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.

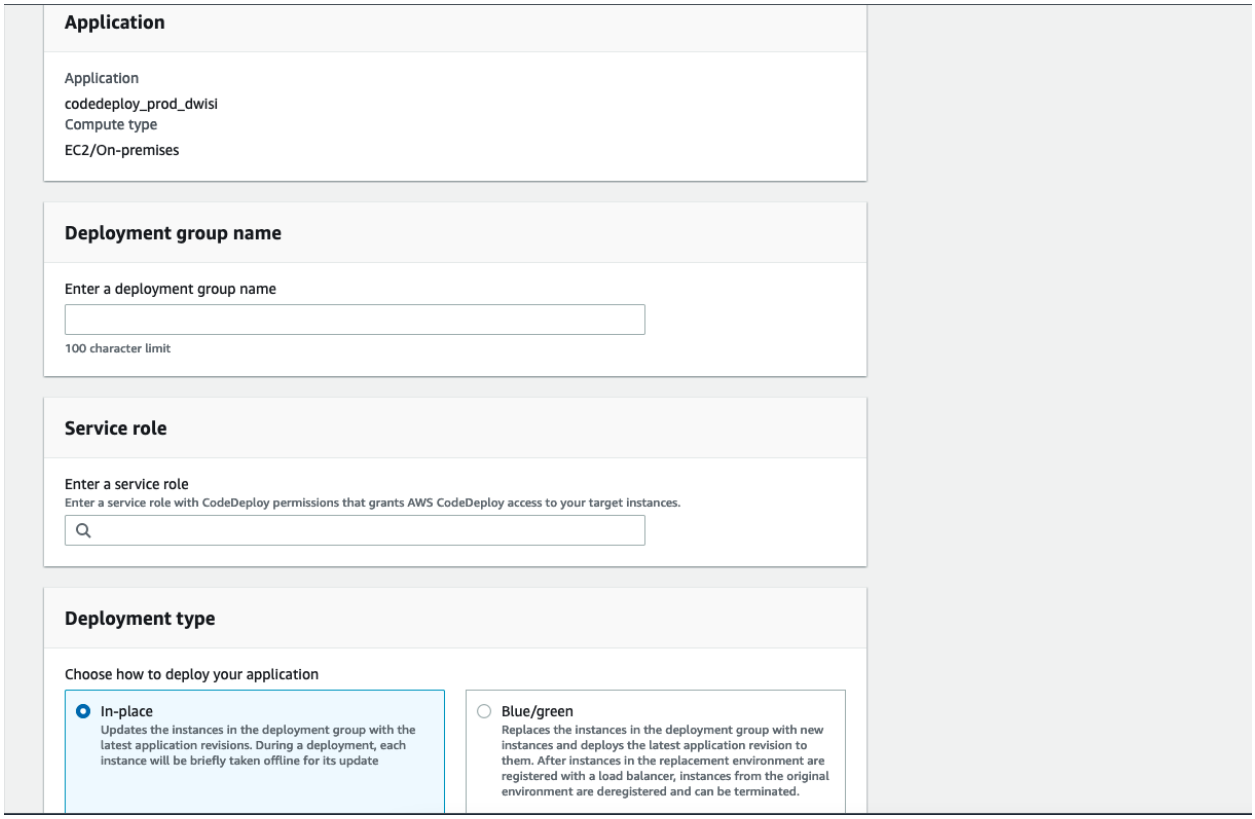
```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "VisualEditor0",
           "Effect": "Allow",
           "Action": [
               "iam:PassRole",
               "ec2:RunInstances",
               "ec2:CreateTags",
               "s3:ListStorageLensConfigurations",
               "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

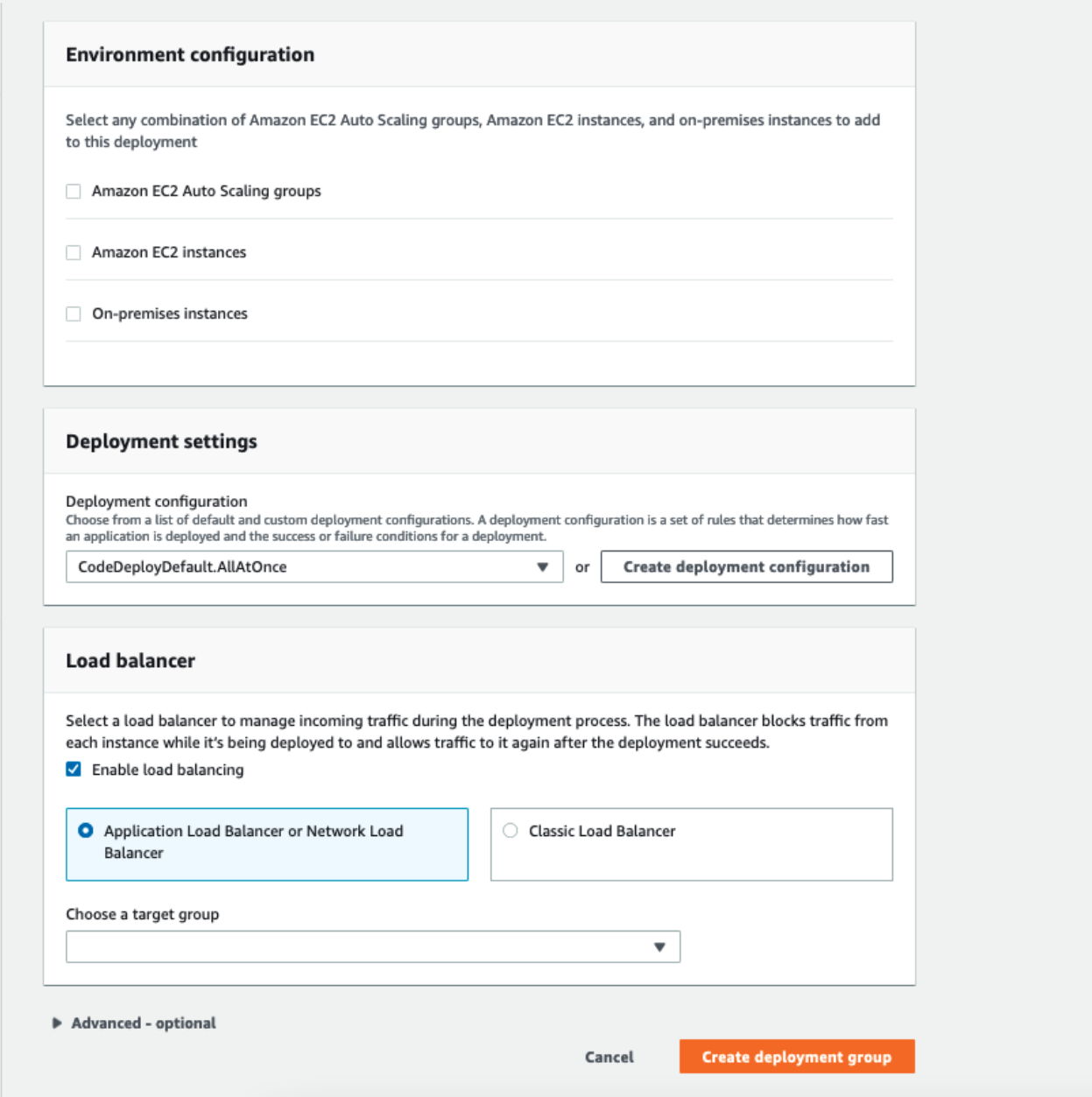
- Go to the launch template console
- Modify the launch template
- At last, we can find the **“Advance details”** section. Find the **“IAM Instance Profile”** and select the “EC2CodeDeployRole”.
- Create an autoscaling group as per your convenience.

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”.



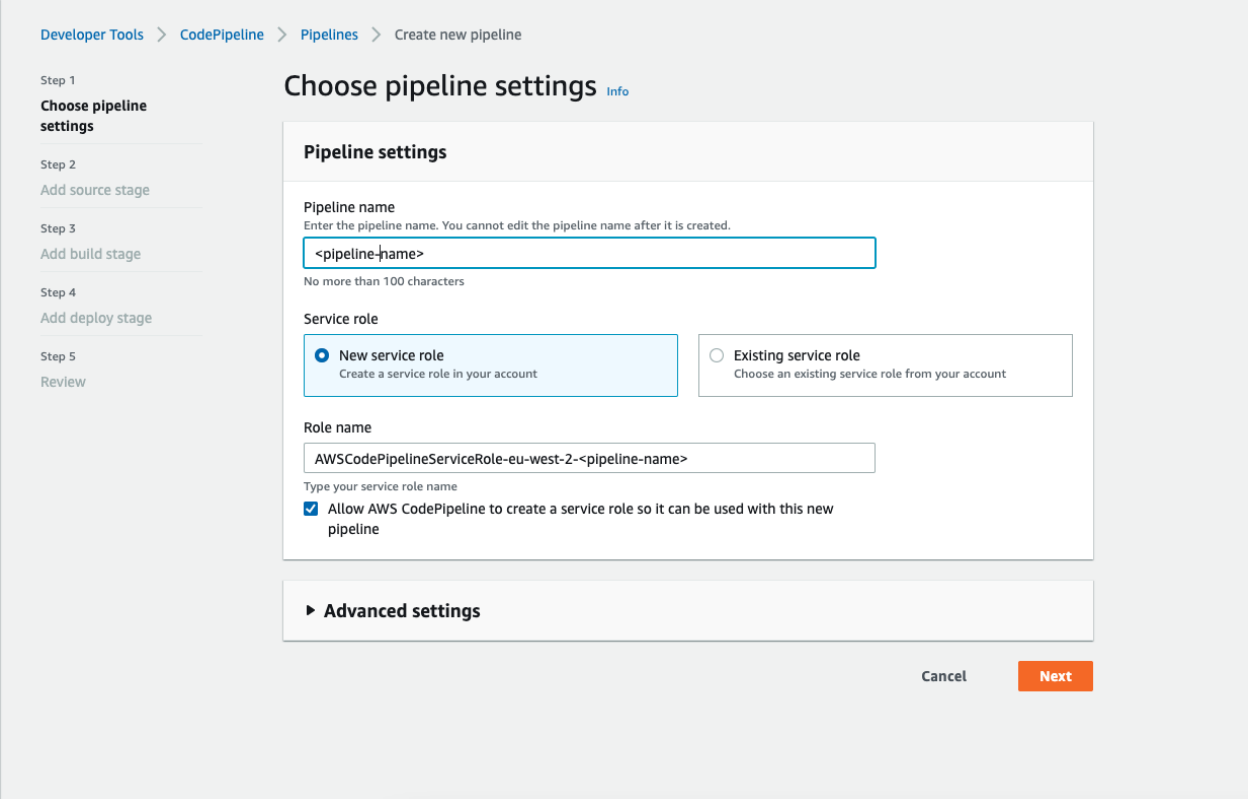
- Give a suitable name to the deployment group
- Select the “`CodeDeployRole`” IAM role under the “Service Role”
- For Deployment Type select “Blue/Green”.



- 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.

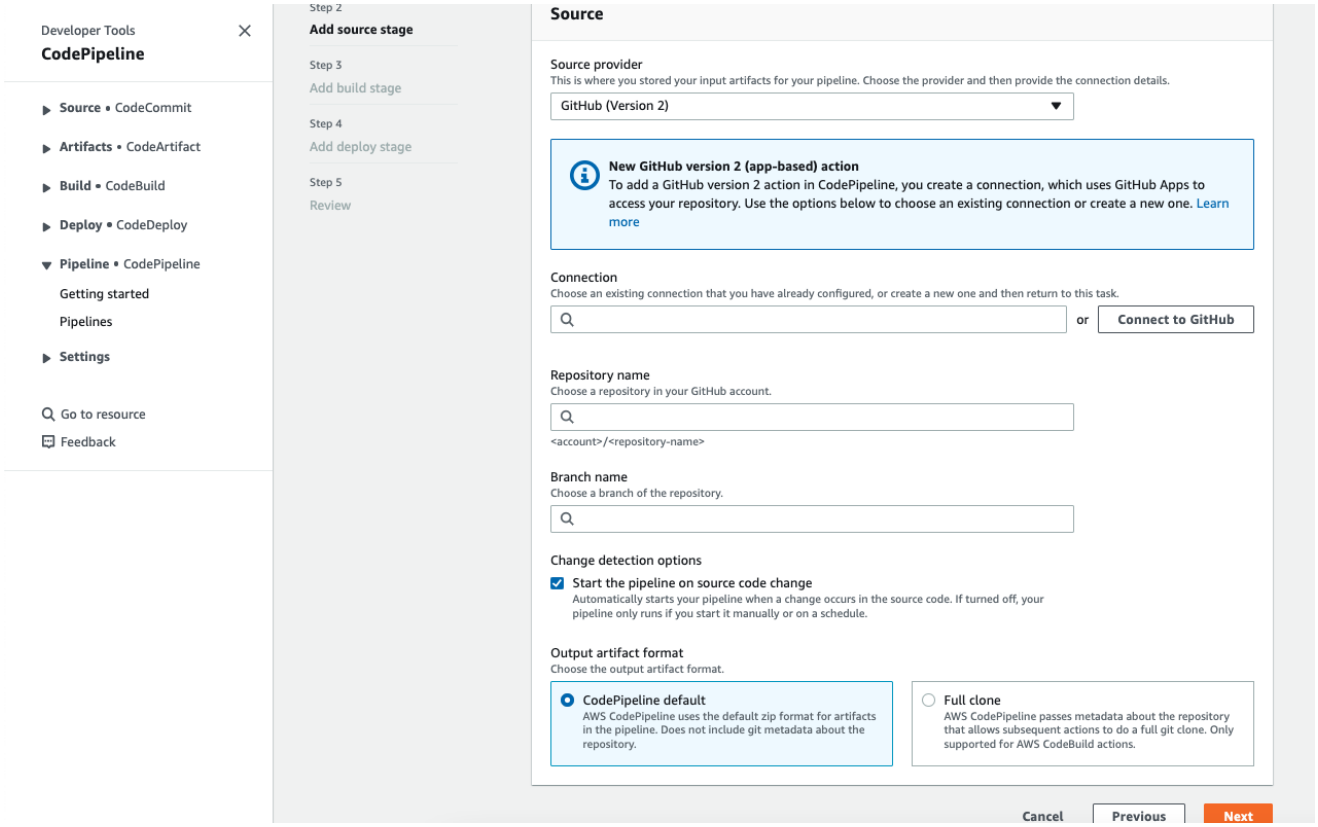
CodePipeline Setup

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



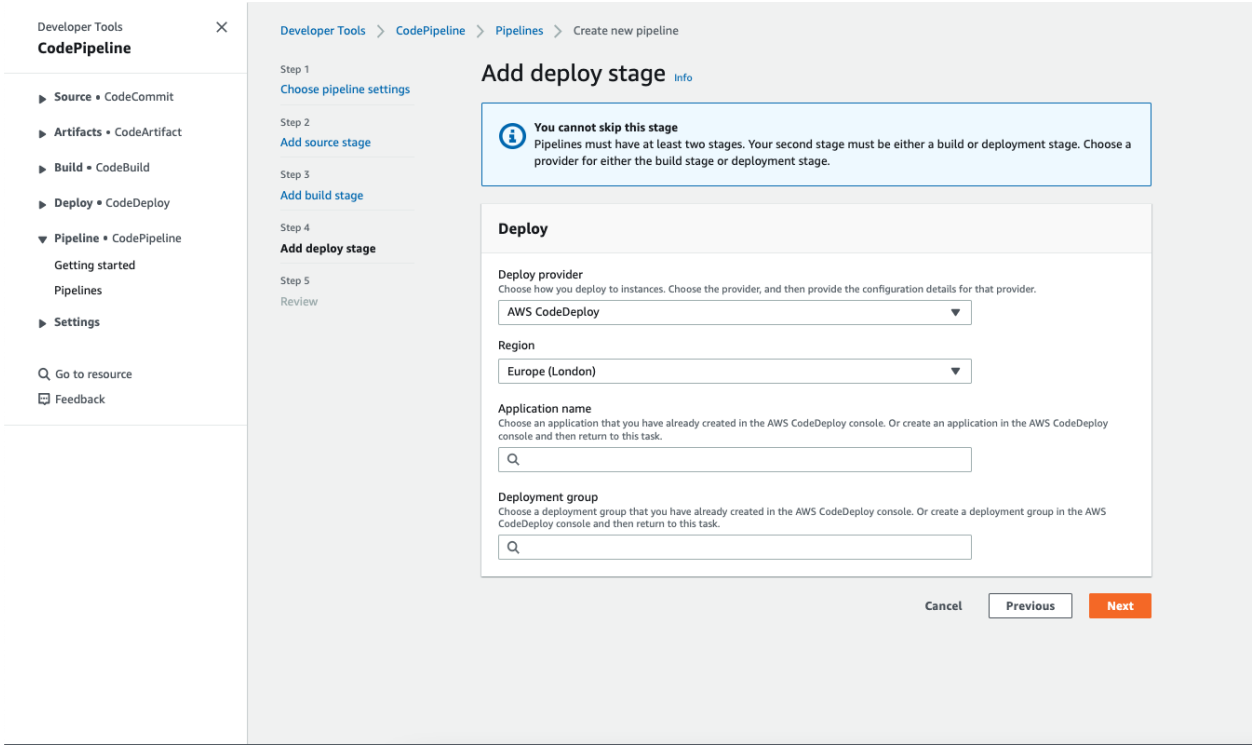
- Give suitable name and select “New Service Role” for service role as shown in the image above.
- Select the checkbox “Allow AWS CodePipeline to create a service role so it can be used with this new pipeline” and leave other things as default.
- Click on next

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



- 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.



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