**Automate Your Deployments: Building a Robust CI/CD Pipeline for Application Releases**

**What is**[**AWS CodePipeline**](https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html)**?**

AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously.

**Why do we need AWS CodePipeline?**

In today’s fast-paced software development world, frequent and reliable application releases are crucial to staying competitive. Manual deployments can be time-consuming, error-prone, and hinder your ability to deliver new features and bug fixes quickly. This is where **Continuous Integration and Continuous Delivery (CI/CD)** pipelines come in.

By automating critical stages of the development lifecycle, CI/CD pipelines enable you to **streamline deployments, improve software quality, and decrease time to market**.

# Introducing the Orchestrators of Our CI/CD Pipeline

To achieve automated and efficient application deployments, we will leverage the power of several **AWS services**:

* [**AWS CodeCommit**](https://docs.aws.amazon.com/codecommit/latest/userguide/welcome.html)
* [**AWS CodeBuild**](https://docs.aws.amazon.com/codebuild/latest/userguide/welcome.html)
* [**AWS CodeDeploy**](https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html)
* [**AWS CodePipeline**](https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html)

These four services, working together in harmony within our CI/CD pipeline, will empower us to automate manual tasks, accelerate deployments, and deliver our applications faster and more reliably.

*.*

# Step 1: Set up CodeCommit Repository

* Open the AWS Management Console and navigate to CodeCommit.
* Create a new repository and note down its clone URL.

**Now you need to setup Git Credentials in your AWS IAM**

* Go to IAM console
* Click on Users in the left-hand menu, and then click on your username.
* Add permission for git access for IAM user
* Search and select “AWSCodeCommitFullAccess” and “AWSCodeCommitPowerUser”
* Click Add Permissions button to add the policies to the user
* Select **Security credentials** section and scroll down to  
  “HTTPS Git credentials for AWS CodeCommit” section, then click on “Generate credentials”.
* Git credentials is created.
* Open VS Code terminal and clone the Codecommit Repository (Demo-Repository)

# Step 2: Set up CodeBuild Project

* Before creating a CodeBuild project, let’s create the **index.html**and **buildspec.yml** files, which we’ll need in the next step.

<!-- index.html -->  
  
<!DOCTYPE html>  
<html lang="en">  
  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>My Website</title>  
 <style>  
 body {  
 background-color: #3498db; /\* Set your desired background color \*/  
 color: #ffffff; /\* Set your desired text color \*/  
 text-align: center;  
 padding: 100px; /\* Add padding for better visibility \*/  
 }  
  
 h1 {  
 font-size: 2em;  
 }  
  
 p {  
 font-size: 1.2em;  
 }  
 </style>  
</head>  
  
<body>  
 <h1>Welcome to my website</h1>  
 <p>This is a simple website hosted on AWS CodeCommit.</p>  
</body>  
  
</html>

# buildspec.yml  
  
version: 0.2  
  
phases:  
 install:  
 commands:  
 - echo Installing NGINX - echo apt-get install NGINX  
 - sudo apt-get update  
 - sudo apt-get install nginx -y  
 build:  
 commands:  
 - echo Build started on date  
 - cp index.html /var/www/html  
 post\_build:  
 commands:  
 - echo Configuring NGINX  
artifacts:  
 files:  
 - /var/www/html/index.html

# buildspec.yml  
  
version: 0.2  
  
phases:  
 install:  
 commands:  
 - echo Installing NGINX - echo apt-get install NGINX  
 - sudo apt-get update  
 - sudo apt-get install nginx -y  
 build:  
 commands:  
 - echo Build started on date  
 - cp index.html /var/www/html  
 post\_build:  
 commands:  
 - echo Configuring NGINX  
artifacts:  
 files:  
 - /var/www/html/index.html

**Below are the functions of each build step:**

* The version of the **Buildspec** syntax we’re using is stated as version: **0.2.**
* **Phases:** This lists the stages of construction for our project.
* **install:** utilizes the apt-get package manager to install nginx on the built environment.
* **build** Places a copy of the index.html file in the nginx default web root directory.
* **post build:** If additional nginx settings are required, it is done.
* **artifacts:**Indicates where the index.html file should be included in the **build artifact.**

**Save these two files and push them to the CodeCommit repository.**

# ****Create a Code Build Project:****

* Navigate to Code Build in the AWS Management Console.
* Create a new build project, specifying the source as Code Commit and configuring the build settings according to your project requirements.
* Provide the environment details on which operating system the code needed to be run with the latest image.
* Select Use build spec file in Buildspec section
* Select ‘Amazon S3’ in the Artifacts section and specify the S3 bucket where artifacts will be stored.
* Now Click on “Create build Project”
* verify the project build details, and then initiate the CodeBuild process.
* The CodeBuild process has been completed; now, verify the build phases.
* All build phases have been successfully completed.
* All build phases have been successfully completed.

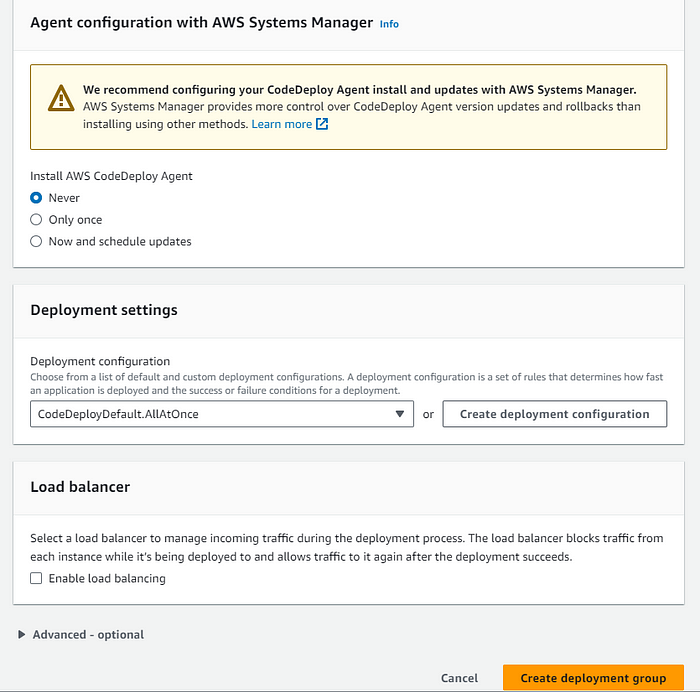
# Step 3: Set up CodeDeploy Application and Deployment Group

* Before creating a CodeDeploy application, let’s first create an EC2 instance, which we will need in the next steps.
* Navigate to the EC2 console and click on “Launch instances.”
* Enter Name for EC2 Instance and Select **UBUNTU**in AMI section
* Choose the “t2.micro” instance type, which is eligible for the Free Tier.
* In the Key Pair section, choose an existing key pair or create a new one.
* Keep the network settings and storage details as default.
* Expand **Advanced details**and scroll down to User data
* Enter the following commands in User Data to install the CodeDeploy Agent and AWS Systems Manager (SSM) Agent on the server, then click on Launch Instance

#!/bin/bash  
apt-get update  
apt-get install -y ruby  
apt-get install -y wget  
cd /home/ubuntu  
wget https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/install  
chmod +x ./install  
./install auto  
  
# For SSM agent  
apt-get install -y amazon-ssm-agent  
systemctl start amazon-ssm-agent  
systemctl enable amazon-ssm-agent

# Now, let’s navigate to the CodeDeploy service

* Create an application by providing the name and compute platform. Here we are using EC2 instance.
* The application is created now. Now create a deployment group.
* Before that, we need to give access to deployment to access the AWS resource for which we need to create a role in IAM. Provide all the necessary access as mentioned below.
* Edit the trust policy and add Statement for CodeDeploy and update policy.
* Also create a role for giving access to EC2 instance with all the neccessary permission policies as shown below.
* Now, navigate to the instance and modify the IAM role. Select the IAM role created above.
* Now, create a deployment group by specifying the deployment group name and service role. Enter the ARN (Amazon Resource Name) from the previously updated IAM role (code-deployment-service-role).
* Provide the EC2 instance details in Environment configuration.
* Provide the below configurations. Now, Create deployment Group.



* A deployment group is now created.
* Add appspec.yaml file to CodeCommit Repository and complete the deployment process.
* An AppSpec file is required to establish a connection between AWS CodeDeploy and an EC2 instance. Create the YAML file.

# appspec.yml  
  
version: 0.0 # specifies the version number of the file format.  
os: linux # specifies the operating system to be used.  
files: # is an array of files to be copied from the source to the destination directory.  
 - source: / # specifies the source directory.  
 destination: /var/www/html # specifies the destination directory where the files will be copied to.  
hooks: # is an array of to be executed at different points during the deployment process.  
 AfterInstall:   
 - location: /install\_nginx.sh   
 timeout: 300 # specifies the maximum amount of time in seconds that the script can run for.  
 runas: root # specifies the user that the script should be run as.  
 ApplicationStart: # specifies a script to be executed after the application has started.  
 - location: /start\_nginx.sh  
 timeout: 300  
 runas: root

* Create the dependency files as well for installing and starting nginx on the server.
* **install\_nginx.sh**

#!/bin/bash  
  
sudo apt-get update && sudo apt-get install -y nginx

* **start\_nginx.sh**

#!/bin/bash  
  
sudo systemctl start nginx  
sudo systemctl enable nginx

* Make sure to change the buildspec.yaml file so that the CodeBuild will build the appspec.yml file and transfer the artifact to S3 bucket.

# buildspec.yml  
  
version: 0.2  
  
phases:  
 install:  
 commands:  
 - echo Installing NGINX - echo apt-get install NGINX  
 - sudo apt-get update  
 - sudo apt-get install nginx -y  
 build:  
 commands:  
 - echo Build started on date  
 - cp index.html /var/www/html  
 post\_build:  
 commands:  
 - echo Configuring NGINX  
artifacts:  
 files:  
 - '\*\*/\*'

Here is the [**GitHub repository**](https://github.com/bvskarthik18/CI-CD) where all the required files are stored.

* Commit the code and transfer all to the CodeCommit repository.
* You can view the new code files in the CodeCommit repository.

**Step 4: Create CodePipeline**

* Navigate to CodePipeline in the AWS Management Console create a new pipeline.
* Provide the Pipeline naming details and let the service role to be default.
* Provide the code repository details. In our case, the code is located in the “Demo-Repository” folder within the AWS CodeCommit repository. Choose AWS CodePipeline to automatically trigger the pipeline in response to any changes in the code.
* Now add the Build stage details. Select the build provider and give the project name.
* Now add the deploy stage. Select the provider and name of the already created application and deployment group.
* Now verify the details and create the code pipeline.
* The pipeline will now fetch the code from the CodeCommit repository.
* Then, it will build the code and deploy in the server as we can see in the below screenshot.
* Now, navigate to the public URL of the instance and check the Webpage.
* We have successfully built the pipeline. Let’s witness the magic. Let’s modify our code and view the pipeline.
* I have changed my index.html file and committed it in the CodeCommit repository.
* The pipeline has automatically picked up the code from the repository, built and deployed it in the server.