

## Blue/Green Deployment with AWS Services

For the deployment, we need to create 2 IAM roles for EC2 and CodeDeploy respectively.

### IAM Role Creation

#### For EC2 Server

- Go to the AWS Console
- IAM > Role > Create Role
- Choose a use case: EC2
- Select "AmazonEC2RoleforAWSCodeDeploy"
- Give a Name "EC2CodeDeployRole"
- Click on the "Create" Button
- Now select the "EC2CodeDeployRole" from Roles and goto the "Trust Relationships" >> click on "Edit trust relationships".
- Add the following line in the services,  
"codedeploy.us-east-2.amazonaws.com",  
"codedeploy.<region-name>.amazonaws.com"
- your policy will look like this,

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Service": [  
          "codedeploy.us-east-2.amazonaws.com",  
          "ec2.amazonaws.com"  
        ]  
      },  
      "Action": "sts:AssumeRole"  
    }  
  ]  
}
```

- Click on "Update Trust Policy"
- Additionally if we want to access the S3 bucket, we must add a following inline policy to this role.
- Click on IAM Role > Create Permission > Create Inline Policy and paste the following content with the change in S3 ARN.
- Give policy name "AutoScale Policy".

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "s3:ListStorageLensConfigurations",
        "s3:ListAccessPointsForObjectLambda",
        "s3:GetAccessPoint",
        "s3:PutAccountPublicAccessBlock",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAllMyBuckets",
        "s3:ListAccessPoints",
        "s3:PutAccessPointPublicAccessBlock",
        "s3:ListJobs",
        "s3:PutStorageLensConfiguration",
        "s3:ListMultiRegionAccessPoints",
        "s3:CreateJob"
      ],
      "Resource": "*"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "<S3 ARN>",
        "<S3 ARN/*>"
      ]
    }
  ]
}

```

### For CodeDeploy

- Go to the AWS Console
- IAM > Role > Create Role
- Choose a use case: Codedeploy
- Select "AWSCodeDeployRole"
- Give a Name "CodeDeployRole"

- Click on the "Create" Button.
- Additionally, we need to add the following inline policy to the IAM Role for autoscaling.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole",
        "ec2:RunInstances",
        "ec2:CreateTags",
        "s3:ListStorageLensConfigurations",
        "s3:ListAccessPointsForObjectLambda",
        "s3:GetAccessPoint",
        "s3:PutAccountPublicAccessBlock",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAllMyBuckets",
        "s3:ListAccessPoints",
        "s3:PutAccessPointPublicAccessBlock",
        "s3:ListJobs",
        "s3:PutStorageLensConfiguration",
        "s3:ListMultiRegionAccessPoints",
        "s3:CreateJob"
      ],
      "Resource": "*"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "<S3 ARN>",
        "<S3 ARN/*>"
      ]
    }
  ]
}
```

## Role attach to the Launch template

- Go to the launch template console
- Modify the launch template
- At last, we can find the **“Advance details”** section. Find the **“IAM Instance Profile”** and select the **“EC2CodeDeployRole”**.
- Create an autoscaling group as per your convenience.

## CodeDeploy Setup

- Need to create an Application and Deployment group for codedeploy.
- Go to the Codedeploy console and from the left side panel select **Application**.
- Click on the **“Create Application”** button.
- Give a suitable name (eg: codedeploy\_<project-name>)and for **“Compute Platform”** select **“EC2/On-premises”**.
- Now we need to create a deployment group under the application which we had created.
- Select the codedeploy application which we have created earlier and select **“create deployment group”**.

The screenshot shows the 'Create Application' form in the AWS CodeDeploy console. It is divided into four main sections: 'Application', 'Deployment group name', 'Service role', and 'Deployment type'. The 'Application' section shows the application name 'codedeploy\_prod\_dwisi' and the compute type 'EC2/On-premises'. The 'Deployment group name' section has a text input field with a 100 character limit. The 'Service role' section has a search input field for selecting a service role. The 'Deployment type' section has two radio button options: 'In-place' (selected) and 'Blue/green'. The 'In-place' option description states it updates instances in the deployment group with the latest application revisions, taking them offline briefly. The 'Blue/green' option description states it replaces instances with new ones and deploys the latest application revision, registering them with a load balancer and deregistering the original ones.

Application	
Application	
codedeploy_prod_dwisi	
Compute type	
EC2/On-premises	

Deployment group name
Enter a deployment group name
<input type="text"/>
100 character limit

Service role
Enter a service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
<input type="text"/>

Deployment type
Choose how to deploy your application
<input checked="" type="radio"/> In-place
Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update
<input type="radio"/> Blue/green
Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

- Give a suitable name to the deployment group
- Select the **“CodeDeployRole”** IAM role under the **“Service Role”**
- For Deployment Type select **“Blue/Green”**.

### Environment configuration

Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment

☐ Amazon EC2 Auto Scaling groups

☐ Amazon EC2 instances

☐ On-premises instances

### Deployment settings

Deployment configuration

Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.

CodeDeployDefault.AllAtOnce
or
Create deployment configuration

### Load balancer

Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed to and allows traffic to it again after the deployment succeeds.

☒ Enable load balancing

☒ Application Load Balancer or Network Load Balancer
☐ Classic Load Balancer

Choose a target group

► Advanced - optional

Cancel
Create deployment group

- Under the Environment configuration Select "Amazon EC2 Auto Scaling groups" and select the respective autoscaling group from the drop-down list.
- Under Deployment settings select the appropriate settings for your project.
- Under Load Balancer select the "Application/Network Load balancer" and choose the respective target group.
- Click on the create deployment group.

## CodePipeline Setup

- On AWS go to the “Developer Tools Settings” and add the connection got GitHub.
- Go to the Code Pipeline section and create the pipeline

The screenshot shows the 'Choose pipeline settings' page in the AWS CodePipeline console. The breadcrumb trail at the top reads: Developer Tools > CodePipeline > Pipelines > Create new pipeline. On the left, a sidebar lists five steps: Step 1 (Choose pipeline settings, highlighted), 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' with an 'Info' link. It contains a 'Pipeline settings' section with the following fields: 'Pipeline name' (a text box with placeholder '<pipeline-name>' and a note 'Enter the pipeline name. You cannot edit the pipeline name after it is created.' and 'No more than 100 characters'), 'Service role' (two radio buttons: 'New service role' (selected) with subtext 'Create a service role in your account', and 'Existing service role' with subtext 'Choose an existing service role from your account'), and 'Role name' (a text box with placeholder 'AWSCodePipelineServiceRole-eu-west-2-<pipeline-name>' and a note 'Type your service role name'). Below these is a checkbox labeled 'Allow AWS CodePipeline to create a service role so it can be used with this new pipeline', which is checked. At the bottom of the settings section is a link '► Advanced settings'. At the bottom right of the page are 'Cancel' and 'Next' buttons.

- Give suitable name and select “New Service Role” for service role as shown in the image above.
- Select the checkbox “Allow AWS CodePipeline to create a service role so it can be used with this new pipeline” and leave other things as default.
- Click on next

- Select GitHub version 2 as a source provider and fill in the remaining fields as shown in the following image.

The screenshot shows the 'Add source stage' configuration page in the AWS CodePipeline console. The left sidebar contains navigation links for Developer Tools, CodePipeline, and various providers like CodeCommit, CodeArtifact, CodeBuild, CodeDeploy, and CodePipeline. The main panel is titled 'Source' and contains the following sections:

- Source provider:** A dropdown menu with 'GitHub (Version 2)' selected.
- New GitHub version 2 (app-based) action:** A blue box with an information icon and text: '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:** A section with the text 'Choose an existing connection that you have already configured, or create a new one and then return to this task.' It includes a search input field and a 'Connect to GitHub' button.
- Repository name:** A section with the text 'Choose a repository in your GitHub account.' It includes a search input field and a placeholder '<account>/<repository-name>'.
- Branch name:** A section with the text 'Choose a branch of the repository.' It includes a search input field.
- Change detection options:** A section with the text 'Start the pipeline on source code change' checked. Below it, a description: '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:** A section with the text 'Choose the output artifact format.' It includes two radio buttons: 'CodePipeline default' (selected) and 'Full clone'.

At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Next'.

- Click on the next button
- Skip the “Build Provider stage”.
- Click on the next button.

- Select “AWS Codedeploy” under the deploy tab.
- Fill the remaining fields as per the configurations.

The screenshot shows the AWS CodePipeline console interface. On the left is a navigation sidebar with the following items: 'Developer Tools' (with a close icon), 'CodePipeline', 'Source • CodeCommit', 'Artifacts • CodeArtifact', 'Build • CodeBuild', 'Deploy • CodeDeploy', 'Pipeline • CodePipeline' (expanded), 'Getting started', 'Pipelines', and 'Settings'. Below these are search and feedback links: 'Go to resource' and 'Feedback'. The main content area has a breadcrumb trail: 'Developer Tools > CodePipeline > Pipelines > Create new pipeline'. A vertical list of steps is on the left: 'Step 1: Choose pipeline settings', 'Step 2: Add source stage', 'Step 3: Add build stage', 'Step 4: Add deploy stage' (highlighted), 'Step 5: Review', and 'Review'. The main panel is titled 'Add deploy stage' with an 'Info' link. A blue information box 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' configuration form includes: 'Deploy provider' (dropdown set to 'AWS CodeDeploy'), 'Region' (dropdown set to 'Europe (London)'), 'Application name' (text input with a search icon), and 'Deployment group' (text input with a search icon). At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

- Click on “Review”.
- Review the configuration and create the pipeline configuration.
- We can see the pipeline is running under the codepipeline section.