

Name:Shreyash Kamat

Div/Roll No. : D15C/22

## Exp 02:To Build Your Application using AWS Code Build and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS Code Deploy.

### Step 1: Create our Elastic Beanstalk Environment

Login into your AWS account and navigate to services. Search for Elastic Beanstalk service and click on create application. Give your application a suitable name. For the platform, select PHP. The rest of the configuration settings are to be kept as default.

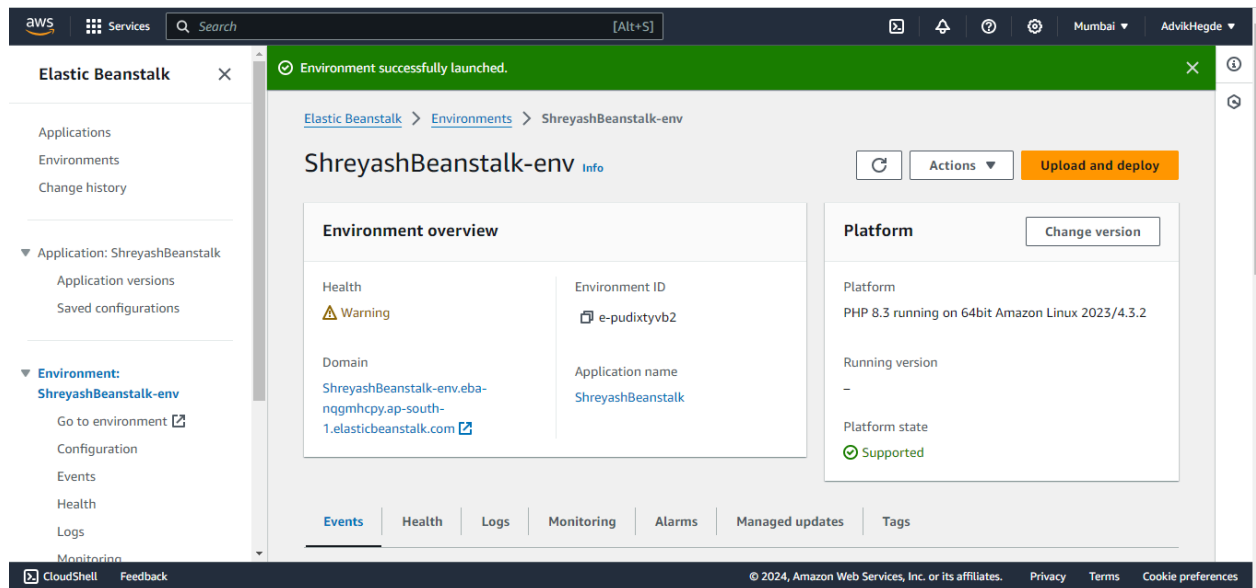
The screenshot shows the AWS Elastic Beanstalk console. The main heading is 'Configure environment'. On the left, a sidebar lists steps: Step 1 (Configure environment), Step 2 (Configure service access), Step 3 (optional: Set up networking, database, and tags), Step 4 (optional: Configure instance traffic and scaling), Step 5 (optional: Configure updates, monitoring, and logging), and Step 6 (Review). The 'Environment tier' section offers two options: 'Web server environment' (selected) and 'Worker environment'. The 'Application information' section includes a text input for 'Application name' containing 'ShreyashBeanstalk' and a section for 'Application tags (optional)'.

This screenshot displays the 'Custom platform' configuration step in the AWS Elastic Beanstalk console. It shows three dropdown menus: 'Platform' set to 'PHP', 'Platform branch' set to 'PHP 8.3 running on 64bit Amazon Linux 2023', and 'Platform version' set to '4.3.1 (Recommended)'. Below these, the 'Application code' section has 'Sample application' selected. The footer of the console includes 'CloudShell', 'Feedback', and a copyright notice for 2024.

Now, while creating the environment, we are asked to provide an IAM role with the necessary EC2 permissions. We are supposed to make sure that we have made an existing IAM role with the following set of permissions:

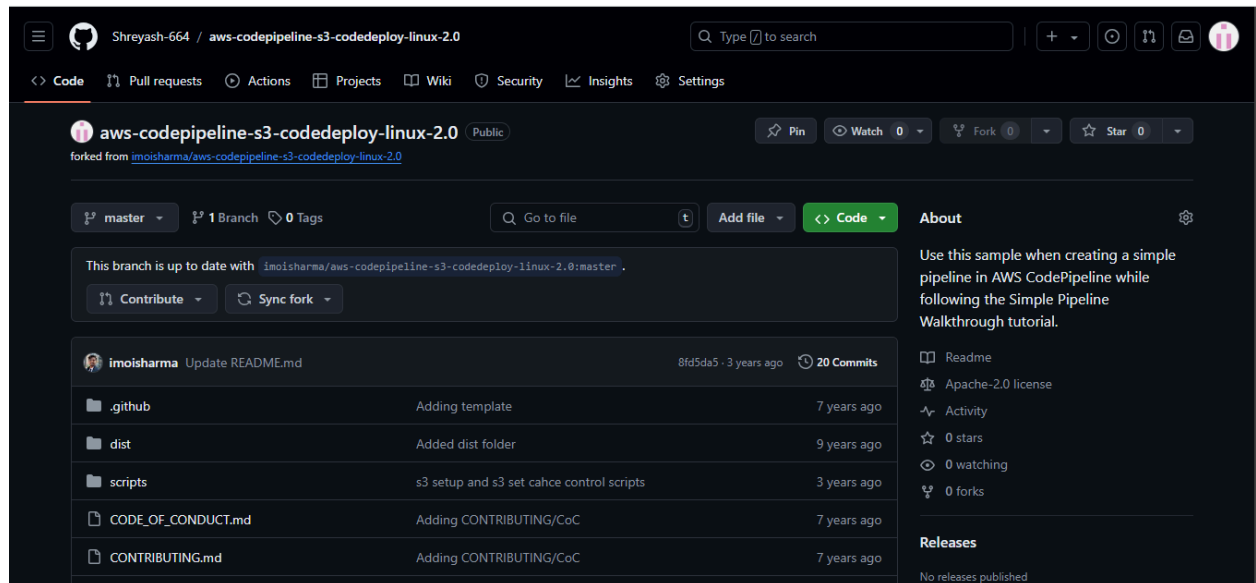
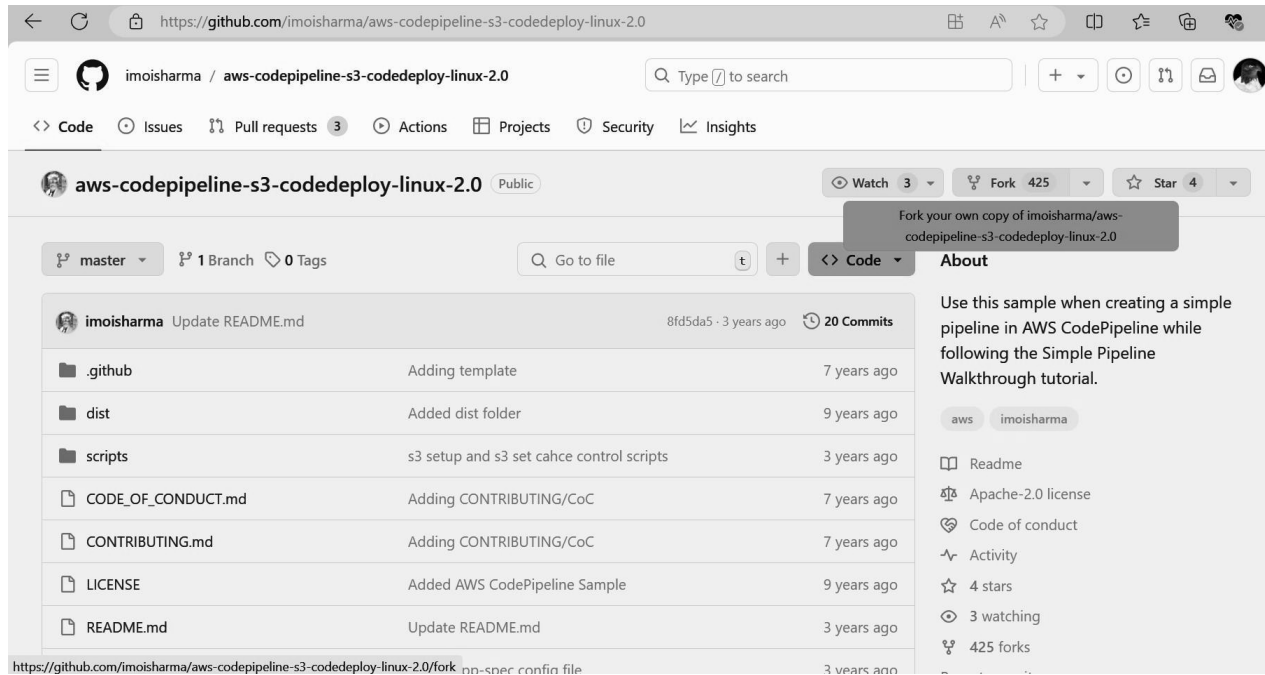
1. AWSElasticBeanStalkWebTier
2. AWSElasticBeanStalkWorkerTier
3. AWSElasticBeanStalkMulticontainerDocker

We can skip the steps to follow after the initial few steps mentioned above and move straight to review the settings of our environment. After reviewing everything properly, our environment can successfully be created.



## Step 2: Fork the required repository onto our github account

The repository to be forked is - imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0



This step is necessary for the execution of the steps to follow. It will be helpful in the creation of a pipeline.

### Step 3: Creation of the Pipeline

Navigate to Codepipeline inside Developer Tools. Give a suitable name to the pipeline you want to create.

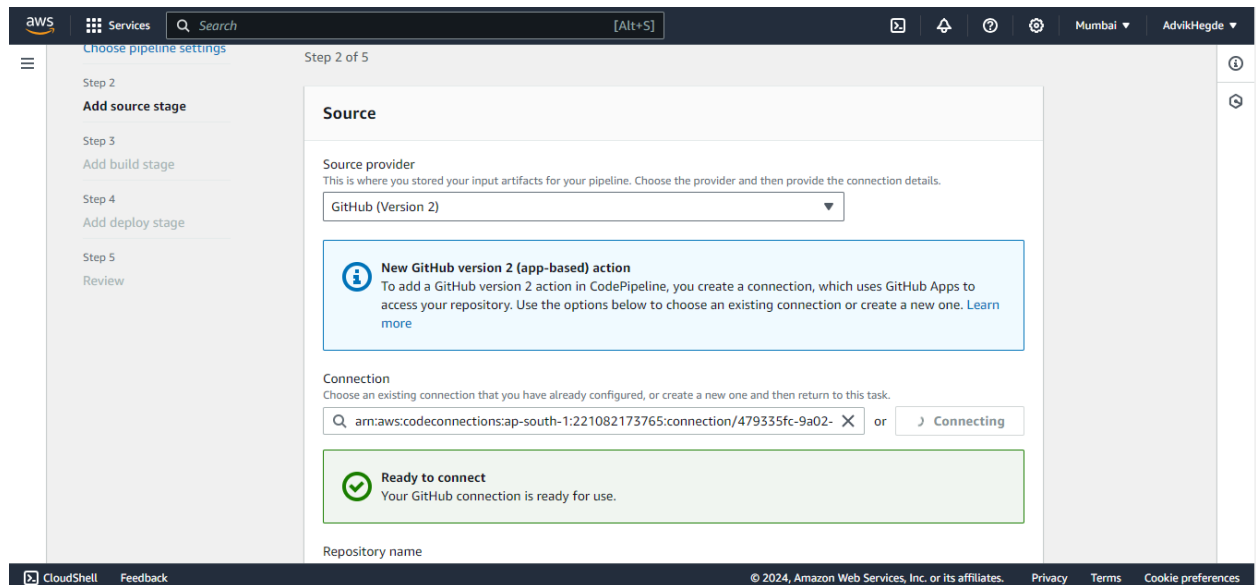
The screenshot shows the AWS CodePipeline console interface. The top navigation bar includes the AWS logo, 'Services', a search bar, and user information. The breadcrumb trail indicates the path: Developer Tools > CodePipeline > Pipelines > Create new pipeline. On the left, a sidebar lists the steps: Step 1 (Choose pipeline settings), 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 is labeled 'Step 1 of 5'. It contains a 'Pipeline settings' section with a 'Pipeline name' field (containing 'ShreyashPipeline') and a 'Pipeline type' dropdown (set to 'V2'). A message box states: 'You can no longer create V1 pipelines through the console. We recommend you use the V2 pipeline type with improved release safety, pipeline triggers, parameterized pipelines, and a new billing model.' Below this, the 'Execution mode' is set to 'Superseded'. At the bottom, a status bar shows 'Queued (Pipeline type V2 required)'.

And click on next ...

The screenshot shows the AWS CodePipeline console interface for the 'Add source stage' step. The breadcrumb trail is: Developer Tools > CodePipeline > Pipelines > Create new pipeline. The sidebar shows the steps: 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 'Add source stage' and is labeled 'Step 2 of 5'. It contains a 'Source' section with a 'Source provider' dropdown (set to 'GitHub (Version 2)'). A message box states: 'New GitHub version 2 (app-based) action. To add a GitHub version 2 action in CodePipeline, you create a connection, which uses GitHub Apps to access your repository. Use the options below to choose an existing connection or create a new one. Learn more'. Below this, the 'Connection' field is empty, and the 'Repository name' field is empty. The 'Default branch' field is empty.

## Step 4: GitHub connection

In this step, we are supposed to create a GitHub connection and add our existing repository over here i.e. the one we forked earlier.



We are supposed to enter our GitHub username so as to proceed towards making the connection.



## Create a connection [Info](#)

### Create GitHub App connection [Info](#)

Connection name

► Tags - *optional*

Connect to GitHub



Now to finalize our connection, we are to install an application which connects AWS to our GitHub account and repository.

Post the establishment of the connection, this is the message that is displayed. We can further select the branch of our repository that we want to connect.

The screenshot shows the AWS CodePipeline console interface. At the top, there's a navigation bar with the AWS logo, 'Services', a search bar, and a keyboard shortcut '[Alt+S]'. On the right, there are icons for help, notifications, and settings, along with the location 'Mumbai' and the user 'AdvikHegde'. The main content area is titled 'Repository name' with the instruction 'Choose a repository in your GitHub account.' Below this, a text input field contains 'Shreyash-664/aws-codepipeline-s3-codedeploy-linux-2.0'. A yellow warning box follows, stating: 'An unspecified error occurred. Check your network connectivity, and then check to see if there are any issues with the service at the [Service Health Dashboard](#). (Click here to retry)'. Below the warning, a note says: 'You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.' The 'Default branch' section has the instruction 'Default branch will be used only when pipeline execution starts from a different source or manually started.' and a text input field with 'master'. The 'Output artifact format' section has the instruction 'Choose the output artifact format.' and two radio button options: 'CodePipeline default' (selected) and 'Full clone'. The 'CodePipeline default' option description is: 'AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.' The 'Full clone' option description is: 'AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.' At the bottom, there's a footer bar with 'CloudShell', 'Feedback', '© 2024, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

aws Services Search [Alt+S]

Repository name  
Choose a repository in your GitHub account.

Shreyash-664/aws-codepipeline-s3-codedeploy-linux-2.0

**⚠ An unspecified error occurred. Check your network connectivity, and then check to see if there are any issues with the service at the [Service Health Dashboard](#). (Click here to retry)**

You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

**Default branch**  
Default branch will be used only when pipeline execution starts from a different source or manually started.

master

**Output artifact format**  
Choose the output artifact format.

☒ **CodePipeline default**  
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

☐ **Full clone**  
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## Step 5: Deployment stage:

We are expected to skip the build stage and move towards the deployment step. In the deployment step we are supposed to choose the Elastic Beanstalk application and the environment that we created earlier and proceed with our pipeline creation.

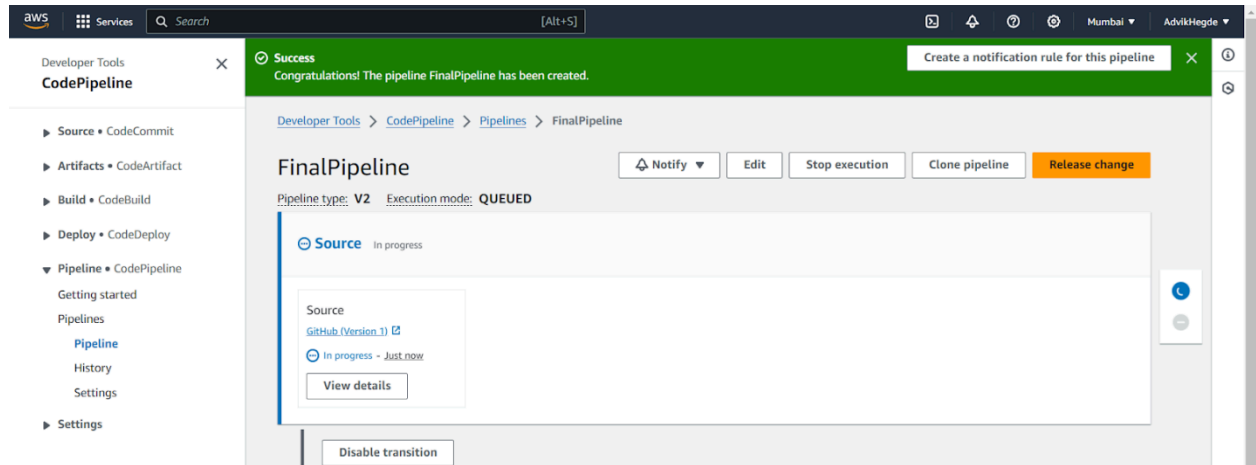
The screenshot shows the AWS CodePipeline console interface for configuring a deployment stage. The top navigation bar includes the AWS logo, 'Services', a search bar, and user information for 'Mumbai' and 'AdvikHegde'. The left sidebar shows the pipeline structure with 'Add deploy stage', 'Step 5', and 'Review' options. The main 'Deploy' section contains the following configuration fields:

- Deploy provider:** A dropdown menu set to 'AWS Elastic Beanstalk'.
- Region:** A dropdown menu set to 'Asia Pacific (Mumbai)'.
- Input artifacts:** A section with a text input field and a 'Learn more' link. Below the field is a note: 'No more than 100 characters'.
- Application name:** A search input field containing 'ShreyashBeanstalk'.
- Environment name:** A search input field containing 'ShreyashBeanstalk-env'.

At the bottom of the main section, there is a checkbox labeled 'Configure automatic rollback on stage failure'. The footer of the console includes 'CloudShell', 'Feedback', copyright information '© 2024, Amazon Web Services, Inc. or its affiliates.', and links for 'Privacy', 'Terms', and 'Cookie preferences'.

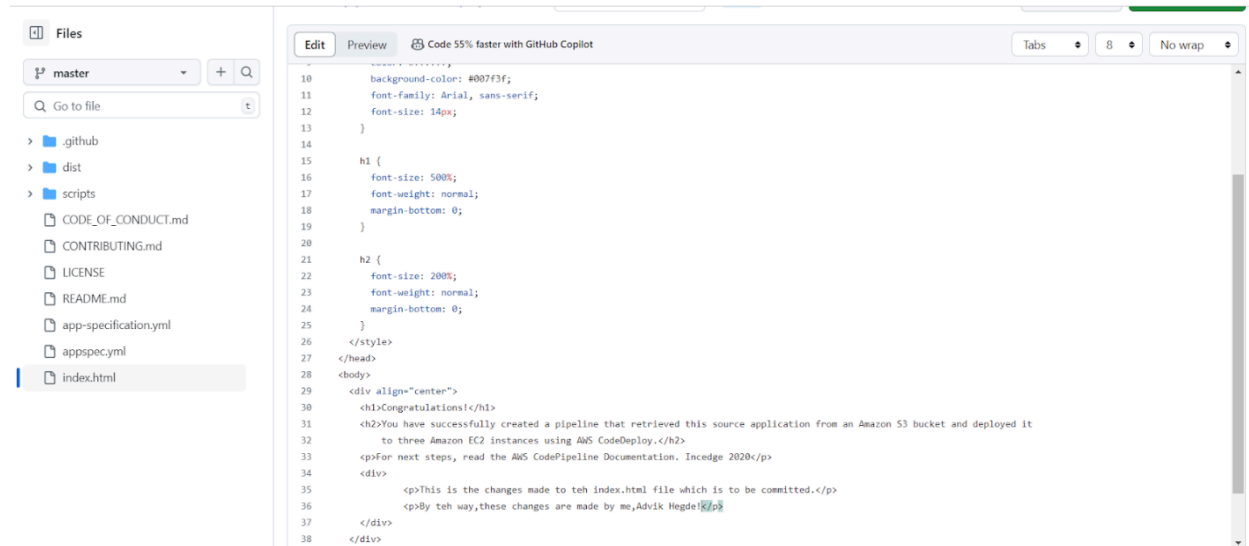


**Step 6: Post deployment stage:** When all the stages run successfully, this is what is displayed onto the screen. It shows us that our application and our environment have successfully been deployed using a dedicated pipeline created.



## Step 7: Committing changes to your GitHub code

Now, we will go to our forked repository and make some changes to the index.html file. On making the desired changes, we are supposed to commit those changes on our forked repository. Write a good commit message so as to recognize it when it appears on the pipeline.



Once the changes have been applied, we see the commit message that we wrote for the latest commit on our repository being reflected on our pipeline. Over here, it would be seen somewhere near the bottom of the image that is attached. "Update index.html" was the latest commit message in the GitHub repository.

Success  
Congratulations! The pipeline FinalPipeline has been created.

Success  
The most recent change will re-run through the pipeline. It might take a few moments for the status of the run to show in the pipeline view.

Developer Tools > CodePipeline > Pipelines > FinalPipeline

### FinalPipeline

Pipeline type: V2 Execution mode: QUEUED

Notify Edit Stop execution Clone pipeline Release change

Source Succeeded  
Pipeline execution ID: 0f9bb793-f278-42d7-8c0c-146557e9b8fb

Source  
GitHub (Version 1)  
Succeeded - Just now  
3cf895ae  
View details

3cf895ae Source: Updating the index.html file to contain a div and paragraph elements that are ad \*\*\*

## Step 9: Open the Domain of our Elastic Beanstalk environment

Now, we navigate back to our Elastic Beanstalk environment and open the environment domain of our deployed application.

The text in this image is clearly distinguishable from the earlier website's text meaning that the changes that we made to our code in index.html has successfully been applied to the website that we deployed.

# Congratulations!

You have successfully created a pipeline that retrieved this source application from an Amazon S3 bucket and deployed it to three Amazon EC2 instances using AWS CodeDeploy.

For next steps, read the AWS CodePipeline Documentation. Incedge 2020  
This is the changes made to teh index.html file which is to be committed.  
By teh way,these changes are made by me,Advik Hegde!



