1. Select **master** as the **Branch name**

**A screenshot of a computer

Description automatically generated**

1. Once the details are entered, click on the **Next** button  
     
   A screenshot of a computer

   Description automatically generated
2. Select **AWS CodeBuild** as the **Build provider** for building and automating tests  
     
   A screenshot of a computer

   Description automatically generated
3. Click on the **Create project** button and a new pop-up window will appear  
     
   
4. In the new window, enter the name of the project  
     
   A screenshot of a project

   Description automatically generated

1. In the **Environment** section, add the details as shown in the screenshots below:

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. Under **Build specifications**, select the **Insert build commands** option and click on the   
    **Switch to editor** button  
     
   **A screenshot of a computer

   Description automatically generated**

1. Remove the existing build commands from the editor and enter the YAML code below:  
     
   **version: 0.2**

**phases:**

**build:**

**commands:**

**- "mvn clean install"**

**- "cp target/\*.war ROOT.war"**

**artifacts:**

**files:**

**- "ROOT.war"**

**name: $(date +%Y-%m-%d)**

**discard-paths: yes**

**base-directory: .  
  
A screenshot of a computer

Description automatically generated**

1. Scroll down to the bottom of the page and click on **Continue to CodePipeline  
     
   A screenshot of a computer

   Description automatically generated**

**A screenshot of a computer

Description automatically generated**

The project is successfully created.

1. Now, click the **Next** button  
     
   **A screenshot of a computer

   Description automatically generated**

1. Select **AWS Elastic Beanstalk** as the **Deploy provider** and **CounterApplication** as the **Application name**

|  |
| --- |
| **Note**: Refer to the previous demo to set up a Java-based Elastic Beanstalk environment on AWS and access a sample application deployed on Elastic Beanstalk |

A screenshot of a computer

Description automatically generated

1. Select **CounterApplication-env** as the **Environment name**  
     
   A screenshot of a computer

   Description automatically generated
2. Click on the **Next** button  
     
   A screenshot of a computer

   Description automatically generated

1. Scroll to the bottom of the page and click on the **Create pipeline** button **A screenshot of a computer

   Description automatically generated**

After creating the pipeline, the execution will start.  
  
A screenshot of a computer

Description automatically generated  
  
  
A screenshot of a computer

Description automatically generated

1. Once the build stage is completed, click on the **View logs** button in the **Build** section for validation  
    **A screenshot of a computer

   Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. Click on the **AWS Elastic Beanstalk** link in the **Deploy** section **A white rectangular object with a white background

   Description automatically generated**
2. Now, click on the **Domain** link **A screenshot of a computer

   Description automatically generated**

You will be able to see the following interface: **A screenshot of a computer

Description automatically generated**

By following these steps, you have successfully implemented a blue/green deployment strategy using CodeDeploy and established proficiency in setting up a continuous integration pipeline with CodePipeline and CodeBuild to automate Maven-based build and test processes.