

Experiment 2

Name: **Dev Gaonkar**

Div/Roll no: **D15C/ 12**

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

Step 1: Create a Deployment Environment

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. Open up Elastic Beanstalk and name your web app.

Application information [Info](#)

Application name

Maximum length of 100 characters.

► Application tags (optional)

Environment information [Info](#)

Choose the name, subdomain and description for your environment. These cannot be changed later.

Environment name

Must be from 4 to 40 characters in length. The name can contain only letters, numbers, and hyphens. It can't start or end with a hyphen. This name must be unique within a region in your account.

2. Choose PHP from the drop-down menu and then click Create Application.

Platform [Info](#)

Platform type
☒ Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#)
☐ Custom platform
Platforms created and owned by you. This option is unavailable if you have no platforms.

Platform

Platform branch

Platform version

3. Beanstalk creates a sample environment for you to deploy your application.
By default, it creates an EC2 instance, a security group, an Auto Scaling group, an Amazon S3 Bucket, Amazon CloudWatch alarms and a domain name for your application.

Step 2: Get a copy of your sample code



In this step, we will get the sample code from [this](#) GitHub Repository to later host it. The pipeline takes code from the source and then performs actions on it.

For this experiment, as a source, we will use this forked GitHub repository. We can alternatively also use Amazon S3 and AWS CodeCommit.

Go to the repository shared above and simply fork it.



Step 3: Creating a CodePipeline

In this step, we'll create a simple pipeline that has its source and deployment information. In this case, however, we will skip the build stage where you get to plug in our preferred build provider.

1. Go to AWS Developer Tools -> CodePipeline and create a new Pipeline. Fill in the initial settings first.

Pipeline settings

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.

pipeline-dev12

No more than 100 characters

Pipeline type

i 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.

Execution mode
Choose the execution mode for your pipeline. This determines how the pipeline is run.

☐ Superseded
A more recent execution can overtake an older one. This is the default.

☒ Queued (Pipeline type V2 required)
Executions are processed one by one in the order that they are queued.

☐ Parallel (Pipeline type V2 required)
Executions don't wait for other runs to complete before starting or finishing.

Service role

☒ New service role
Create a service role in your account

☐ Existing service role
Choose an existing service role from your account

Role name

AWSCodePipelineServiceRole-ap-south-1-pipeline-dev12

Type your service role name

☒ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

2. In the source stage, choose GitHub v2 as the provider, then connect your GitHub account to AWS by creating a connection. You'd need your GitHub credentials and then you'd need to authorize and install AWS on the forked GitHub Repository.

Source

Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (Version 2)



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](#)

Connection

Choose an existing connection that you have already configured, or create a new one and then return to this task.

arn:aws:codeconnections:ap-south-1:860015268757:connection/47dc3241

or

Connect to GitHub



Ready to connect

Your GitHub connection is ready for use.

Repository name

Choose a repository in your GitHub account.

GaonkarDev/aws-codepipeline-s3-codedeploy-linux-2.0

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.

3. Then, simply choose this forked repository and the branch which you will be able to find in the search box. After that, click Continue and skip the build stage. Proceed to the Deployment stage.

Step 4: Deployment

1. Choose Beanstalk as the Deploy Provider, same region as the Bucket and Beanstalk, name and environment name. Click Next, Review and create the pipeline.

Deploy

Deploy provider

Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

AWS Elastic Beanstalk ▼

Region

Asia Pacific (Mumbai) ▼

Input artifacts

Choose an input artifact for this action. [Learn more](#) 

SourceArtifact ▼

No more than 100 characters

Application name

Choose an application that you have already created in the AWS Elastic Beanstalk console. Or create an application in the AWS Elastic Beanstalk console and then return to this task.

🔍 mywebapp12 ✕

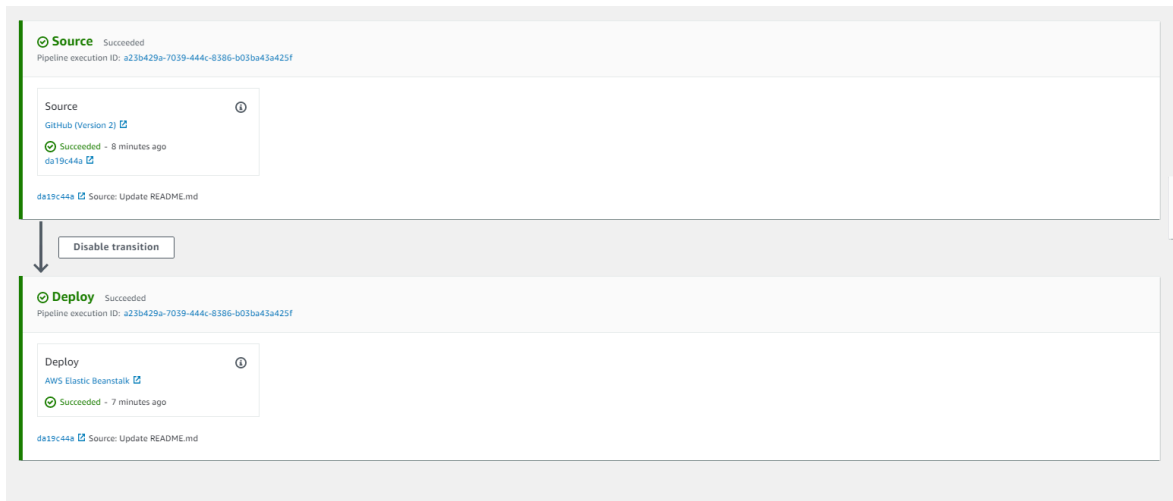
Environment name

Choose an environment that you have already created in the AWS Elastic Beanstalk console. Or create an environment in the AWS Elastic Beanstalk console and then return to this task.

🔍 Mywebapp12-env ✕



☒ Configure automatic rollback on stage failure


2. In a few minutes, we will have our pipeline created. Once we have the success message on the Deploy part, we can go ahead and check our URL provided in the EBS environment.



The screenshot shows the AWS CodePipeline console with a successful pipeline execution. The pipeline has two stages: Source and Deploy. The Source stage is successful, and the Deploy stage is also successful. The pipeline execution ID is a23b429a-7039-444c-8386-b03ba43a425f.


Source Succeeded
Pipeline execution ID: a23b429a-7039-444c-8386-b03ba43a425f


Source
GitHub (Version 2) 
Succeeded - 8 minutes ago
da19c44a 

da19c44a  Source: Update README.md

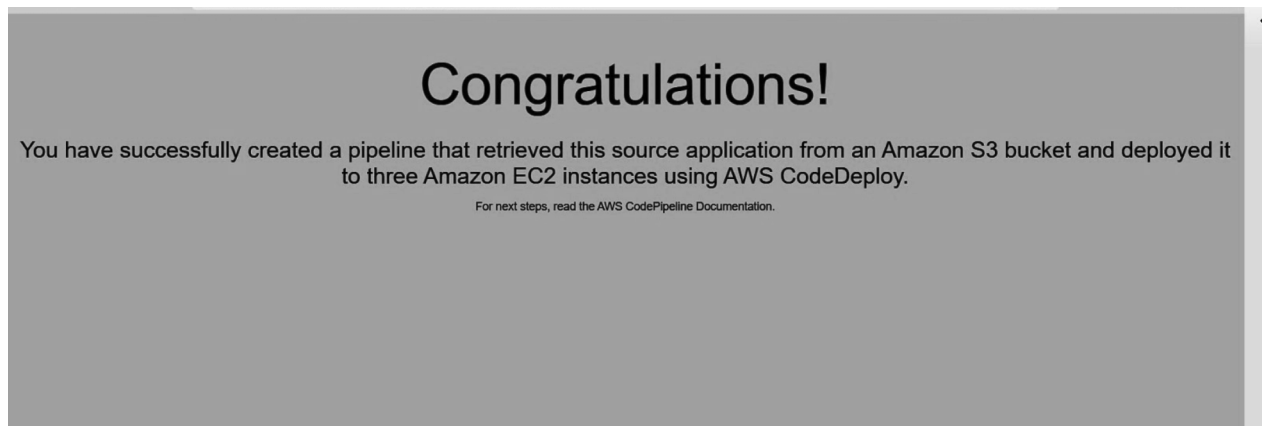
Disable transition

Deploy Succeeded
Pipeline execution ID: a23b429a-7039-444c-8386-b03ba43a425f

Deploy
AWS Elastic Beanstalk 
Succeeded - 7 minutes ago

da19c44a  Source: Update README.md

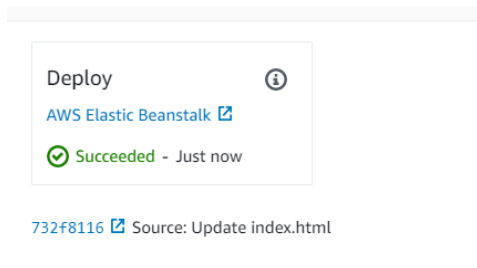
This is the sample website we just created.



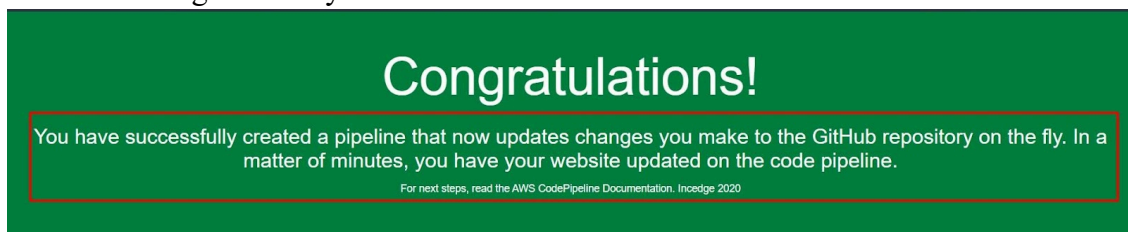
If you can see this, that means that you successfully created an automated software using CodePipeline.

Step 5: Committing changes to update app

1. In this step, we will update the code which we had and make a few changes to the HTML file (keep in mind, this is in our version of the forked repository).
2. In GitHub, open index.html. Then, make changes to either the heading tag or the paragraph tag. Commit these changes on the fly on GitHub.
3. You can view the changes on the website using the same URL, once the deployment section shows success.



4. Check the changes live on your website.



Conclusion:

In this experiment, we learned how to use AWS Elastic Beanstalk Environments to deploy our websites and create a CodePipeline under the AWS Development Console, source data to the Beanstalk using GitHub and finally, make real-time changes to the website just by pushing updates to GitHub.