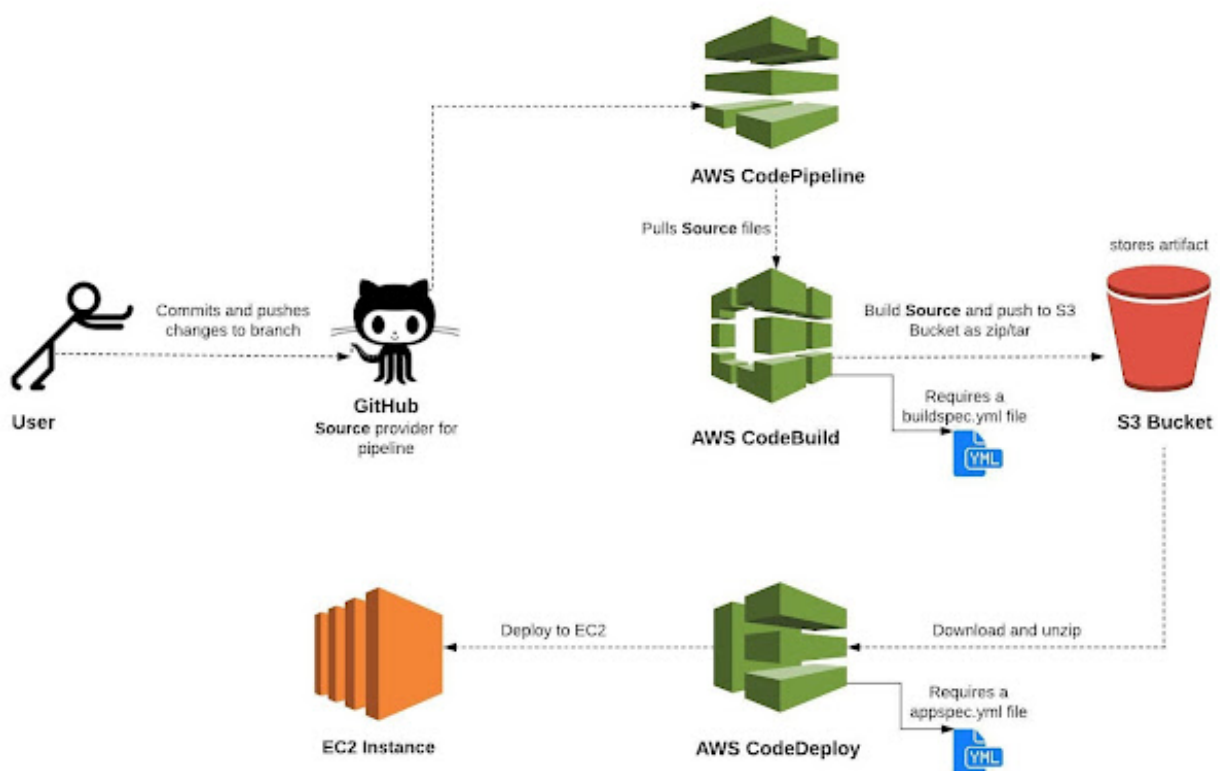


# Deploy a ReactJS Application to AWS EC2 Instance using AWS CodePipeline

AWS CodePipeline Notes ( Topics Covered : CodeBuild,CodeDeploy and Deployment of an App

**AWS CodePipeline** is a continuous integration and continuous delivery (CI/CD) AWS service that allows you to automate the release process for your application or service. Every time you commit a code change to your source(GitHub, AWS CodeCommit, etc), CodePipeline automatically builds, tests, and deploys your code based on the release process models you define while initializing your CodePipeline. This enables you to rapidly and reliably deliver features and updates.



Deploy a ReactJS Application to AWS EC2 Instance using AWS CodePipeline

## Steps that we will follow:

1. Create IAM Role for EC2 & AWS CodeDeploy
2. Launch an EC2 instance & then will attach that EC2 Role.
3. Create a CodePipeline using Github, CodeBuild and CodeDeploy

Abdur Razzaq  
Linkedin :: @abdurrazzaq04

## Step 1) Create IAM Role for EC2 and AWS CodeDeploy

Create a new **role for EC2** and attach **AmazonS3ReadOnlyAccess** policy which will allow our EC2 instance to access stored **artifacts** from the Amazon S3 bucket.

IAM > Roles > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

### Select trusted entity Info

Trusted entity type

- ☒ **AWS service**  
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ☐ **AWS account**  
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ☐ **Web identity**  
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ☐ **SAML 2.0 federation**  
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ☐ **Custom trust policy**  
Create a custom trust policy to enable others to perform actions in this account.

Use case  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

- ☒ **EC2**  
Allows EC2 instances to call AWS services on your behalf.
- ☐ **Lambda**  
Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:  
Choose a service to view use case

Cancel **Next**

IAM > Roles > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

### Name, review, and create

#### Role details

Role name  
Enter a meaningful name to identify this role.

EC2RoleForS3

Maximum 64 characters. Use alphanumeric and '+=, @-\_' characters.

Description  
Add a short explanation for this role.

Allows EC2 instances to call AWS services on your behalf.  
Role name : EC2RoleForS3  
Attached Permission : AmazonS3ReadOnlyAccess

Maximum 1000 characters. Use alphanumeric and '+=, @-\_' characters.

Create a **new service role for CodeDeploy** and attach **AWSCodeDeployRole** policy which will provide the permissions for our service role to read tags of our EC2 instance, publish information to Amazon SNS topics and much more task.

IAM > Roles > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

### Select trusted entity info

Trusted entity type

- ☒ **AWS service**  
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ☐ AWS account  
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ☐ Web identity  
Allow users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ☐ SAML 2.0 federation  
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ☐ Custom trust policy  
Create a custom trust policy to enable others to perform actions in this account.

Use case  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

- ☐ EC2  
Allows EC2 instances to call AWS services on your behalf.
- ☐ Lambda  
Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

- ☒ **CodeDeploy**  
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.
- ☐ CodeDeploy for Lambda  
Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.
- ☐ CodeDeploy - ECS  
Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to IAM topics, and update ECS services on your behalf.

Cancel **Next**

IAM > Roles > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

## Name, review, and create

### Role details

#### Role name

Enter a meaningful name to identify this role.

CodeDeployRole

Maximum 64 characters. Use alphanumeric and '+=, @-\_' characters.

#### Description

Add a short explanation for this role.

Allows EC2 instances to call AWS services on your behalf.

Role name : CodeDeployRole

Attached Permission : AWSCodeDeployRole

Maximum 1000 characters. Use alphanumeric and '+=, @-\_' characters.

## Step 2) Launch an Linux EC2 instance

**Note:** Please Select `t3.medium` Instance Type

And then Install code depoloy agent by follwing commands on terminal.

### For Centos & Amazon Linux 2

```
sudo yum update
sudo yum install ruby
sudo yum install wget
wget https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto
sudo service codedeploy-agent status
```

## For Ubuntu & Debian Based

```
sudo apt-get update
```

```
sudo apt-get install ruby
```

```
sudo apt-get install wget
```

```
cd /home/ubuntu
```

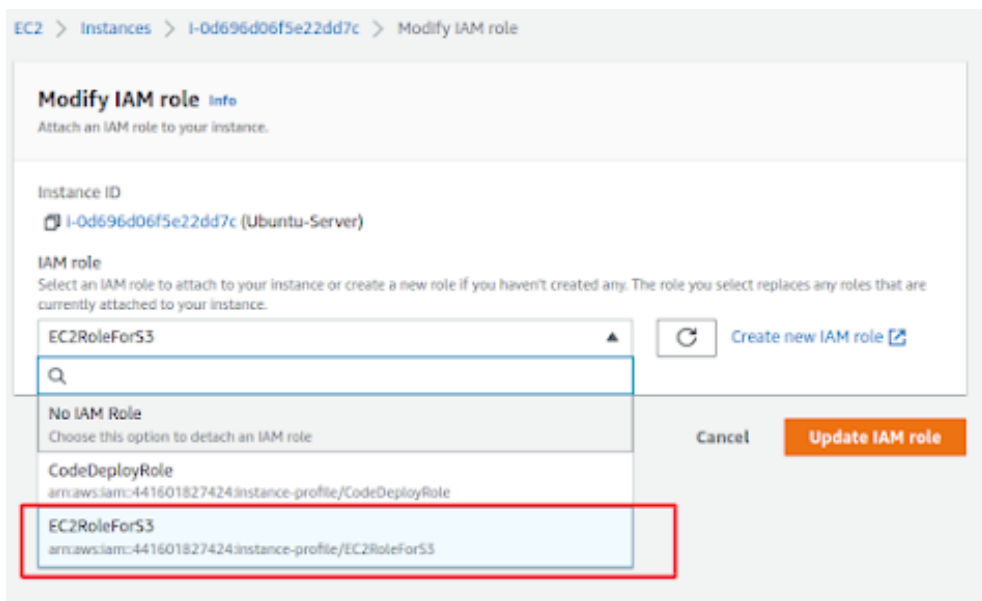
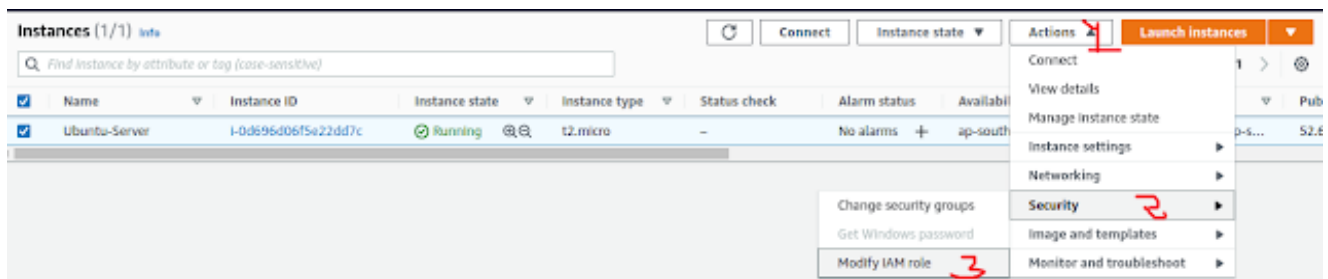
```
wget https://aws-codedeploy-ca-central-1.s3.ca-central-1.amazonaws.com/latest/install
```

```
chmod +x ./install
```

```
sudo ./install auto
```

```
sudo service codedeploy-agent status
```

## Now, Let's Attach that EC2 Role with it.



**Note :** Let's Make a Github Repository and download the following source code and push on it.

## Download-SourceCode

# SourceCode File Structure

```
README.md
package-lock.json
package.json
public/
src/

buildspec.yml
appspec.yml
scripts/
  after_install.sh
  app_start.sh
  before_install.sh
server.js
```

----Build File for AWS CodeBuild

----Information About Server & Hooks

appspec.yml >> file for AWS CodeDeploy

## Step 3) Create a CodePipeline using Github, CodeBuild and CodeDeploy

### a) Create CodePipeline

Let's navigate to CodePipeline via AWS Management Console and click on Create pipeline:

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1: Choose pipeline settings

Step 2: Add source stage

Step 3: Add build stage

Step 4: Add deploy stage

Step 5: Review

### Choose pipeline settings

**Pipeline settings**

**Pipeline name**  
Enter the pipeline name. You cannot edit the pipeline name after it is created.  
React-Pipeline  
No more than 100 characters

**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-southeast-2-React-Pipeline  
Type your service role name

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

► Advanced settings

Cancel Next

### b) Choose Github in Code Source

After selecting GitHub as the source provider, click on the Connect to GitHub button. You'll then be prompt to enter your GitHub login credentials

Once you grant AWS CodePipeline access to your GitHub repository, you can select a repository and branch for CodePipeline to upload commits to this repository to your pipeline

Step 2

Add source stage

Step 3

Add build stage

Step 4

Add deploy stage

Step 5

Review

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.
 

× or Connect to GitHub

**Ready to connect**

Your GitHub connection is ready for use.

Repository name

Choose a repository in your GitHub account.
 

×

 <account>/<repository-name>

Branch name

Choose a branch of the repository.
 

×

Change detection options

☒ **Start the pipeline on source code change**

Automatically starts your pipeline when a change occurs in the source code. If turned off, your pipeline only runs if you start it manually or on a schedule.

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.

Cancel

Previous

Next

### c) Configure CodeBuild (Optional)

If you haven't created a project prior to creating your pipeline, then you can create a project directly from here by clicking Create project button.

6/9

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1  
Choose pipeline settings

Step 2  
Add source stage

Step 3  
**Add build stage**

Step 4  
Add deploy stage

Step 5  
Review

## Add build stage info

### Build - optional

**Build provider**  
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

**Region**  
Asia Pacific (Sydney)

**Project name**  
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

React-EC2-Build-Project or [Create project](#)

Successfully created React-EC2-Build-Project in CodeBuild.

**Environment variables - optional**  
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

[Add environment variable](#)

**Build type**

☒ **Single build**  
Triggers a single build.

☐ **Batch build**  
Triggers multiple builds as a single execution.

Cancel Previous Skip build stage Next

**Note:** Buildspec file is a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build. For my project, I created a buildspec.yaml file and added it in the root of my project directory:

#### d) Add Depoly Stage

Note : Before going to configure Add Depoly Stage, Let's make duplicate tab of current tab.

and then go to code deploy in the navigation, Select Application, then add create a deployment group.

Developer Tools

**CodeDeploy**

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy**
  - Getting started
  - Deployments
  - Applications 1
  - Deployment configurations
  - On-premises instances
- Pipeline • CodePipeline
- Settings

Developer Tools > CodeDeploy > Applications > Create application

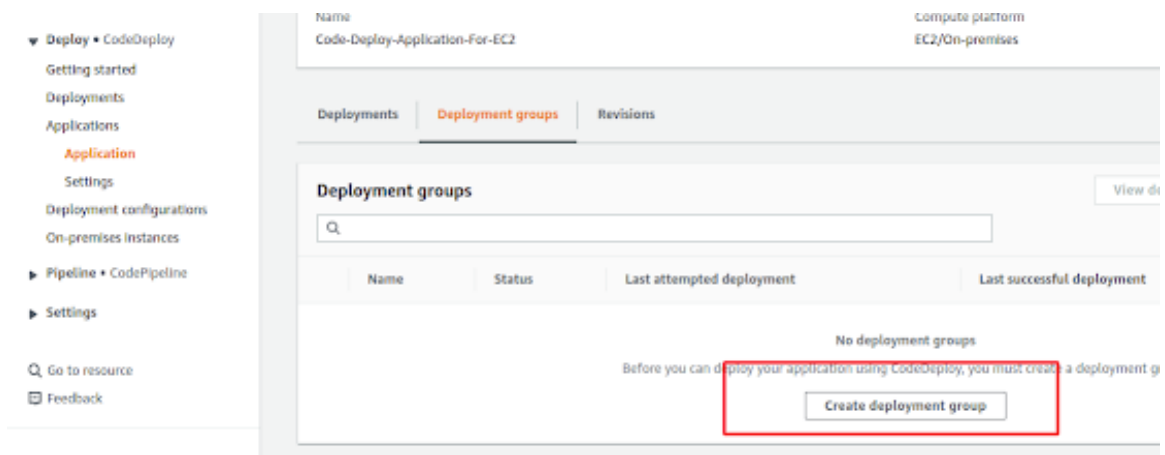
## Create application

### Application configuration

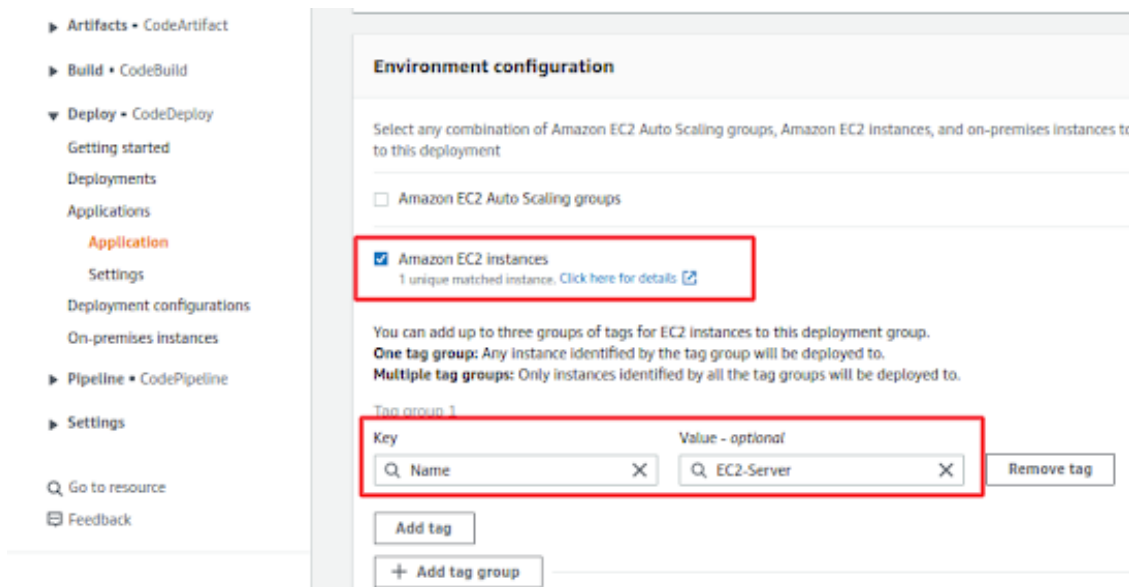
**Application name** 2  
Enter an application name  
Code-Deploy-Application-For-EC2  
100 character limit

**Compute platform** 3  
Choose a compute platform  
EC2/On-premises

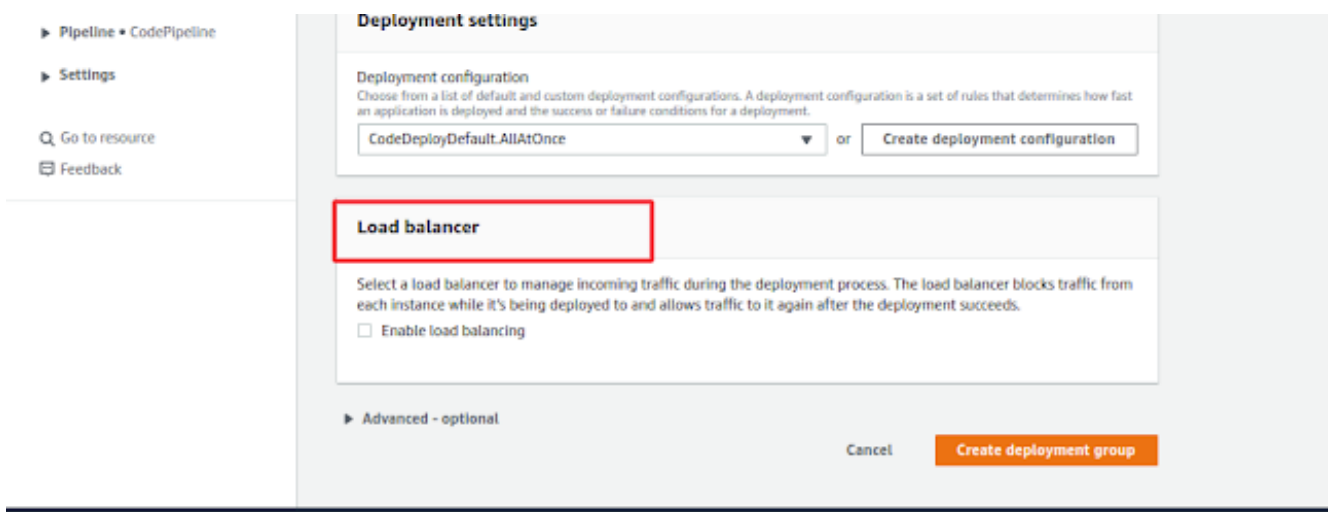
Cancel Create application



In deployment group Select EC2 instances and select Tag and Value



Untick Load Balancer Option



and Finally Come on **Add Deploy Stage** and select that **created Application** name & **Deployment group**



**e) At the end, just review and create.**

**If You will Get below error in pipeline then follow further instruction to resolve it.**

**Error** : CodeDeploy agent was not able to receive the lifecycle event. Check the CodeDeploy agent logs on your host and make sure the agent is running and can connect to the CodeDeploy server.

**Conclusion:** Its happening Because we installed codedeployment-agent before attaching ec2role to EC2 Instance, and error will be resolve when you will first attach ec2role and then install codedeployment-agent on EC2.

**Run the following commands.**

```
sudo systemctl stop codedeploy-agent  
sudo yum erase codedeploy-agent -y  
sudo rm -rf /opt/codedeploy-agent  
cd /home/ec2-user/server  
sudo ./install auto  
sudo systemctl status codedeploy-agent
```