

Name - Mikil. Lalwani

D15B/37

Advance Dev Ops Lab 2

## Experiment 2

- Aim -

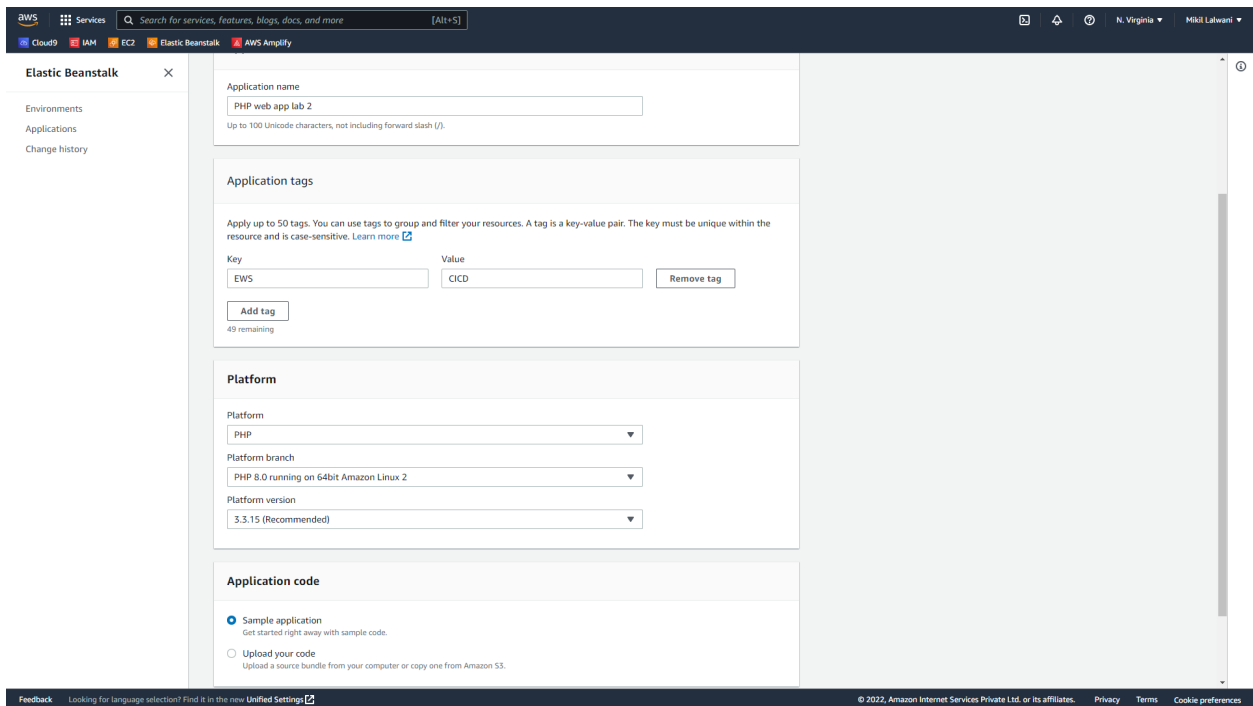
To build your application using AWS Codebuild and deploy on S3/SEBS using AWS Code Pipeline, deploy sample application on EC2 instance using AWS CodeDeploy.

- Theory -

- Continuous deployment allows you to deploy revisions to a production environment automatically without explicit approval from a developer, making the entire software release procedure automated.
- Pipeline is created by using AWS Code Pipeline, a service that builds, tests and deploys your code every time there is code change.
- We can use ~~Git~~ Github account, Amazon S3 bucket or 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 app.

## Step 1: Create a deployment environment

- 1) 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.
  - 1) To simplify the process of setting up and configuring EC2 instances for this tutorial, 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.
- 2) Name your web app and choose PHP from the drop-down menu(or any other language you are interested in) and then click Create Application.



The screenshot shows the AWS Elastic Beanstalk console. The left sidebar contains links for 'Environments', 'Applications', and 'Change history'. The main content area is titled 'Elastic Beanstalk' and contains the following sections:

- Application name:** A text input field containing 'PHP web app lab 2'. Below the field is a note: 'Up to 100 Unicode characters, not including forward slash (/)'.
- Application tags:** A section for adding tags. It includes a description: 'Apply up to 50 tags. You can use tags to group and filter your resources. A tag is a key-value pair. The key must be unique within the resource and is case-sensitive. [Learn more](#)'. Below this is a table with two columns: 'Key' and 'Value'. The first row has 'EWS' in the Key column and 'CICD' in the Value column. There is an 'Add tag' button and a 'Remove tag' button. A note at the bottom says '49 remaining'.
- Platform:** A section for selecting the platform. It includes three dropdown menus: 'Platform' (set to 'PHP'), 'Platform branch' (set to 'PHP 8.0 running on 64bit Amazon Linux 2'), and 'Platform version' (set to '3.3.15 (Recommended)').
- Application code:** A section for selecting the application code. It has two radio buttons: 'Sample application' (selected) and 'Upload your code'. Below 'Sample application' is a note: 'Get started right away with sample code.' Below 'Upload your code' is a note: 'Upload a source bundle from your computer or copy one from Amazon S3.'

The footer of the console shows 'Feedback' and 'Looking for language selection? Find it in the new Unified Settings'.

- 3) Elastic Beanstalk will begin creating a sample environment for you to deploy your application. It will create an Amazon EC2 instance, a security group, an Auto Scaling group, an Amazon S3 bucket, Amazon CloudWatch alarms, and a domain name for your application.

**Note: This will take several minutes to complete.**

## Step2: Get a copy of the sample code

In this step, you will retrieve a copy of the sample app's code and choose a source to host the code. The pipeline takes code from the source and then performs actions on it. You can use one of three options as your source: a GitHub repository, an Amazon S3 bucket, or an AWS CodeCommit repository. Select your preference and follow the steps below:

Fork the repository given below to your Github account.

Github repository: [aws-codepipeline-s3-codedeploy-linux](#)

## Step 3: Create your Pipeline

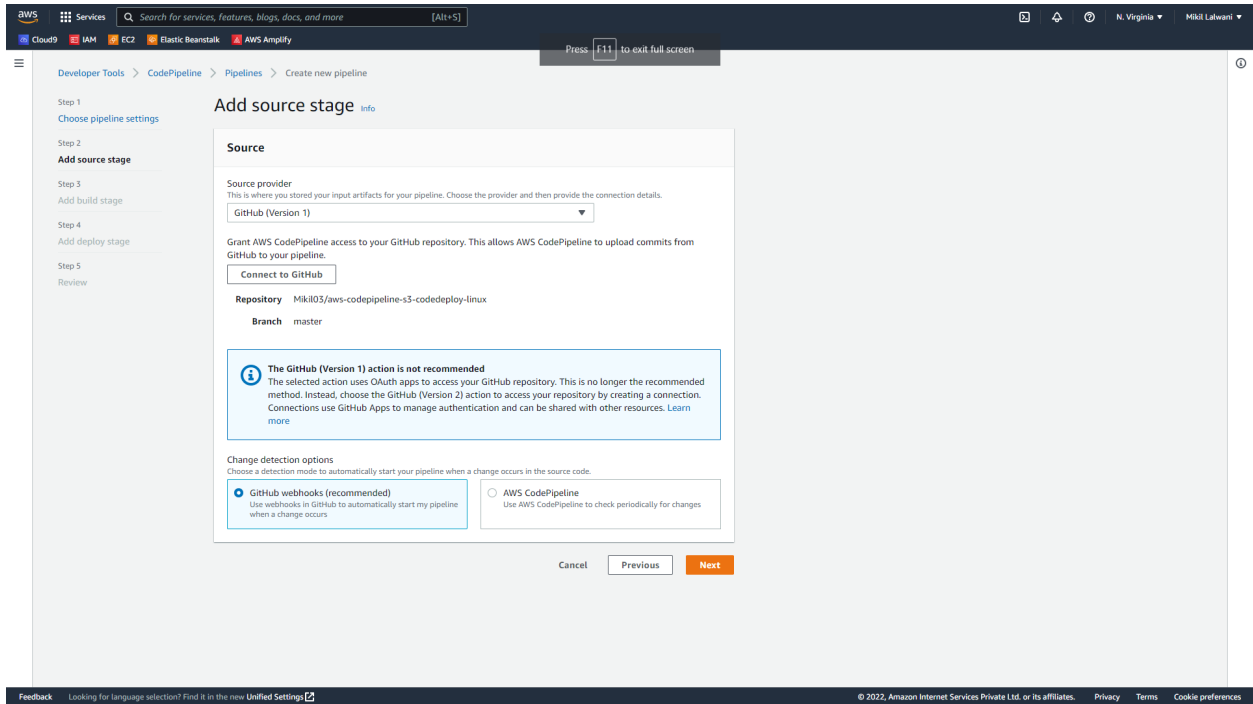
- 1) In this step, you will create and configure a simple pipeline with two actions: source and deploy. You will provide CodePipeline with the locations of your source repository and deployment environment. A true continuous deployment pipeline requires a build stage, where code is compiled and unit tested. CodePipeline lets you plug your preferred build provider into your pipeline. However, in this, we will skip the build stage. Goto Pipeline again

and create it.

The screenshot shows the AWS CodePipeline console interface. The top navigation bar includes the AWS logo, a search bar, and a list of services: Cloud9, IAM, EC2, Elastic Beanstalk, and AWS Amplify. The breadcrumb trail indicates the current location: Developer Tools > CodePipeline > Pipelines > Create new pipeline. A sidebar on the left lists the steps of the pipeline creation process: Step 1: Choose pipeline settings (selected), Step 2: Add source stage, Step 3: Add build stage, Step 4: Add deploy stage, and Step 5: Review. The main content area is titled 'Choose pipeline settings' and contains a 'Pipeline settings' form. The form has two sections: 'Pipeline name' and 'Service role'. In the 'Pipeline name' section, the name 'PHP\_web\_app\_Lab\_2' is entered in a text box, with a note stating 'Enter the pipeline name. You cannot edit the pipeline name after it is created.' and a character limit of 'No more than 100 characters'. In the 'Service role' section, the 'New service role' radio button is selected, with the subtext 'Create a service role in your account'. The 'Existing service role' option is unselected, with the subtext 'Choose an existing service role from your account'. Below this, the 'Role name' section shows the text 'AWSCodePipelineServiceRole-us-east-1-PHP\_web\_app\_Lab\_2' in a text box, with a note 'Type your service role name'. A checkbox labeled 'Allow AWS CodePipeline to create a service role so it can be used with this new pipeline' is checked. At the bottom of the form, there is a section for 'Advanced settings' with a right-pointing arrow. At the bottom right of the form, there are 'Cancel' and 'Next' buttons. The footer of the console shows a feedback link, a language selection notice, and copyright information for Amazon Internet Services Private Ltd. or its affiliates, along with links for Privacy, Terms, and Cookie preferences.

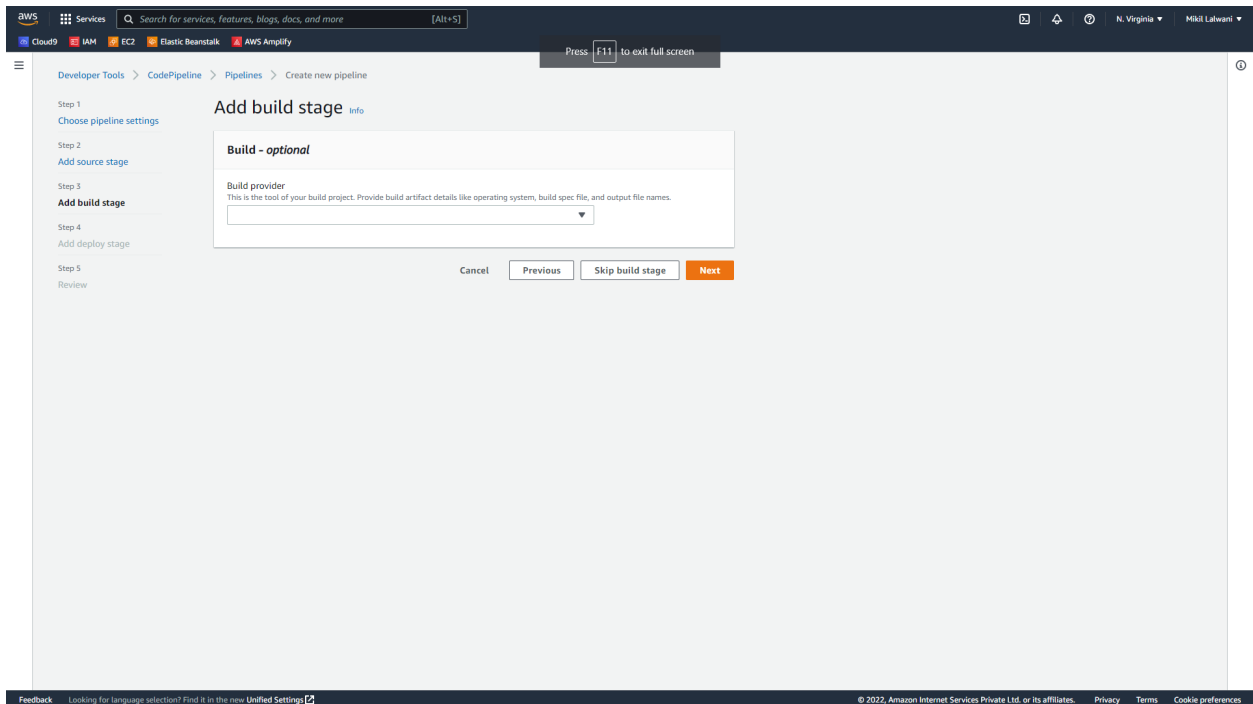
#### Step 4: Source Stage:

- 1) Select the source provider as GitHub (Version 1).
- 2) Click on Connect to GitHub and provide AWS access to your GitHub account.
- 3) Select the appropriate repository from the dropdown menu.
- 4) Select the branch as master.
- 5) Select GitHub webhooks to update the pipeline as soon as changes are detected.
- 6) Click on next.



## Step 5: Build Stage:

### 1) Click on Skip build stage.



## Step 6: Deploy Stage:

- 1) Deployment provider: Click AWS Elastic Beanstalk. Application name: MYEBS. Environment name: Click Myebs-env. Click Next step.

The screenshot shows the AWS CodePipeline console interface. On the left, a sidebar lists the steps of the pipeline: Step 1 (Choose pipeline settings), Step 2 (Add source stage), Step 3 (Add build stage), Step 4 (Add deploy stage - currently selected), and Step 5 (Review). The main area is titled 'Add deploy stage' and contains a 'Deploy' configuration form. A blue information box at the top states: 'You cannot skip this stage. Pipelines must have at least two stages. Your second stage must be either a build or deployment stage. Choose a provider for either the build stage or deployment stage.' The 'Deploy' form includes the following fields: 'Deploy provider' (set to 'AWS Elastic Beanstalk'), 'Region' (set to 'US East (N. Virginia)'), 'Application name' (set to 'PHP web app lab 2'), and 'Environment name' (set to 'Phwebapplab2-env'). At the bottom of the form are 'Cancel', 'Previous', and 'Next' buttons. The top of the console shows the AWS logo, navigation menu, search bar, and user information (N. Virginia, Mital Lalwani).

- 2) 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.
- 3) 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.

**Elastic Beanstalk**

- Environments
- Applications
- Change history

▼ PHP web app lab 2

- Application versions
- Saved configurations

▼ **Phpwebapplab2-env**

- Go to environment
- Configuration
- Logs
- Health
- Monitoring
- Alarms
- Managed updates
- Events
- Tags

**AWS Graviton now supported**  
AWS Graviton, an arm64-based processor, can offer up to 40% better price performance over the comparable x86 processor. To upgrade to an arm64 instance type, choose it in the 'Capacity' settings in 'Additional configuration.'

Elastic Beanstalk > Environments > Phpwebapplab2-env

### Phpwebapplab2-env

Phpwebapplab2-env-eba-5xpgm9z-us-east-1.elasticbeanstalk.com  
Application name: PHP web app lab 2

Refresh
Actions

#### Health

Ok

Causes

#### Running version

Sample Application

Upload and deploy

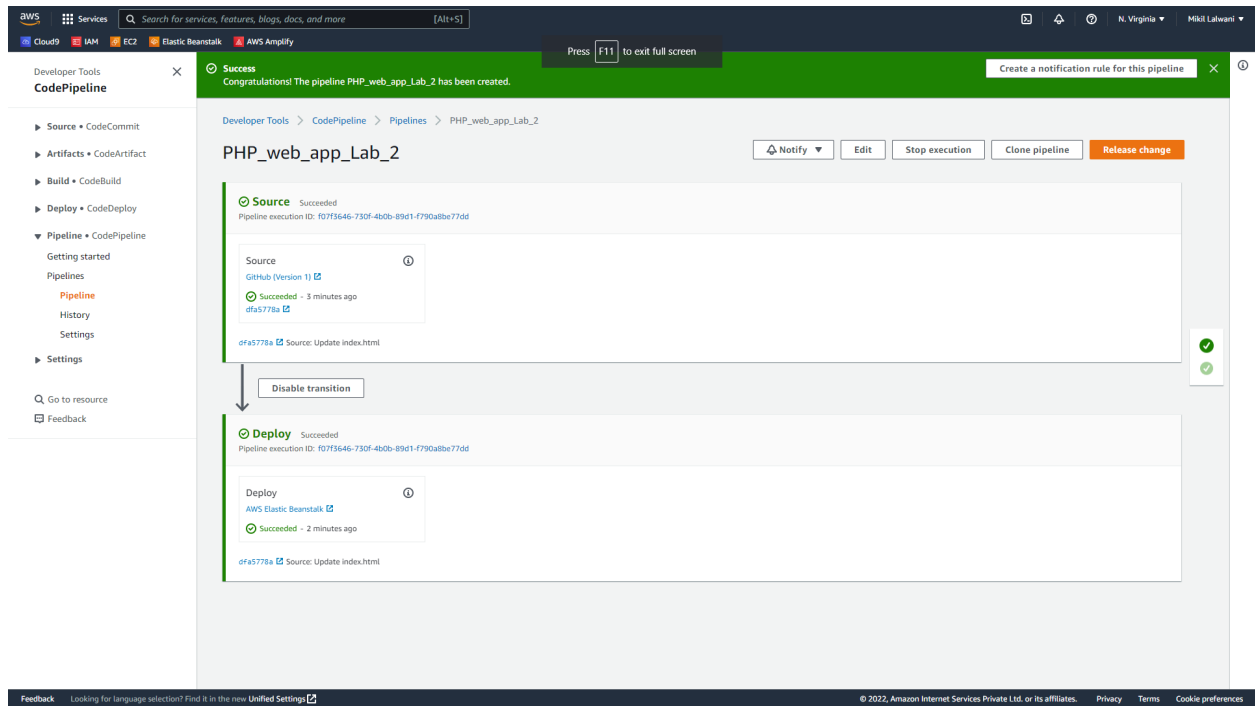
#### Platform

PHP 8.0 running on 64bit Amazon Linux 2/3.3.15

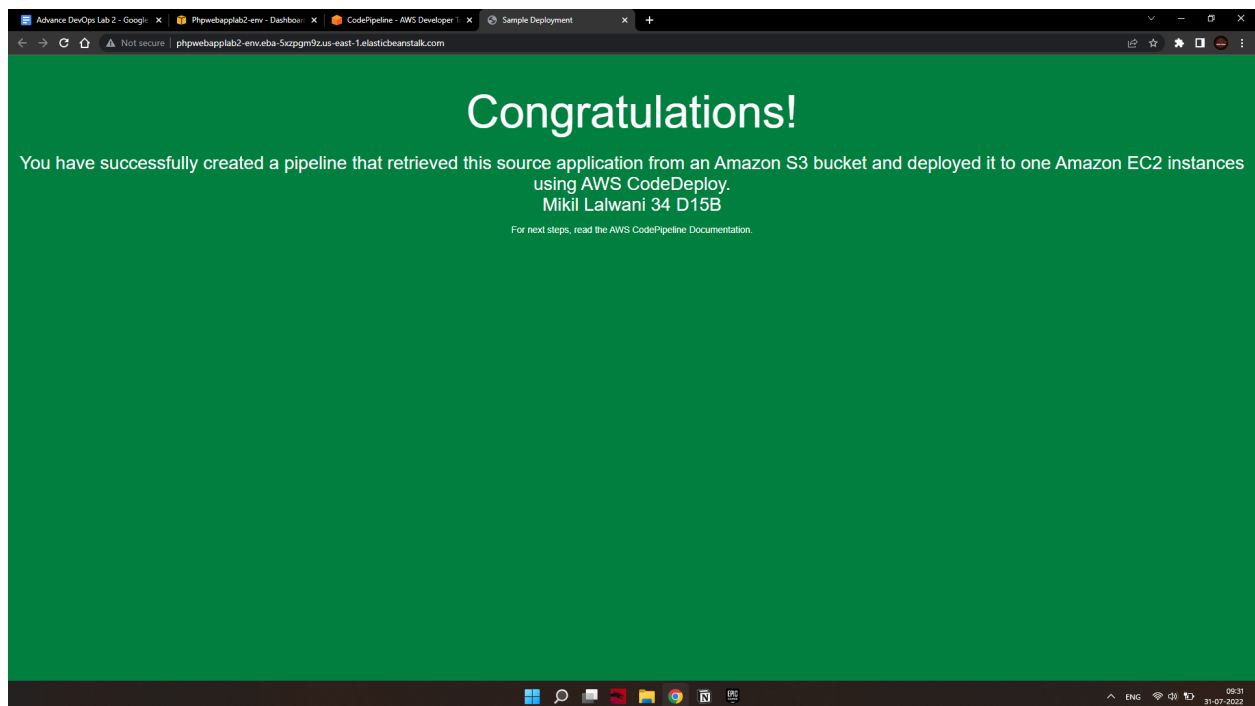
Change

#### Recent events

Time	Type	Details
2022-07-31 09:03:53 UTC+0530	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 35 seconds ago and took 3 minutes.
2022-07-31 09:03:07 UTC+0530	INFO	Successfully launched environment: Phpwebapplab2-env
2022-07-31 09:03:06 UTC+0530	INFO	Application available at Phpwebapplab2-env-eba-5xpgm9z-us-east-1.elasticbeanstalk.com.
2022-07-31 09:02:53 UTC+0530	INFO	Added instance [i-0e087dfa2562b180a] to your environment.
2022-07-31 09:02:51 UTC+0530	INFO	Instance deployment completed successfully.



Using CodePipeline, you 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.



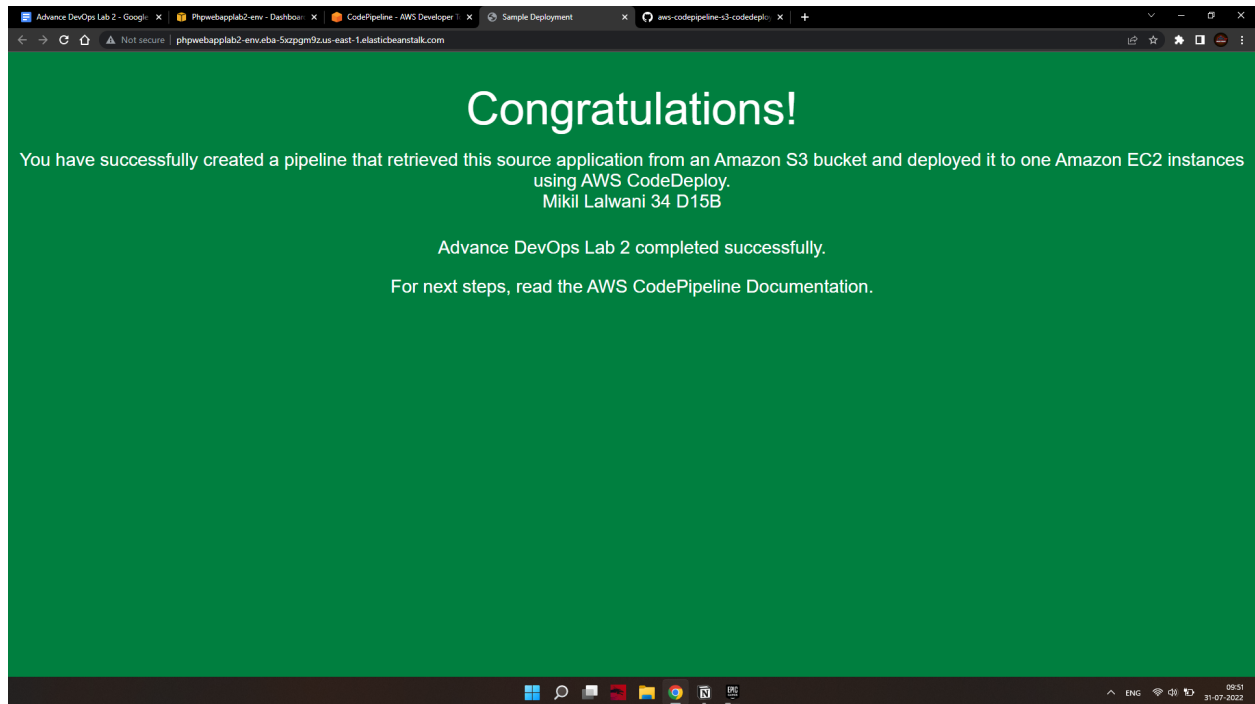


## Step 7: Commit a change and then update your app

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 your EC2 instance via Elastic Beanstalk.

Note that the sample web page you deployed refers to AWS CodeDeploy, a service that automates code deployments. CodePipeline, CodeDeploy is an alternative to using Elastic Beanstalk for deployment actions. Let's update the sample code so that it correctly states that you deployed the sample using Elastic Beanstalk.

- a. Visit your own copy of the repository that you forked in GitHub.
  - Open index.html
  - Select the Edit icon
- b. Update the webpage by copying and pasting the following text on line 30:
- c. Commit the change to your repository.
- d. 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.
- e. The AWS Elastic Beanstalk console opens with the details of the deployment. Select the environment you created earlier. And click the URL again from the EBS environment again.



## Step 8: Clean up your resources

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

a. First, you will delete your pipeline:

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

b. Second, delete your Elastic Beanstalk application:

- Visit the Elastic Beanstalk console.
- Click Actions.
- Then click Terminate Environment.

You have successfully created an automated software release pipeline using AWS CodePipeline! Using CodePipeline, you 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. Your pipeline will automatically deploy your code every time there is a code change.

## Conclusion-

Thus we have successfully built our application using AWS codebuild and AWS CodePipeline on EC2 instance using AWS CodeDeploy.