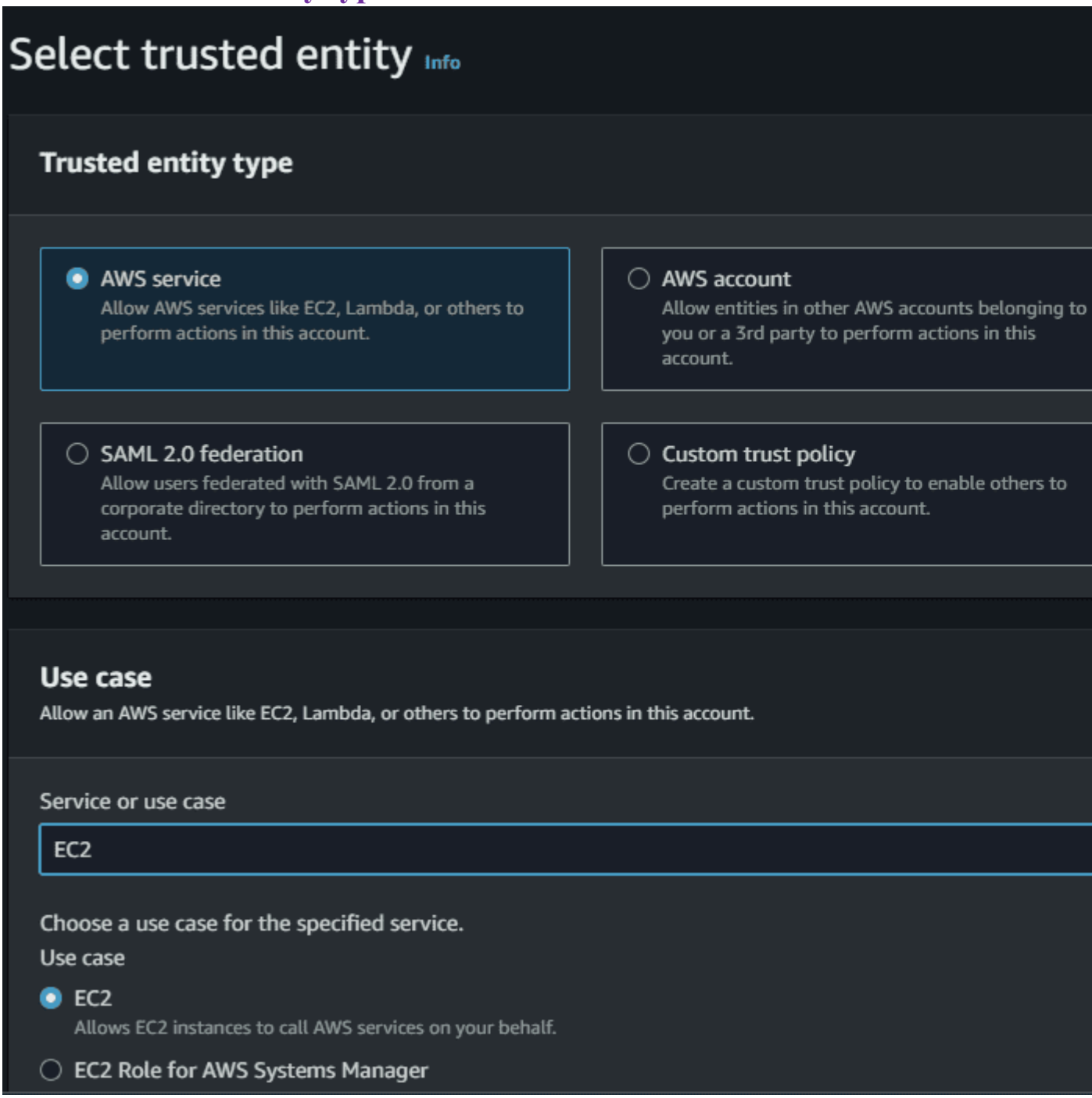
P10 : AWS CI/CD Pipleline

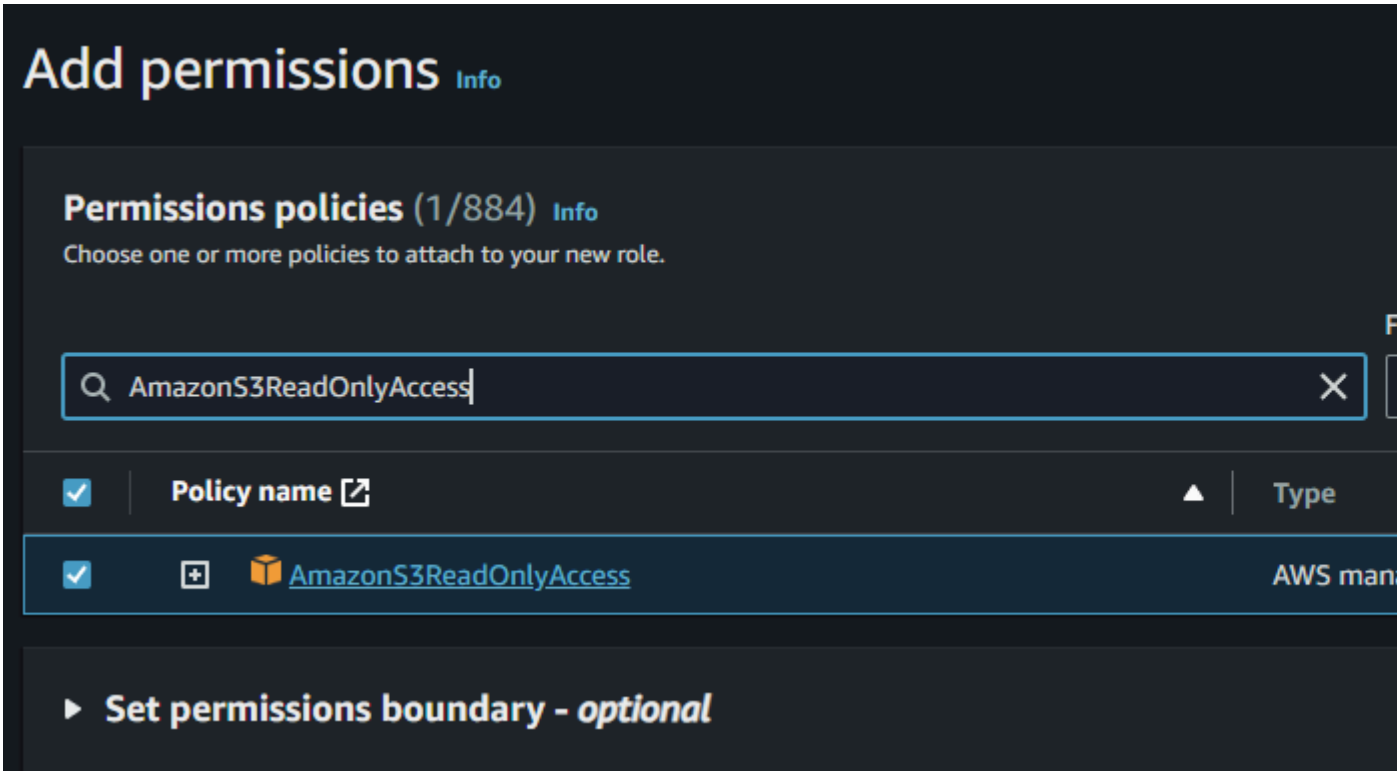
Step 1: Create IAM Role for EC2 and AWS CodeDeploy

- Navigate to IAM service.
- Then go to roles and create a new role.
- Select trusted entity type as AWS Service and use case as EC2



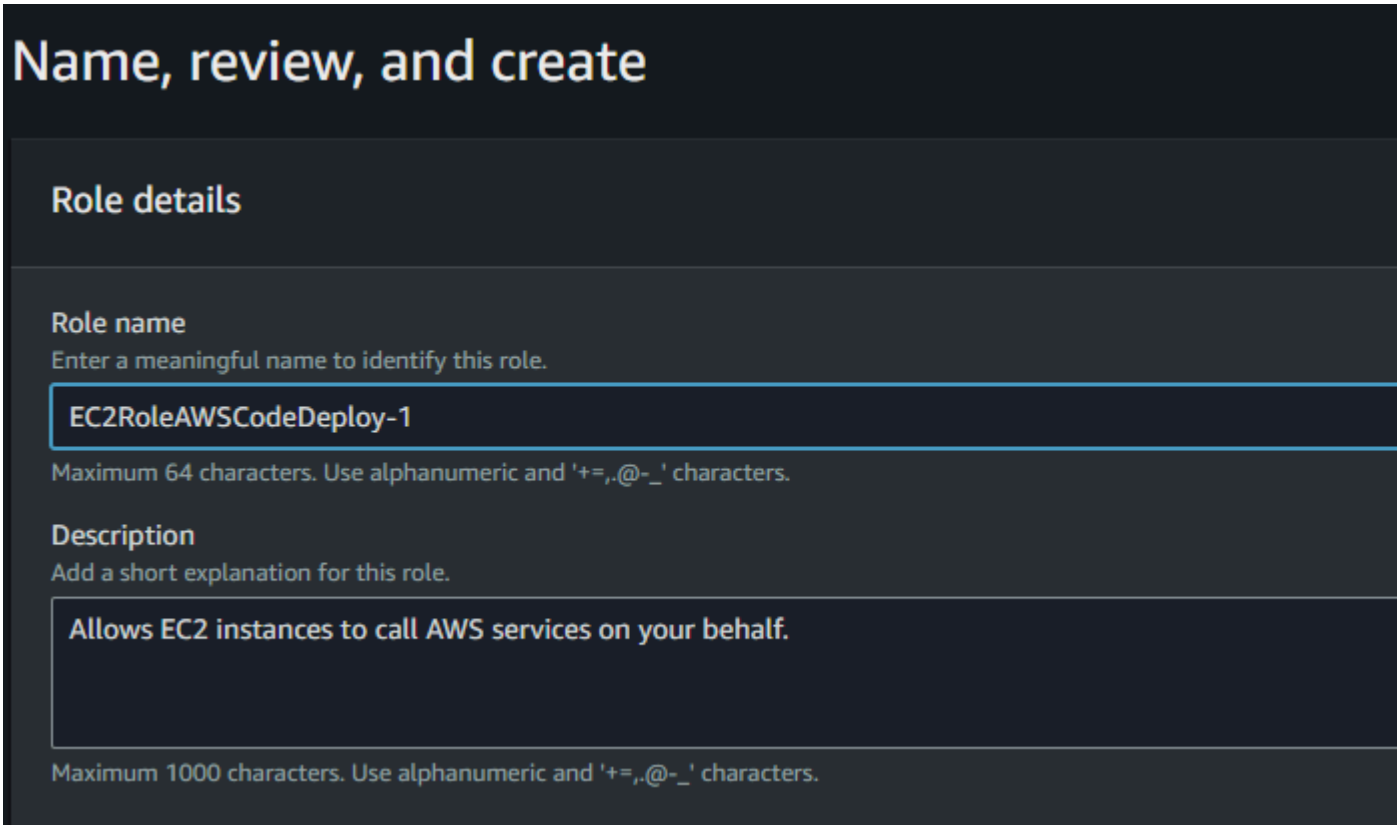
Step 2: Add permissions To IAM Role

- **Select AmazonS3ReadOnlyAccess permission. It will allow our EC2 instance to access stored artifacts from the Amazon S3 bucket.**



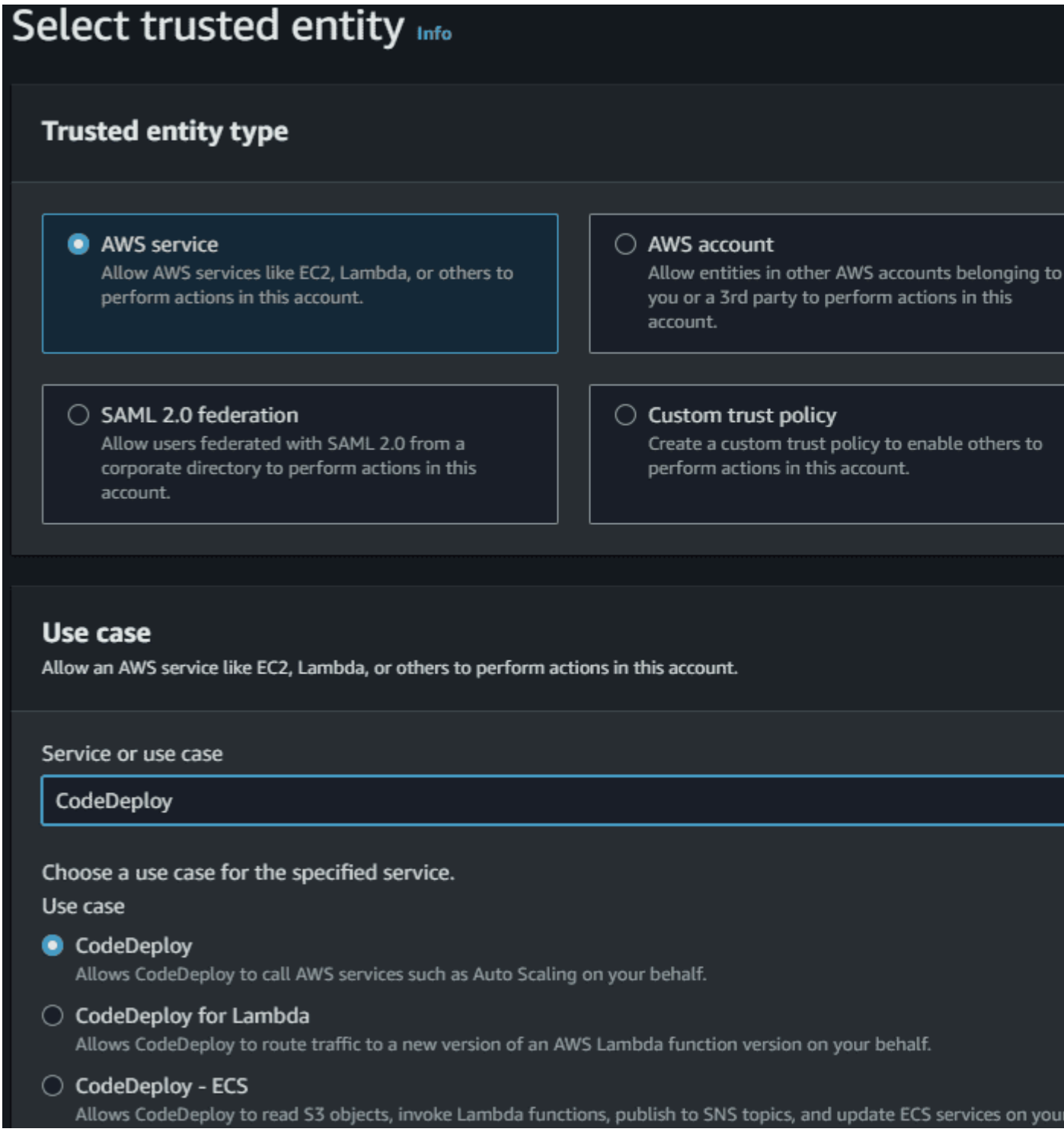
Step 3: Creating The Role For AWS CodeDeploy

- **Provide the Name, review and Click on Create for creating the Role.**
- **Select an appropriate role name and click on create role.**

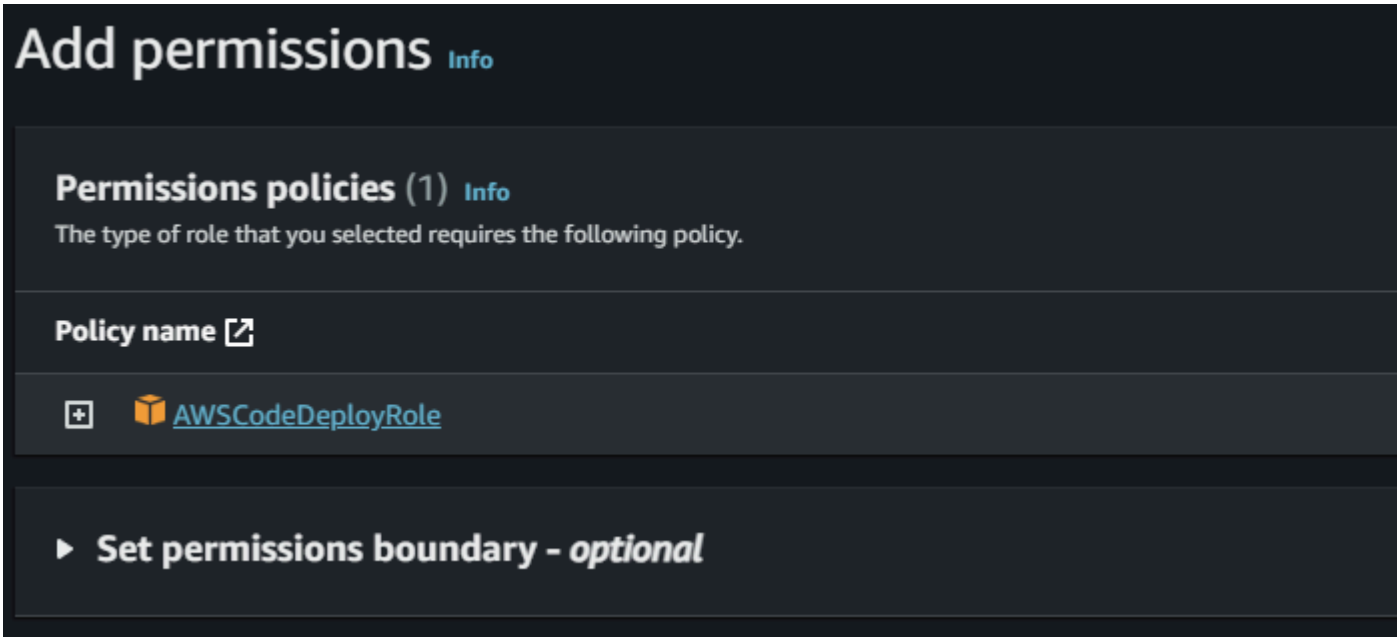


Step 4: Creating New Service Role For CodeDeploy

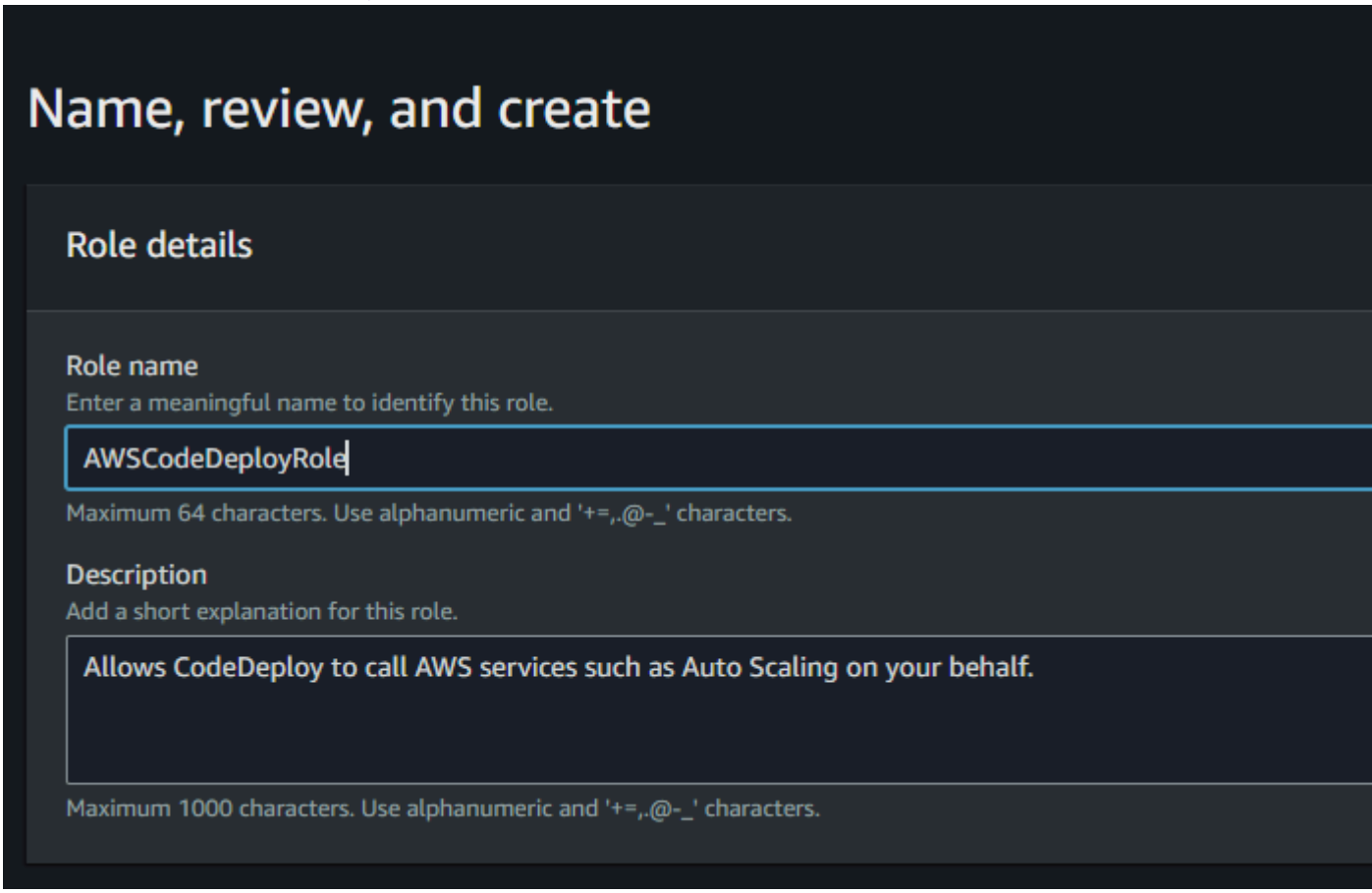
- **Create a new service role for CodeDeploy and attach AWSCodeDeployRole policy which will provide the permissions for our service role to read tags of our EC2 instance, publish information to Amazon SNS topics and much more task.**
- **Repeat the Above 3 steps again with trusted entity type AWS Service, use case CodeDeploy.**



- **Add AWSCodeDeployRole permissions to this creating Role**



- **Provide the Name, review and create the role.**



Step 5: Launch An Linux EC2 instance

- **Select the instance with AMI such as "Amazon Linux" and connect to CLI Console.**

- **Switch to root user from ec2-user to gain admin access power by using following command "sudo su" in Linux.**

```
sudo su
```

Step 6: Update The Packages

- **The command "sudo yum update" is used in Amazon Linux, CentOS, and Red Hat Linux distributions to update installed packages on your system to their latest available versions.**

```
sudo yum update
```

Step 7: Install The Ruby And Wget Software

- **The command 'sudo yum install ruby' is used to install the Ruby programming software using the YUM package manager.**

```
sudo yum install ruby
```

- **The command sudo yum install wget is used to install the "wget" package on a system running Amazon Linux, CentOS, or other Red Hat-based Linux distributions that use the YUM package manager.**

```
sudo yum install wget
```

Step 8: Download CodeDeploy Agent Script

- **Downloading the AWS CodeDeploy agent installation script from the AWS S3 bucket is an essential step in setting up AWS CodeDeploy for your infrastructure.**
- **The CodeDeploy agent is a lightweight, scalable software component that enables AWS CodeDeploy to deploy and manage applications on your EC2 instances or on-premises servers.**

```
wget
https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install
```

Step 9: Run Installation Script

- **The command chmod +x ./install is used to make a file executable in a Unix-like operating system, including Linux.**

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

The command 'sudo ./install auto' is likely used to run an installation script with superuser (administrator) privileges and pass the "auto" argument to the script.

```
sudo ./install auto
```

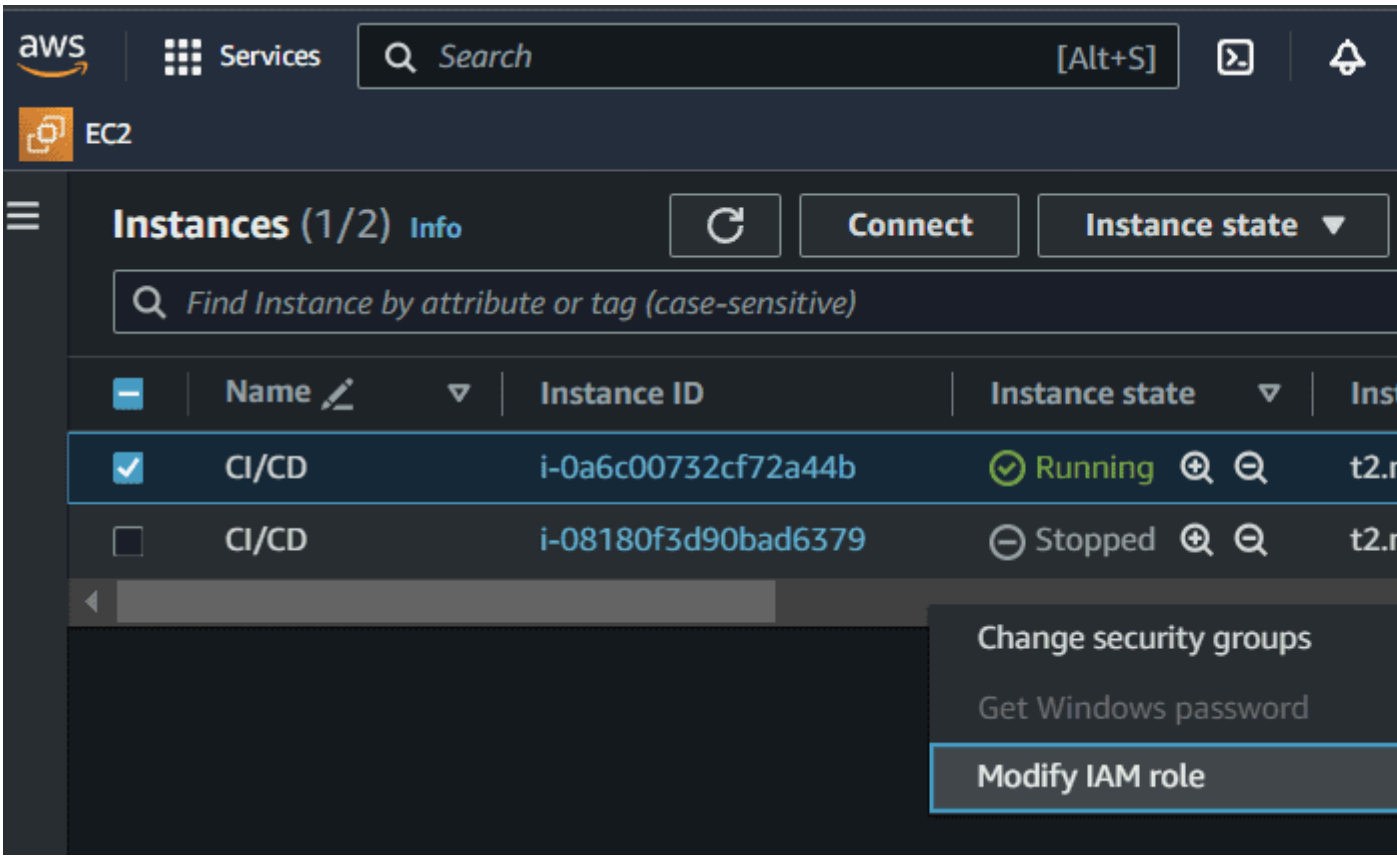
Step 10: Check CodeDeploy Agent Status

- **The command sudo service codedeploy-agent status is used to check the status of the AWS CodeDeploy agent running on your system.**

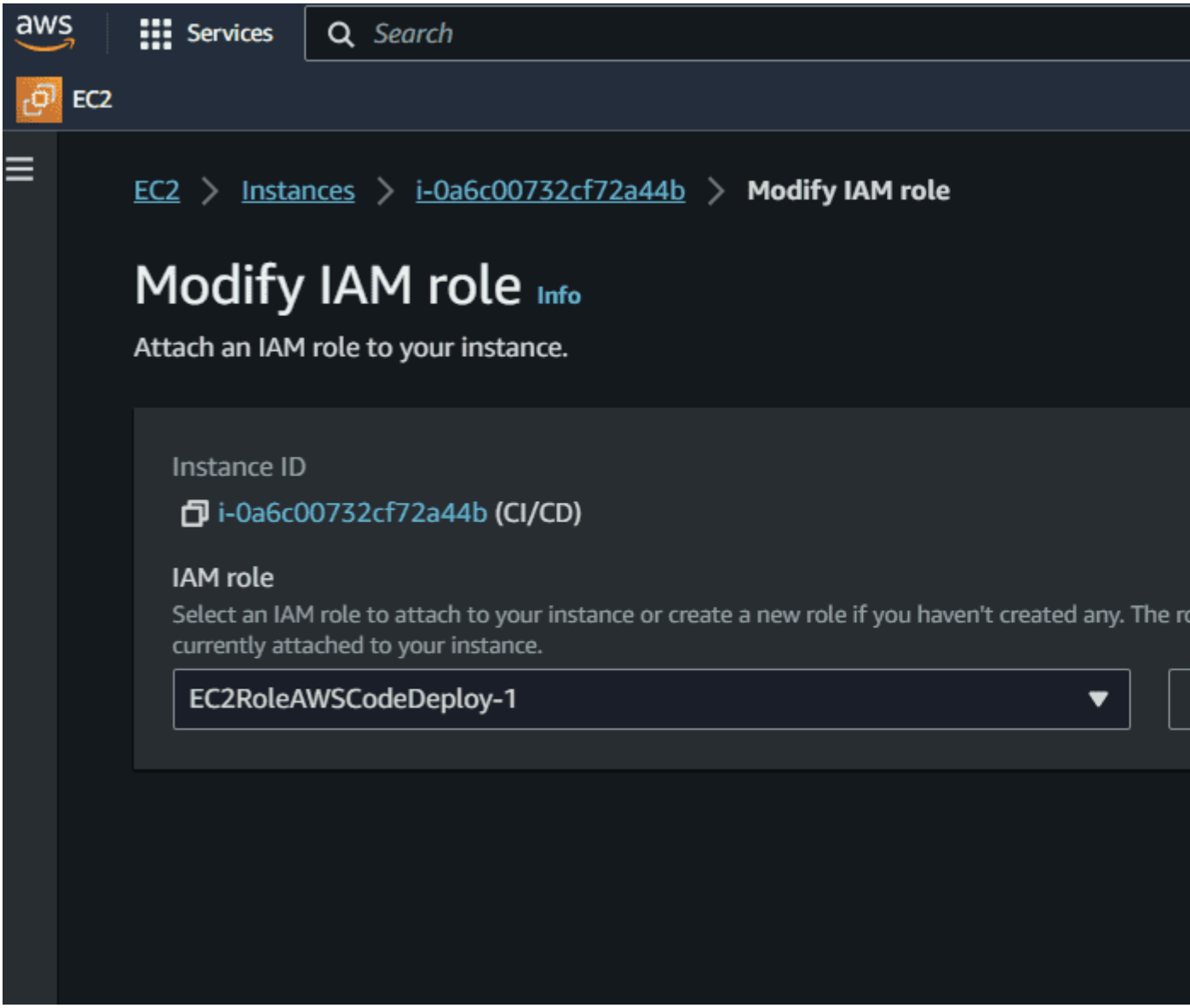
```
sudo service codedeploy-agent status
```

Step 11: Modifying IAM Role

- **After running the following commands, select the instance and click on "Actions", then click on "Security" and click on "Modify IAM Role". Then choose the above created IAM Role and click on "Update IAM Role".**
- **After this step, your EC2 instance gets attached with your above created IAM Role.**

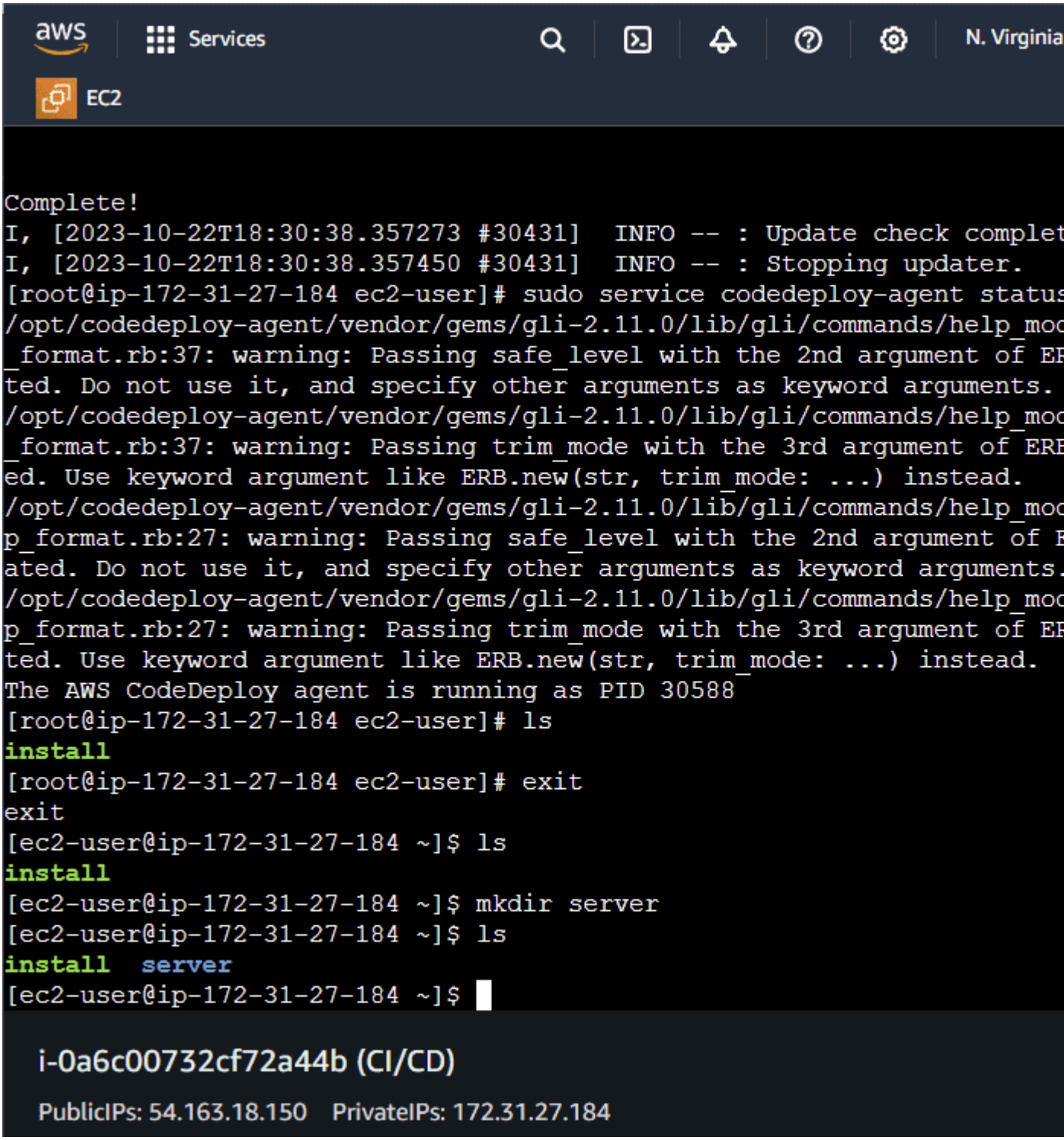


- **Modify the IAM role by clicking on the button Update IAM role as shown in the figure.**



Step 12: Finalizing The Configuration

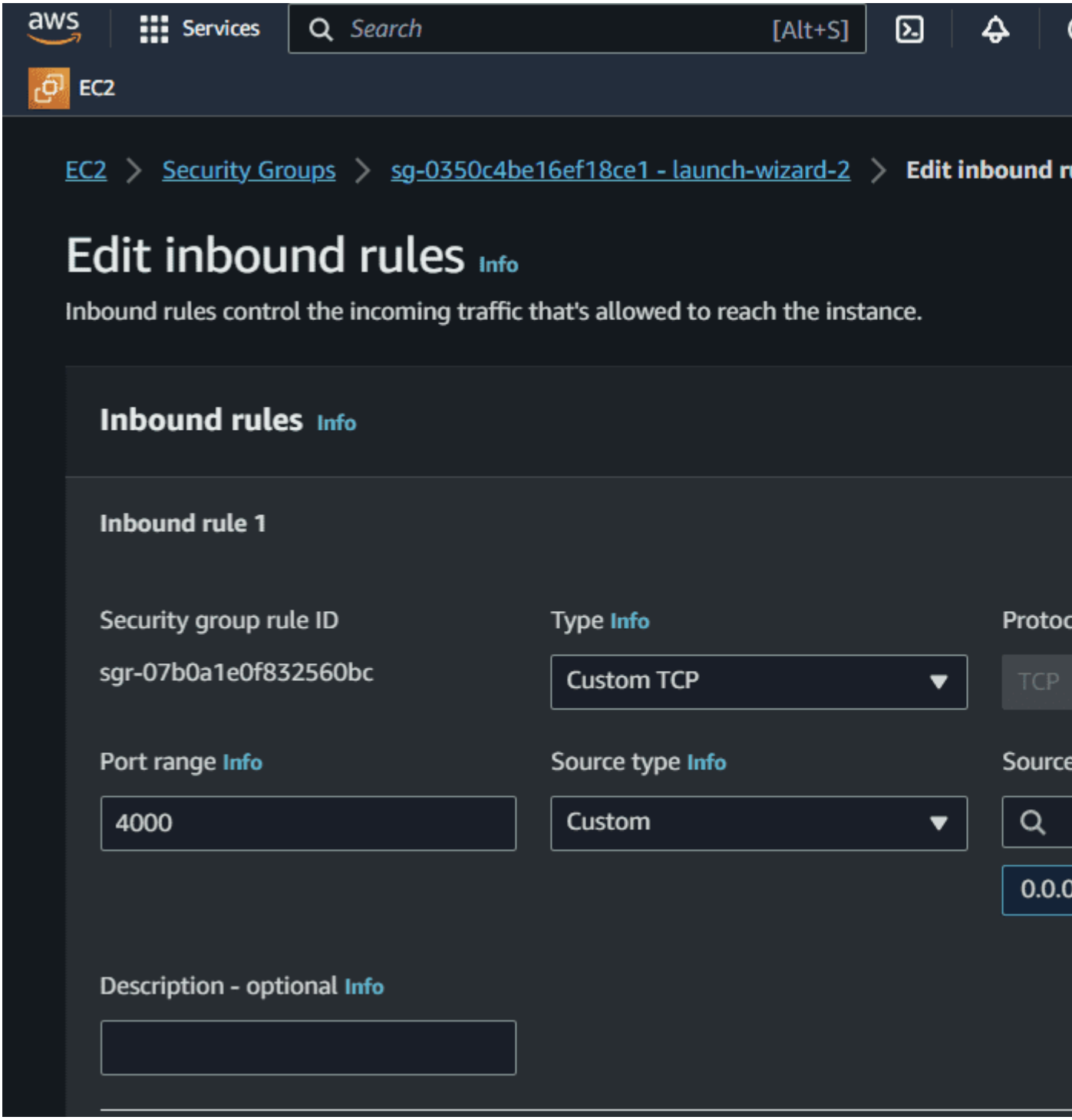
After this process, go to the console where your instance is connected and run the command "exit" to exit from the root folder and go back to the EC2 folder. Make a directory on the EC2 folder named "server", this is the directory where my source code will be deployed.



- **Then after doing the above process, come back to the running instances list.**
- **Select your currently created running instance and go to the "Security" section present at the end of the page.**
- **Click on the link present under the "Security Groups". After redirecting to the required page, click on "Edit Inbound rules"**

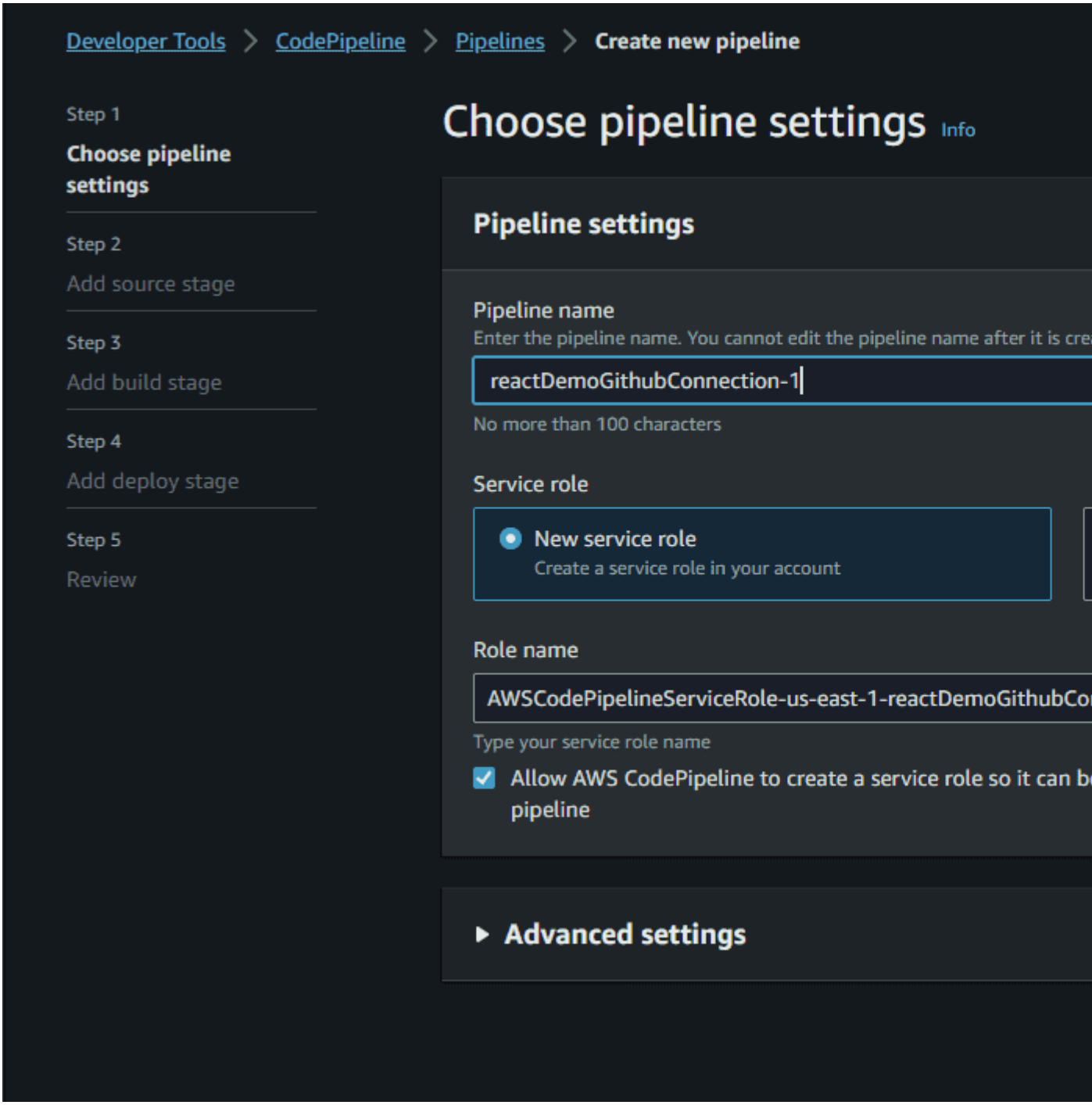
under the section of "Inbound rules" present at the end of the page.

- **Then add a rule, select a port range of your choice and select the source as "Anywhere-IPv4" from the dropdown menu and then click on "Save rules".**
- **Basically, let me give you a overview what we are actually doing here. In brief, when you add an inbound rule to a security group for an instance with port range (in my case, it was 4000) and set the source to "Anywhere-IPv4," you are allowing any computer or device on the internet to connect to your instance through port 4000.**
- **This is like opening a door (port 4000) on your server and letting anyone from anywhere access the service or application running on that port.**



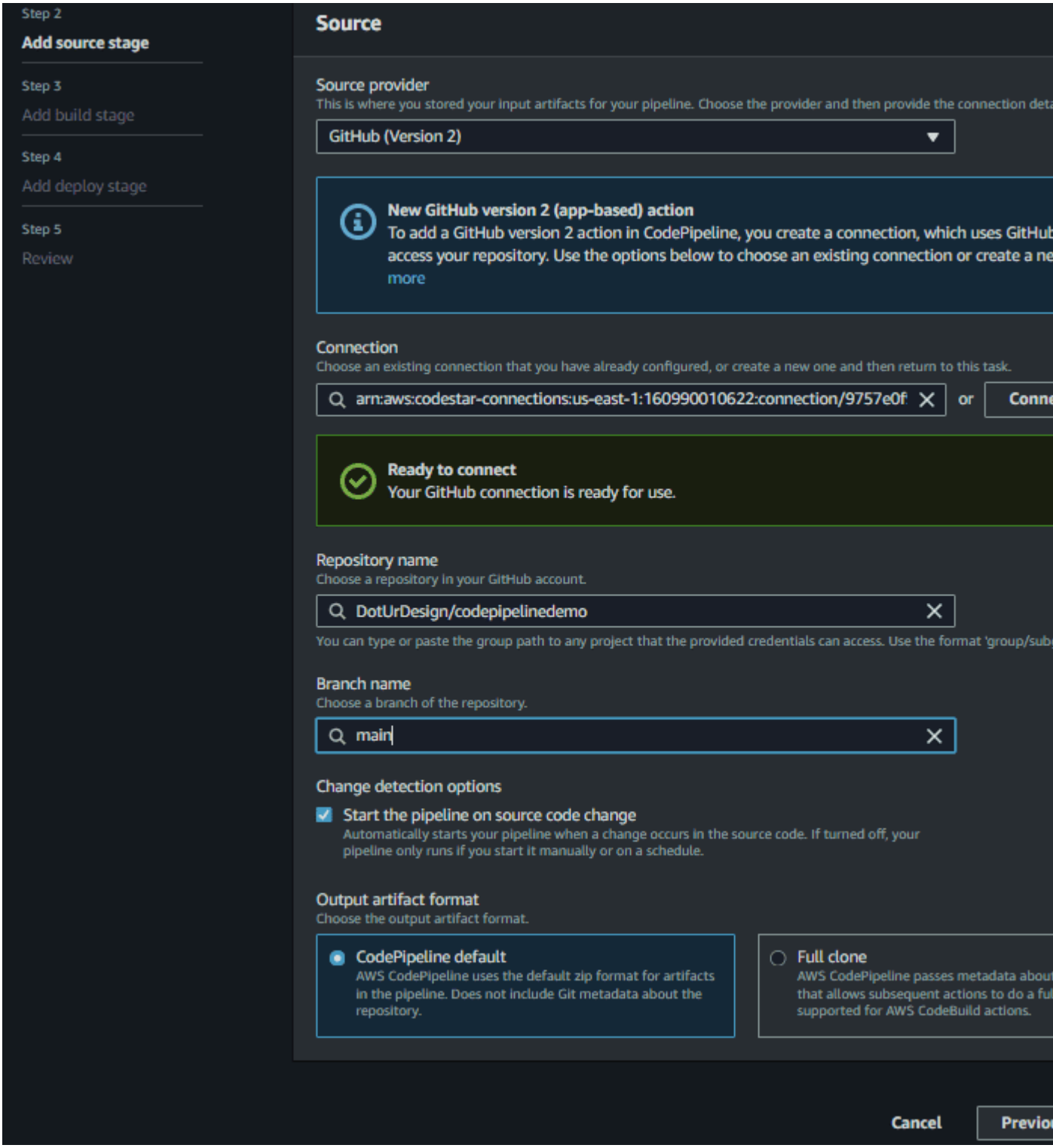
Step 13: Create A New Pipeline

- **Create a CodePipeline using Github, CodeBuild and CodeDeploy**
- **Firstly Create CodePipeline navigate to CodePipeline via AWS Management Console and click on Create pipeline.**



Step 14: Choose Github In Code Source

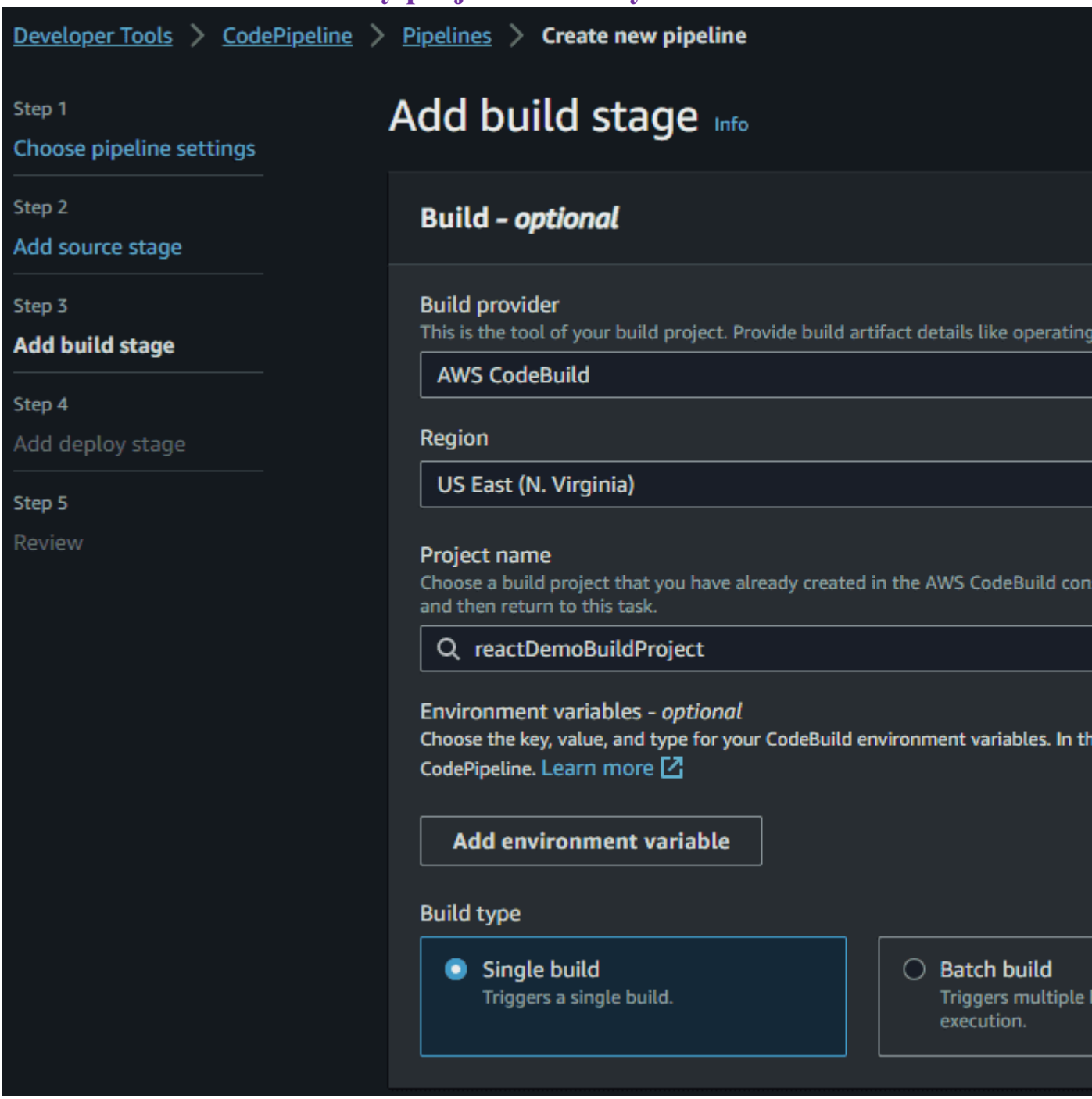
- After selecting GitHub as the source provider, click on the Connect to GitHub button. You'll then be prompt to enter your GitHub login credentials.
- Once you grant AWS CodePipeline access to your GitHub repository, you can select a repository and branch for CodePipeline to upload commits to this repository to your pipeline.



Step 15: 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.
- Note: Buildspec file is a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a

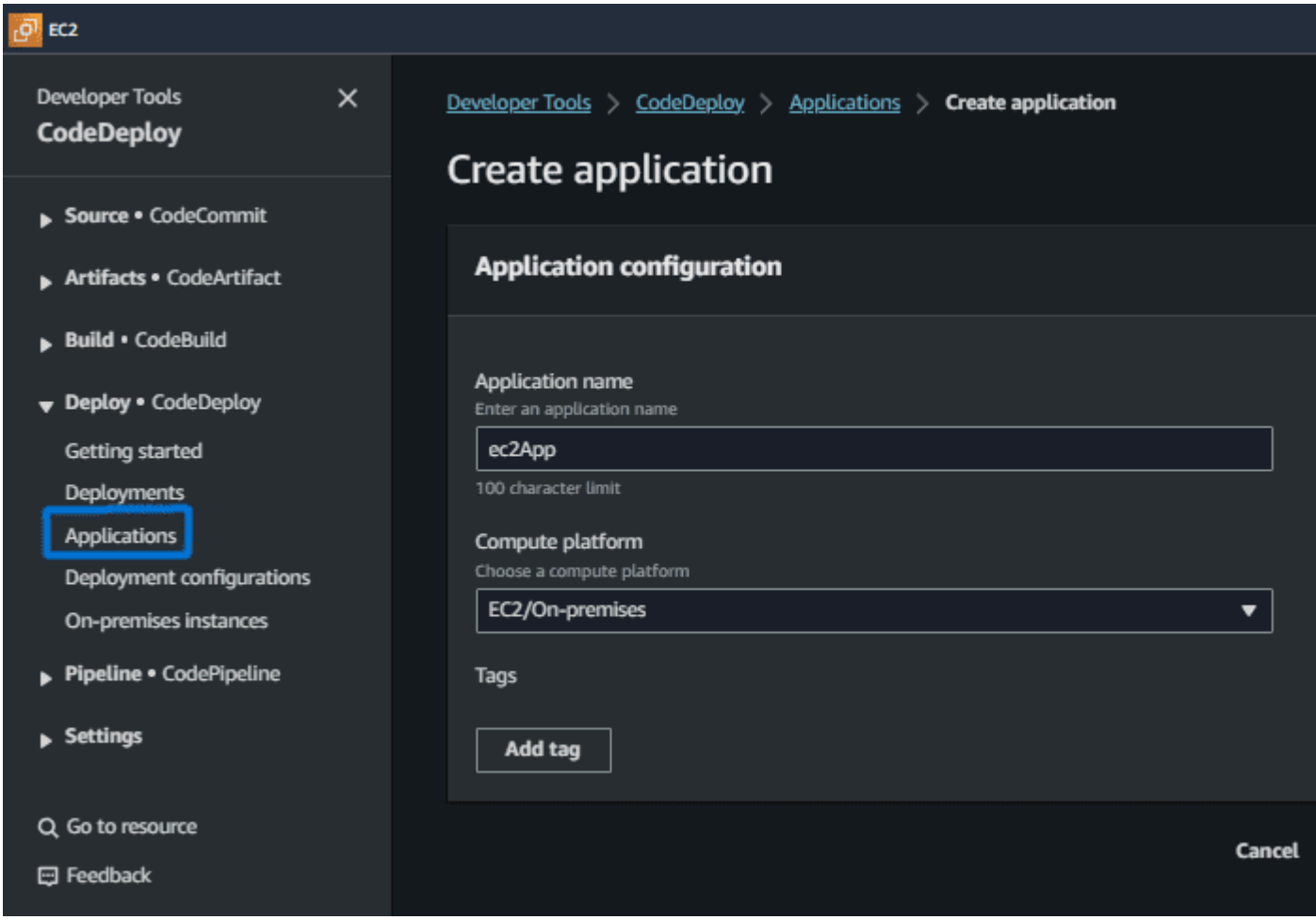
build. For my project, I created a buildspec.yaml file and added it in the root of my project directory.



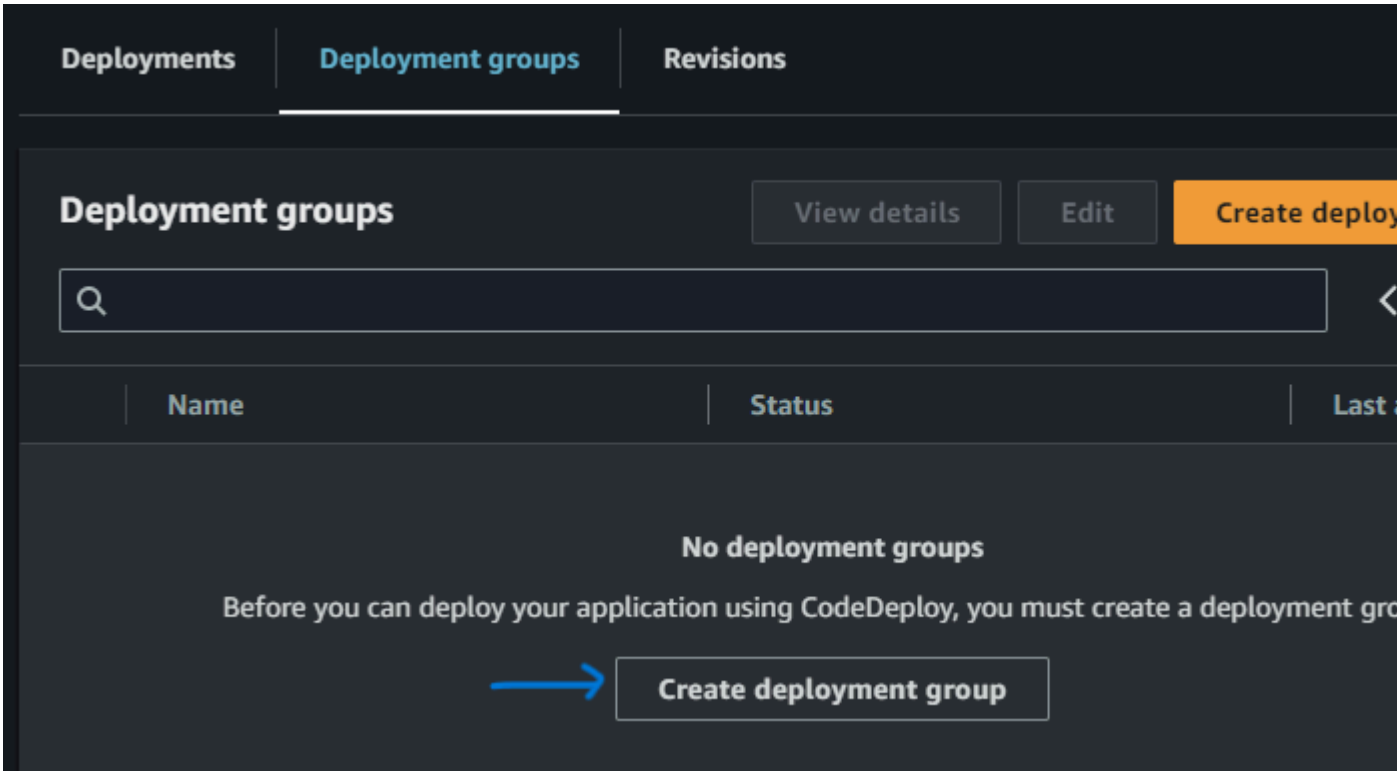
Step 16: Add Deploy Stage

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

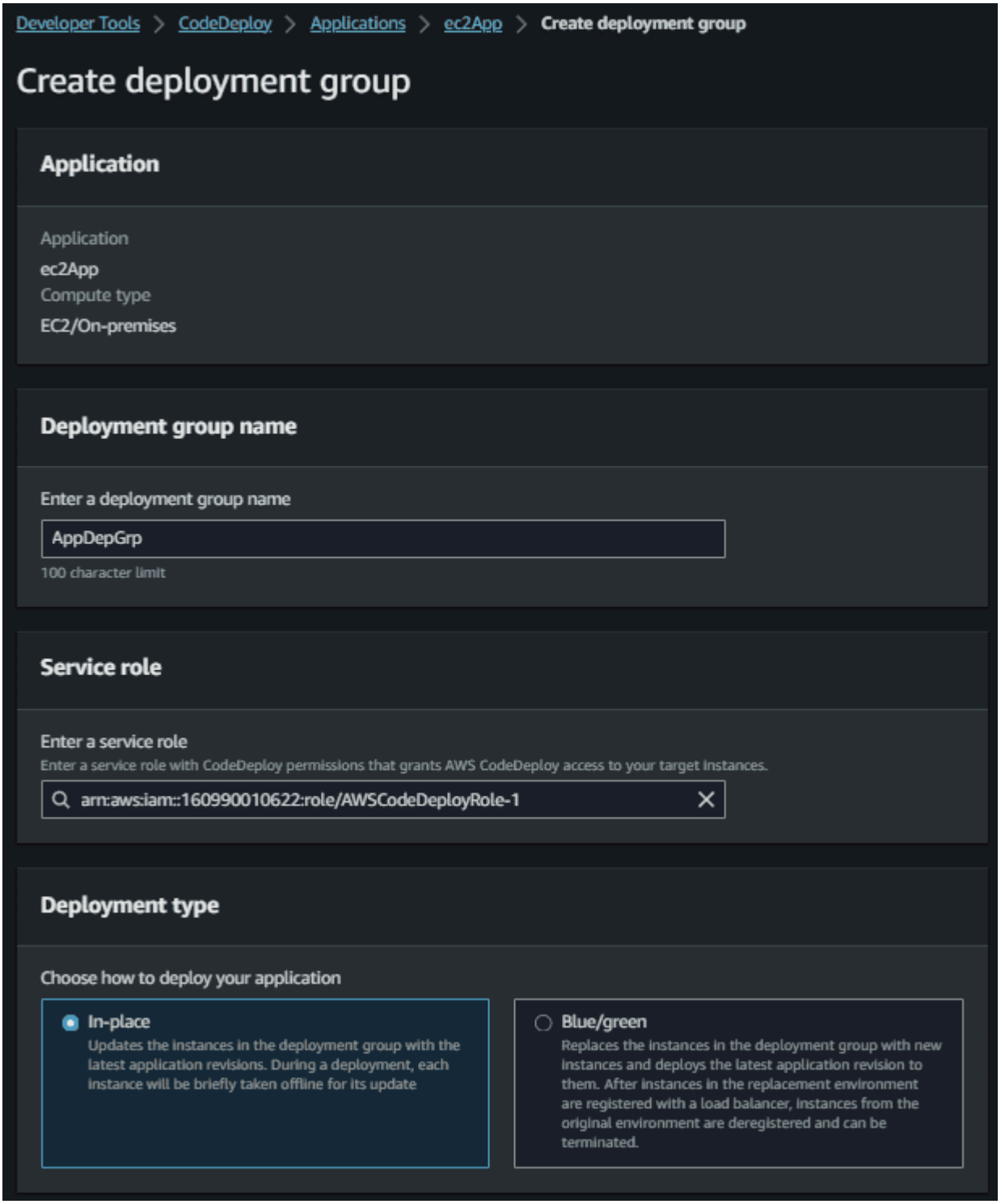
- Go to code deploy in the navigation, Select Application, then add create a deployment group.



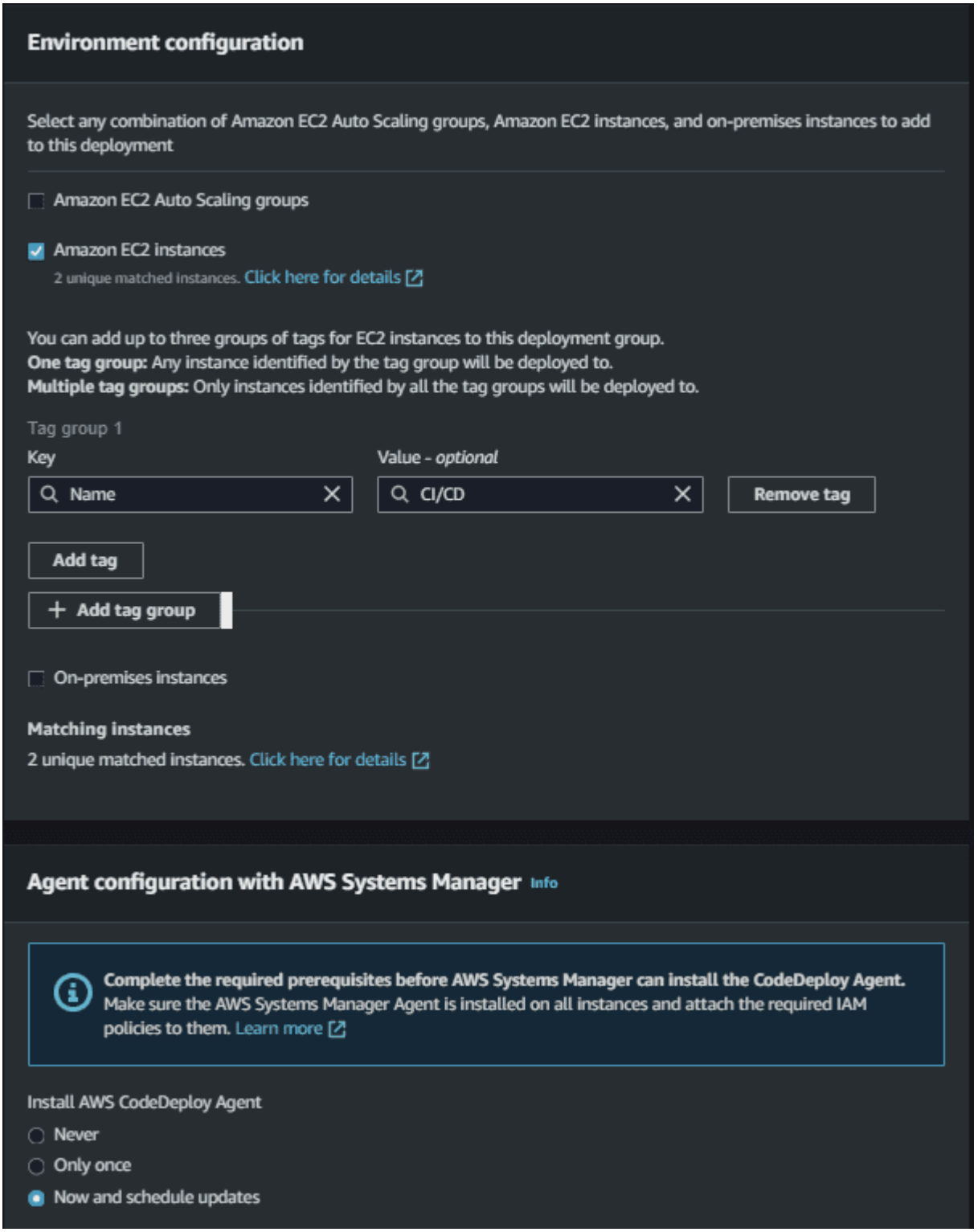
- **Create a deployment Group by clicking on the button "Create deployment group", the following screenshot illustrates with practically.**



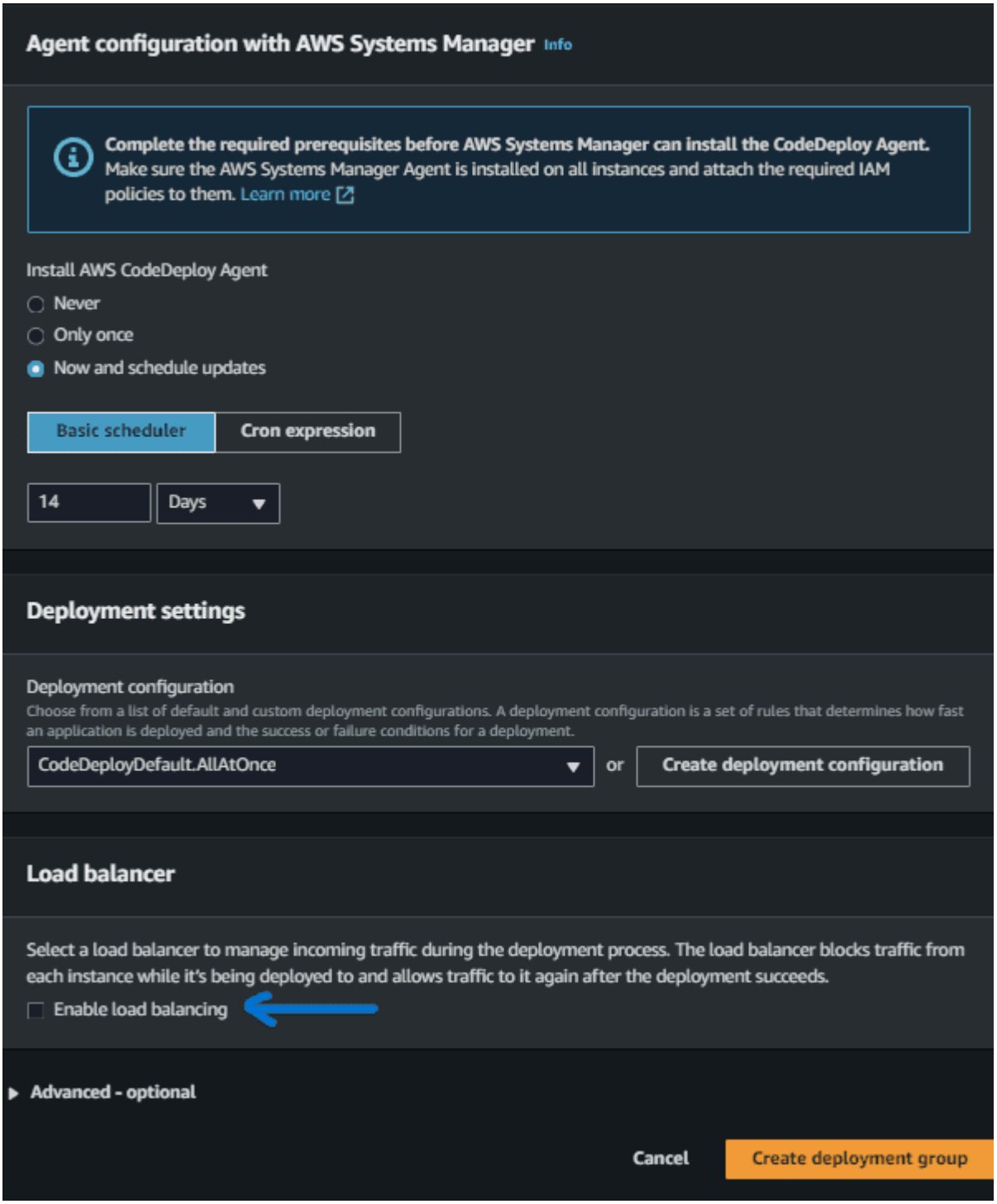
- **In deployment group Select EC2 instances and select Tag and Value**



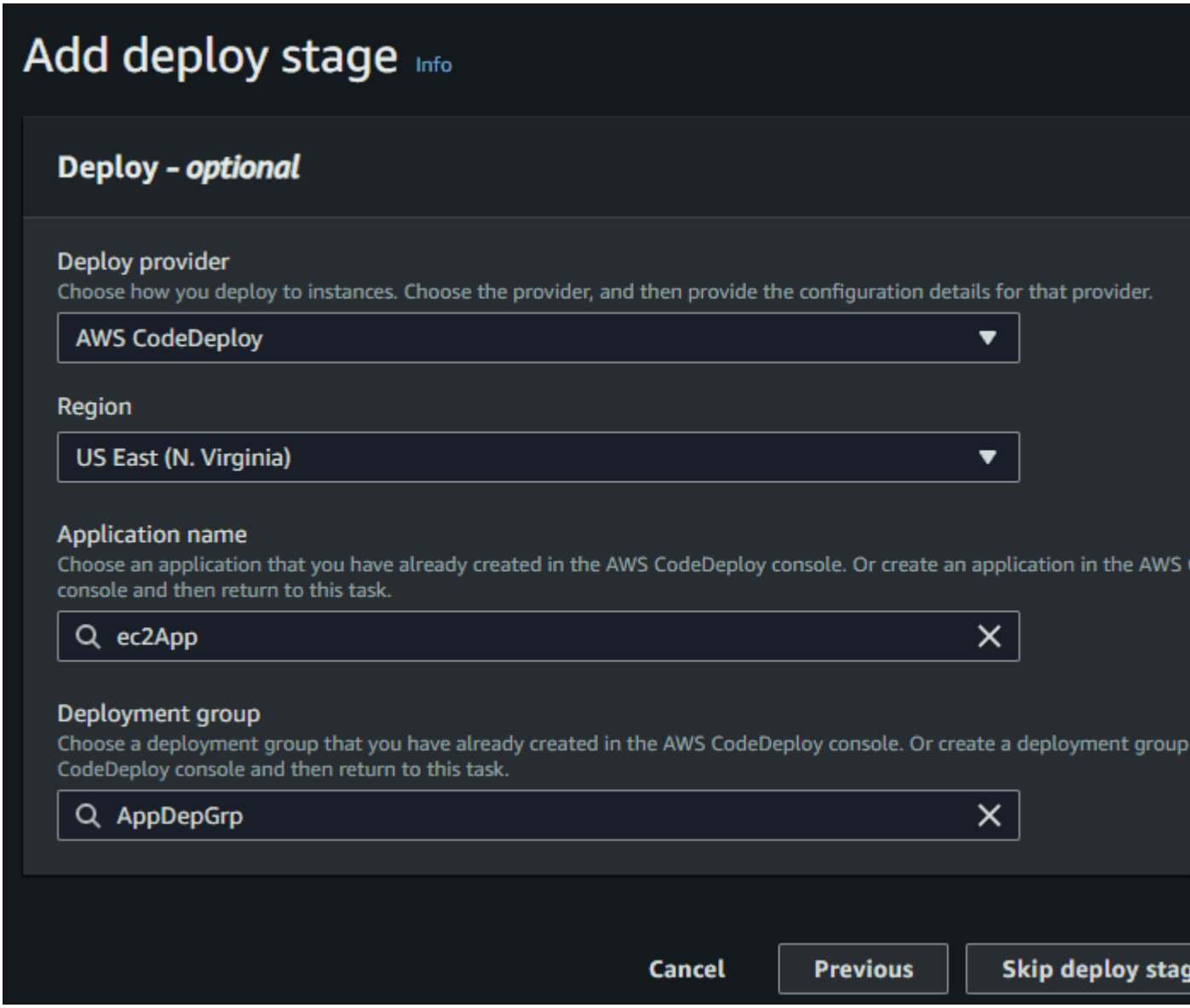
- **Provide the Environment configurations such as select the Amazon EC2 Instances and provide the key and values to it.**



- **Uncheck Load Balancer Option**

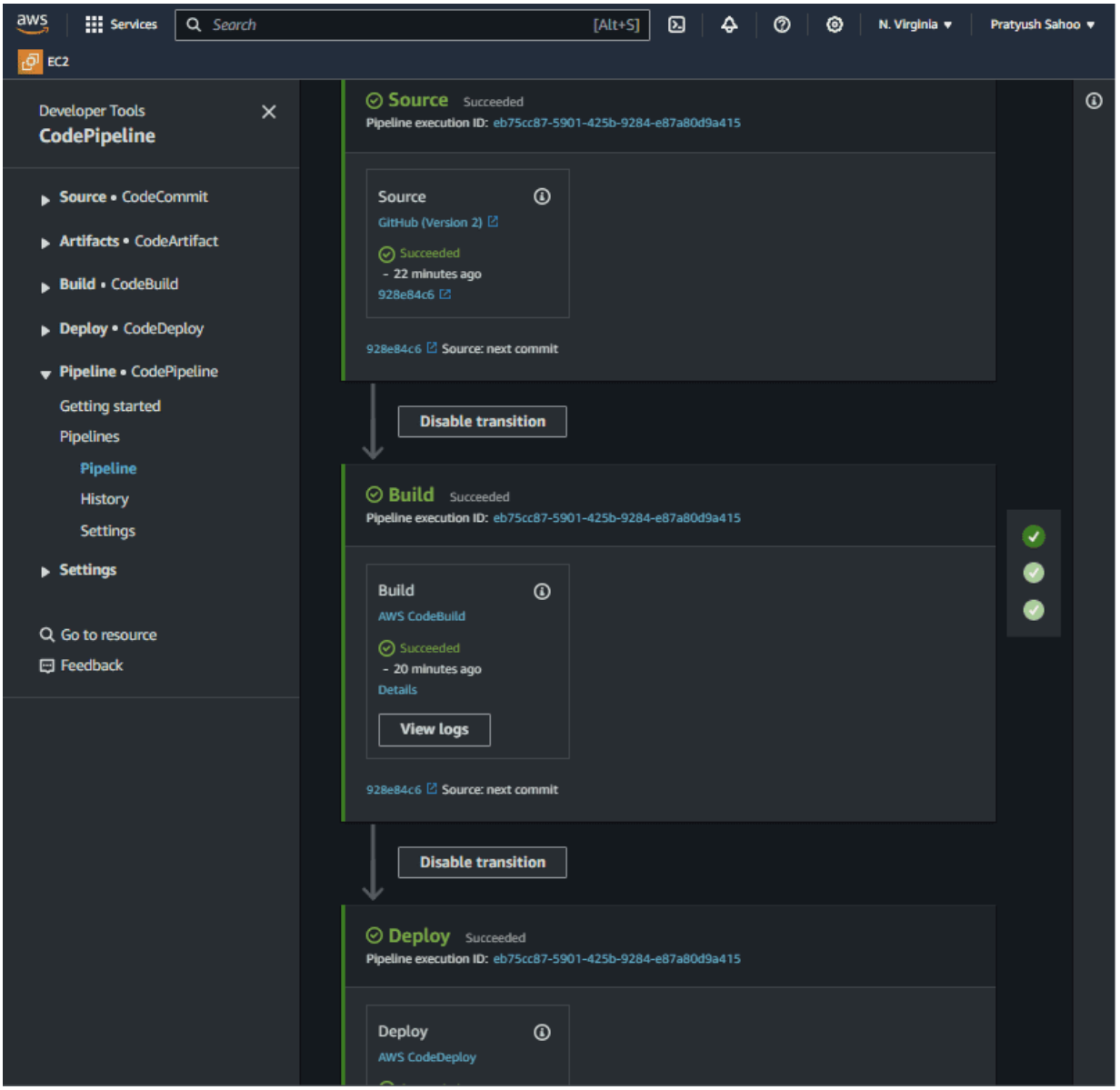


- **Finally Come on Add Deploy Stage and select that created Application name & Deployment group**

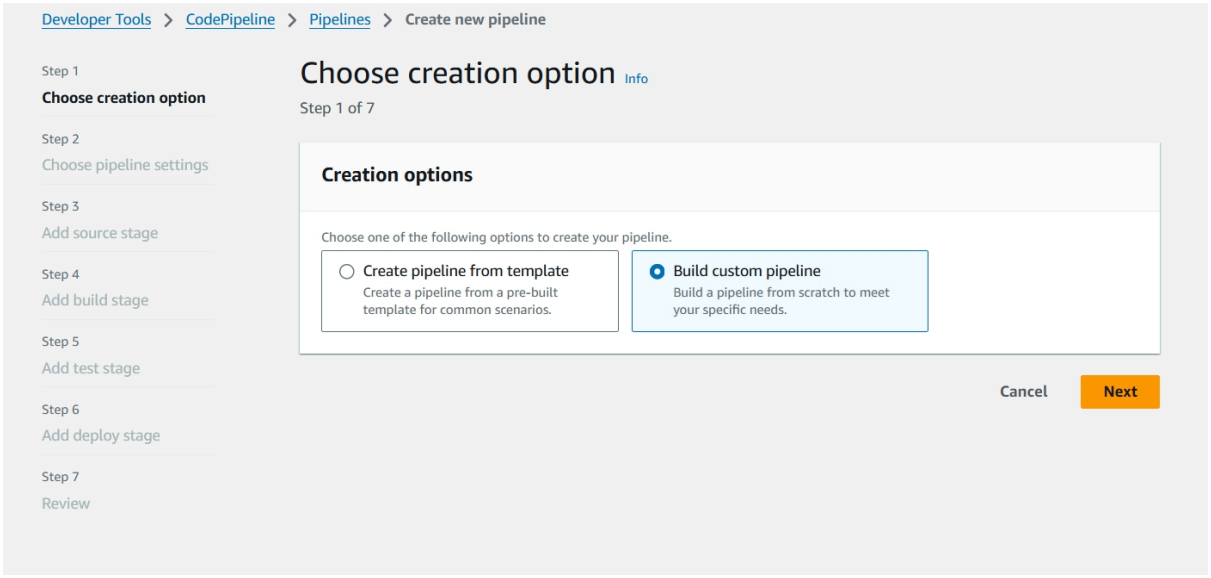


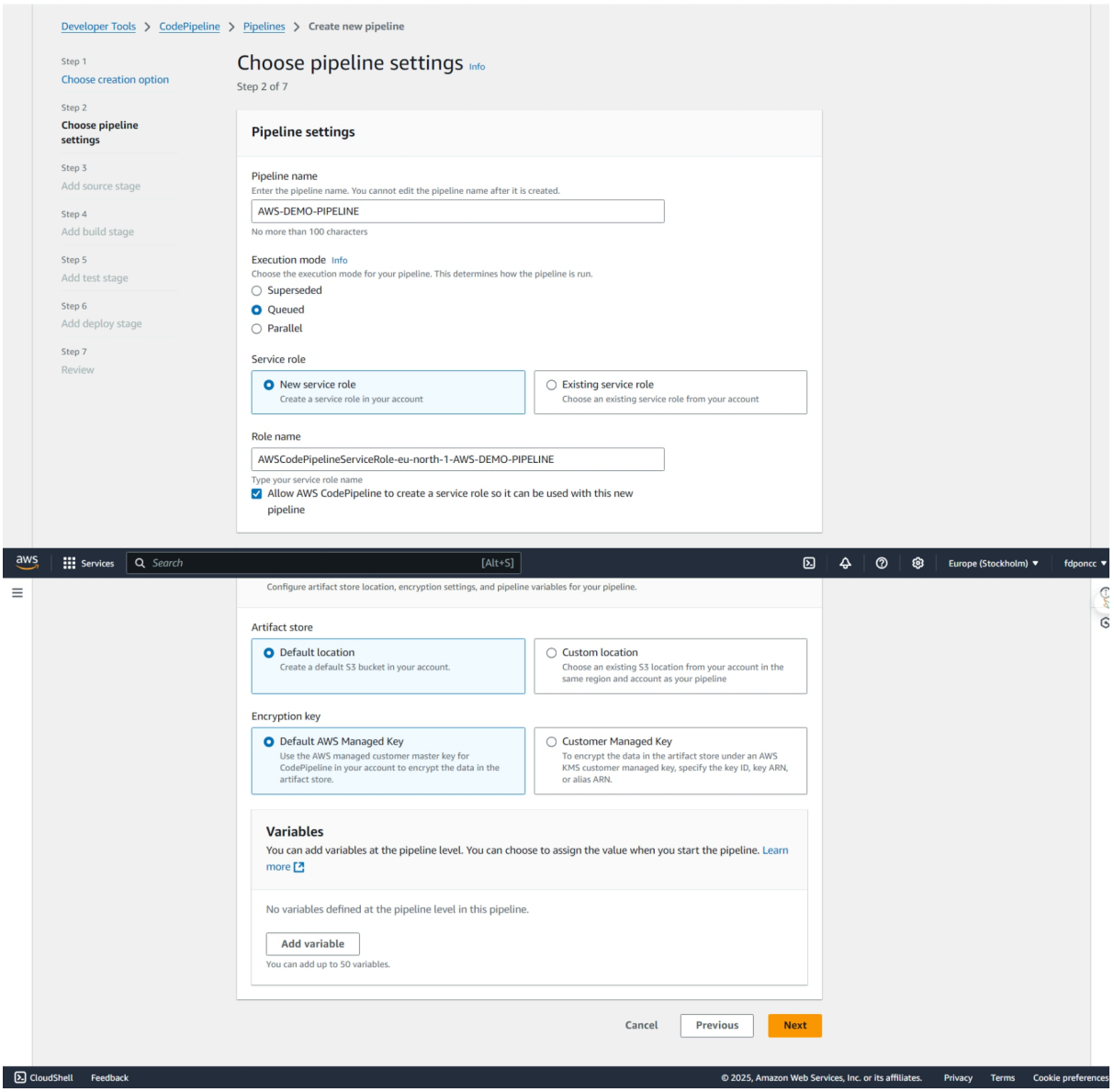
Step 17: Review And Create

- As a final step review and create it. By creating this we have successful the created a CI/CD pipeline in AWS.


aws
Services
Search
[Alt+S]
N. Virginia
Pratyush Sahoo
EC2
Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Feedback
Source Succeeded
Pipeline execution ID: eb75cc87-5901-425b-9284-e87a80d9a415
Source
GitHub (Version 2)
Succeeded
- 22 minutes ago
928e84c6
928e84c6 Source: next commit
Disable transition
Build Succeeded
Pipeline execution ID: eb75cc87-5901-425b-9284-e87a80d9a415
Build
AWS CodeBuild
Succeeded
- 20 minutes ago
Details
View logs
928e84c6 Source: next commit
Disable transition
Deploy Succeeded
Pipeline execution ID: eb75cc87-5901-425b-9284-e87a80d9a415
Deploy
AWS CodeDeploy

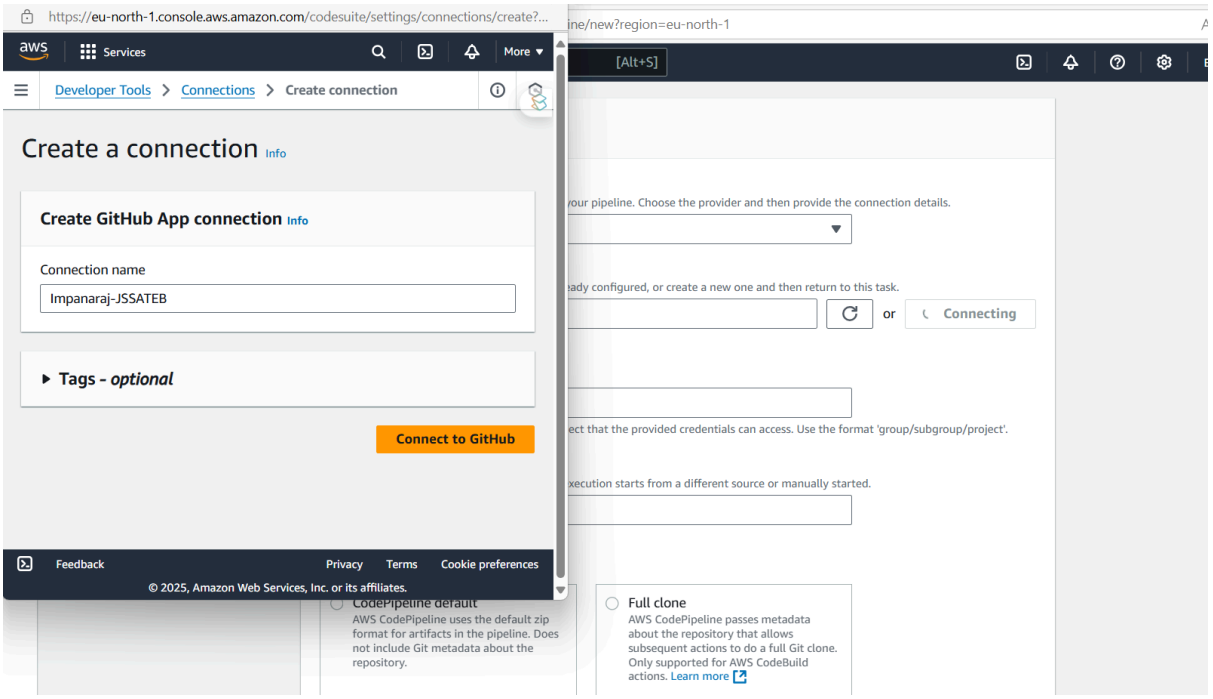
PIPELINE CREATED STAGE ALTERNATE METHOD



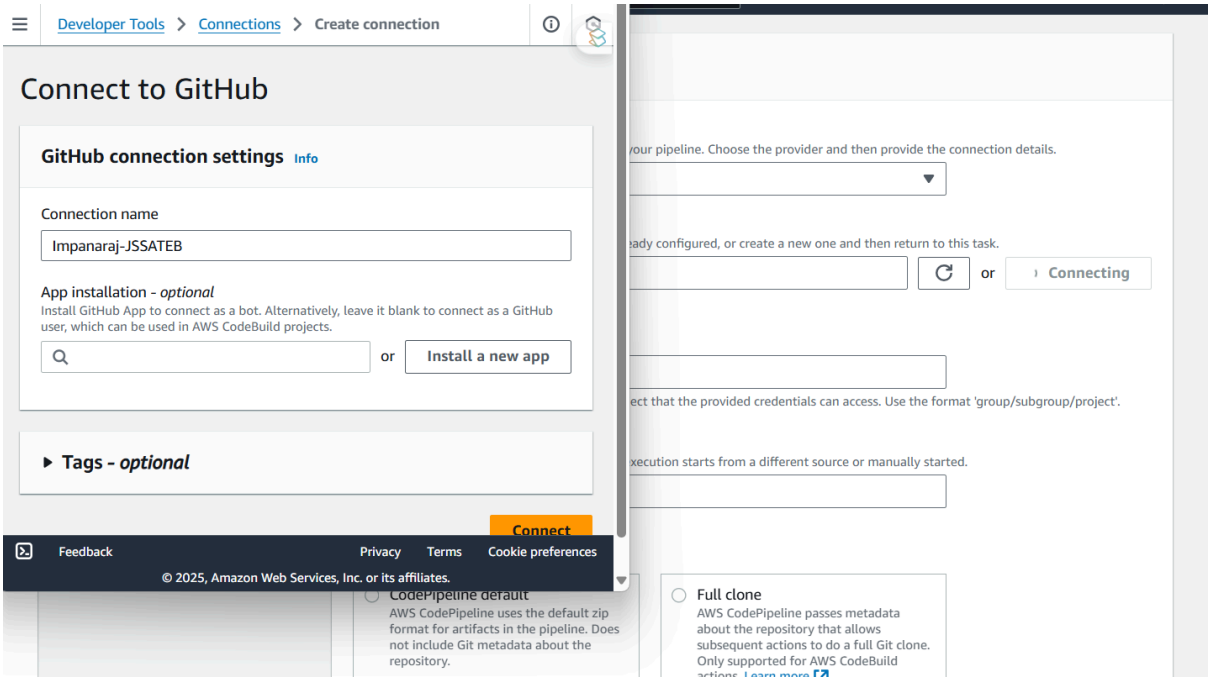


NEXT

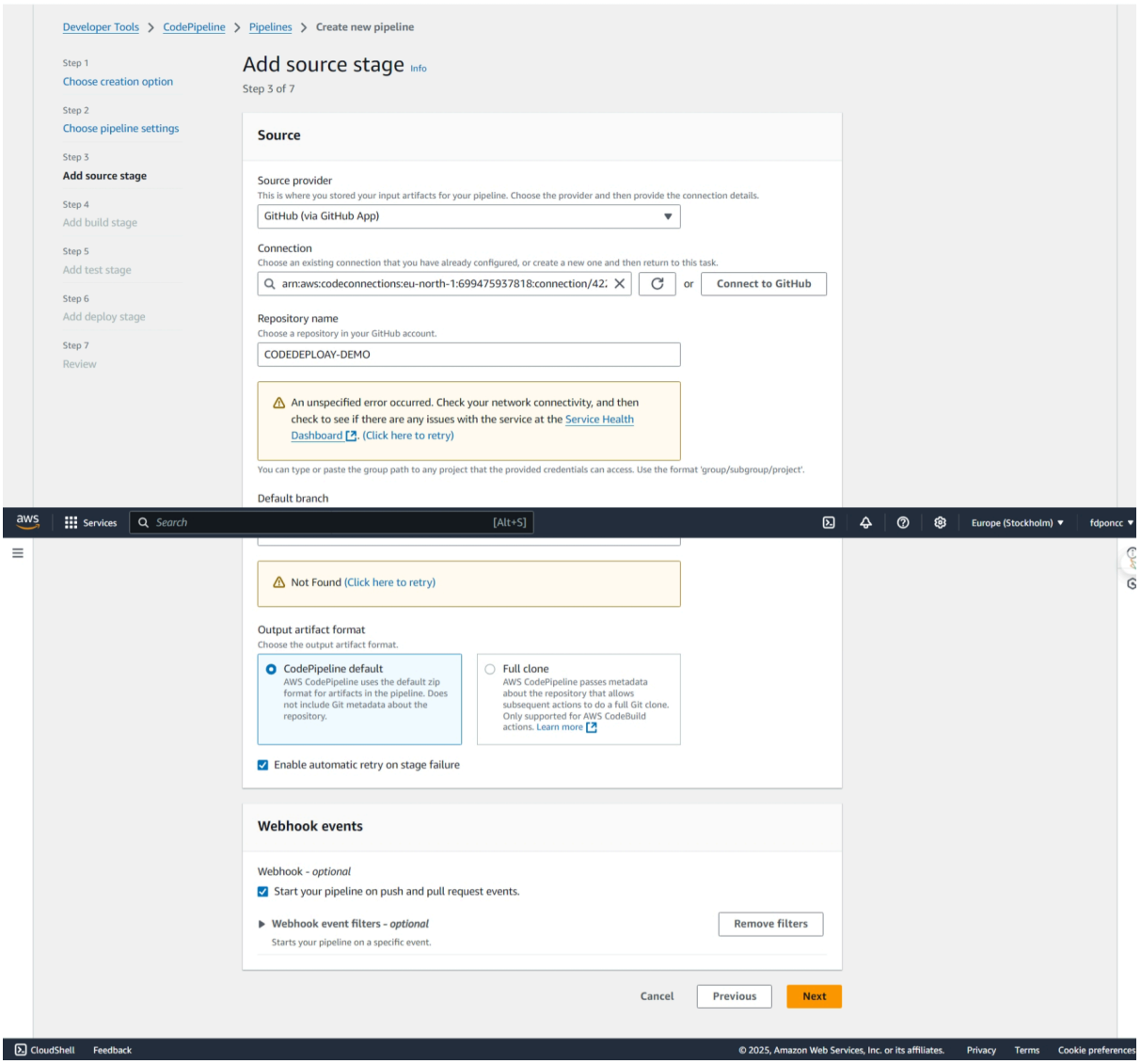
CONNECT TO GITHUB

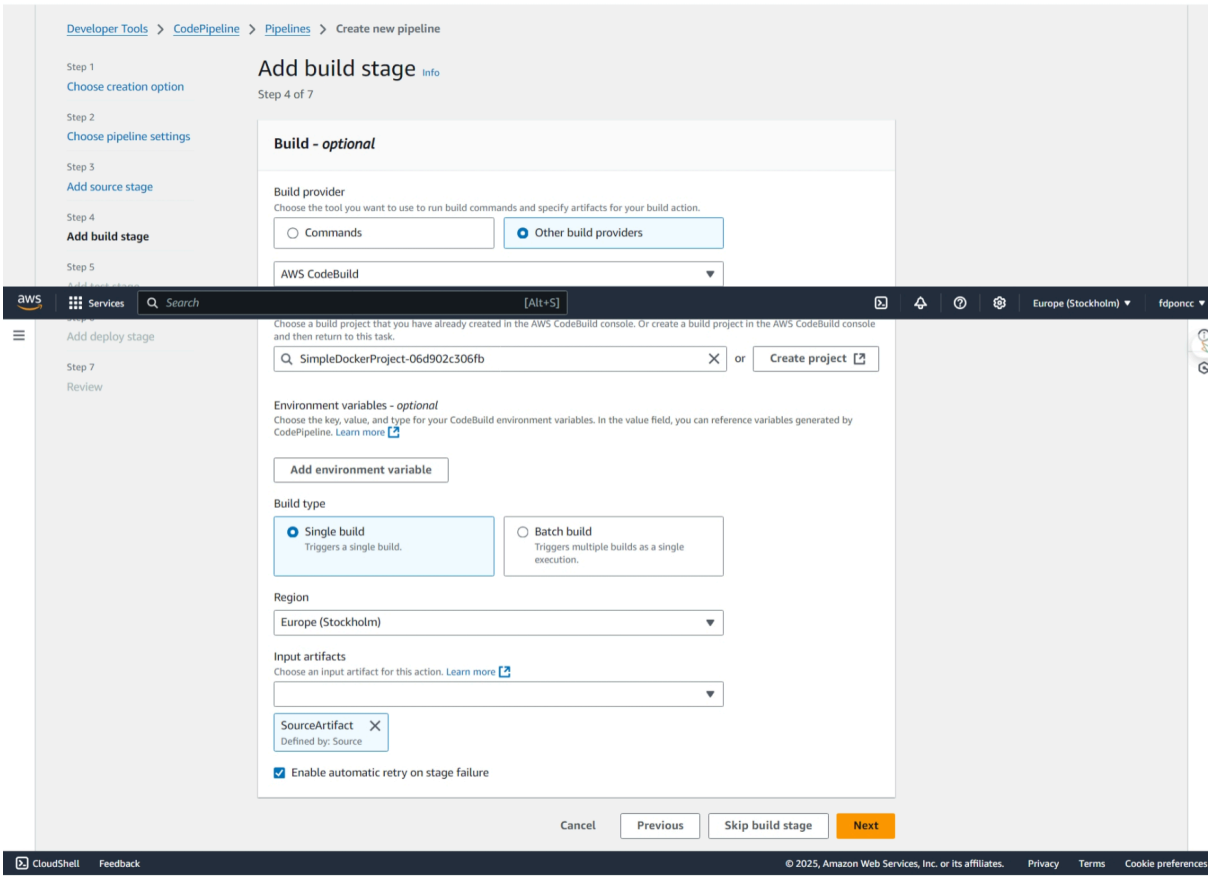


CONNECT TO GITHUB

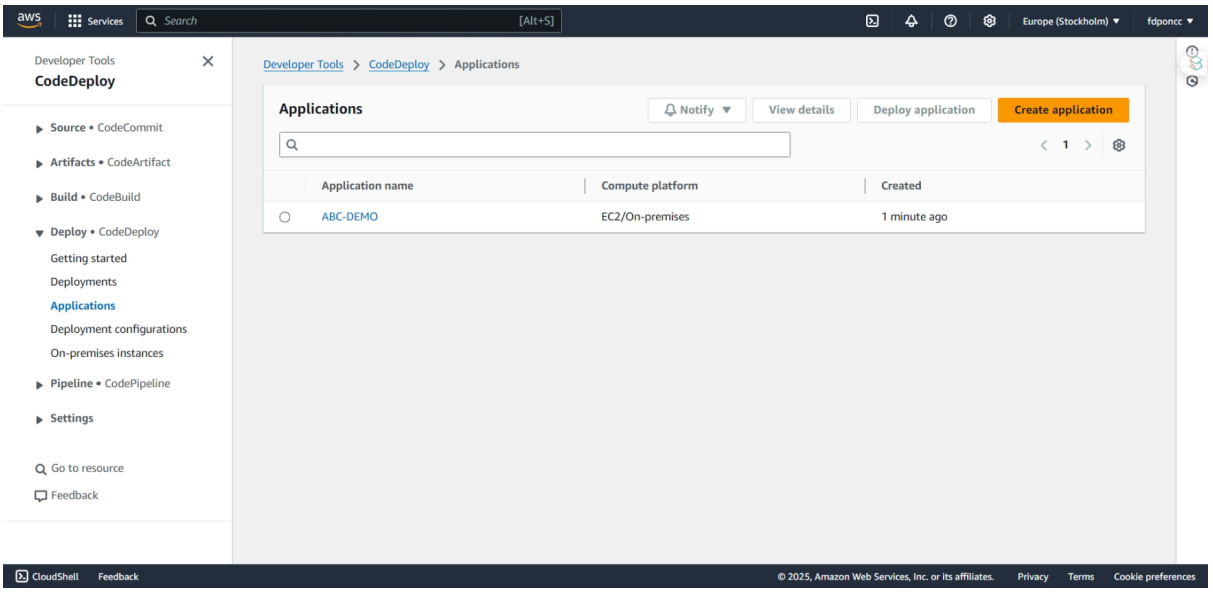


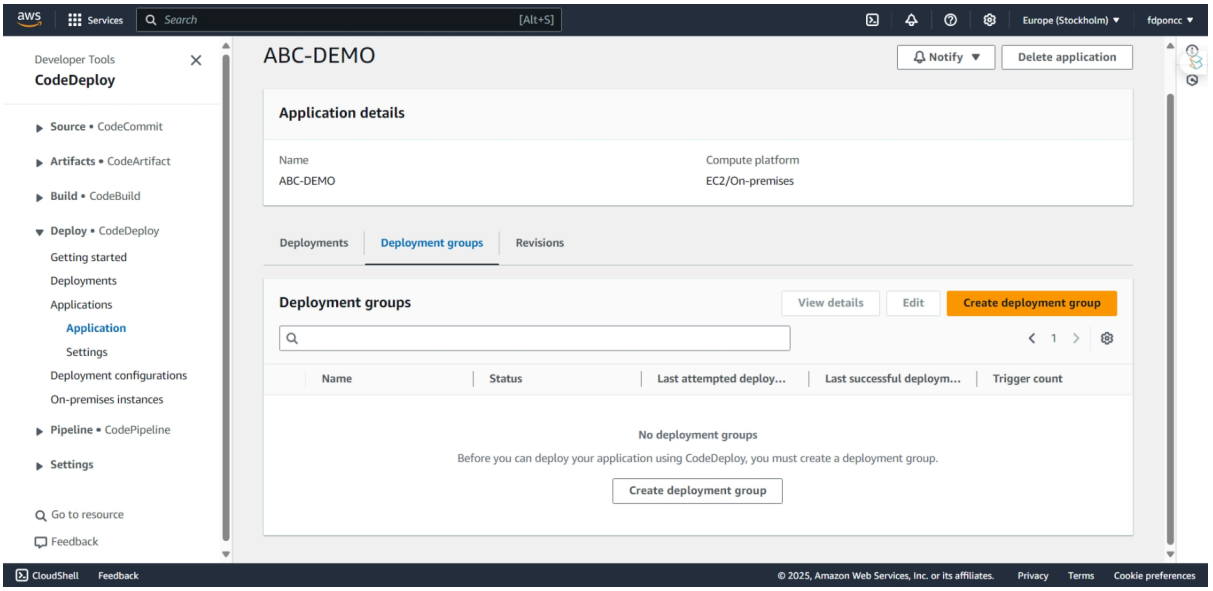
CONNECT

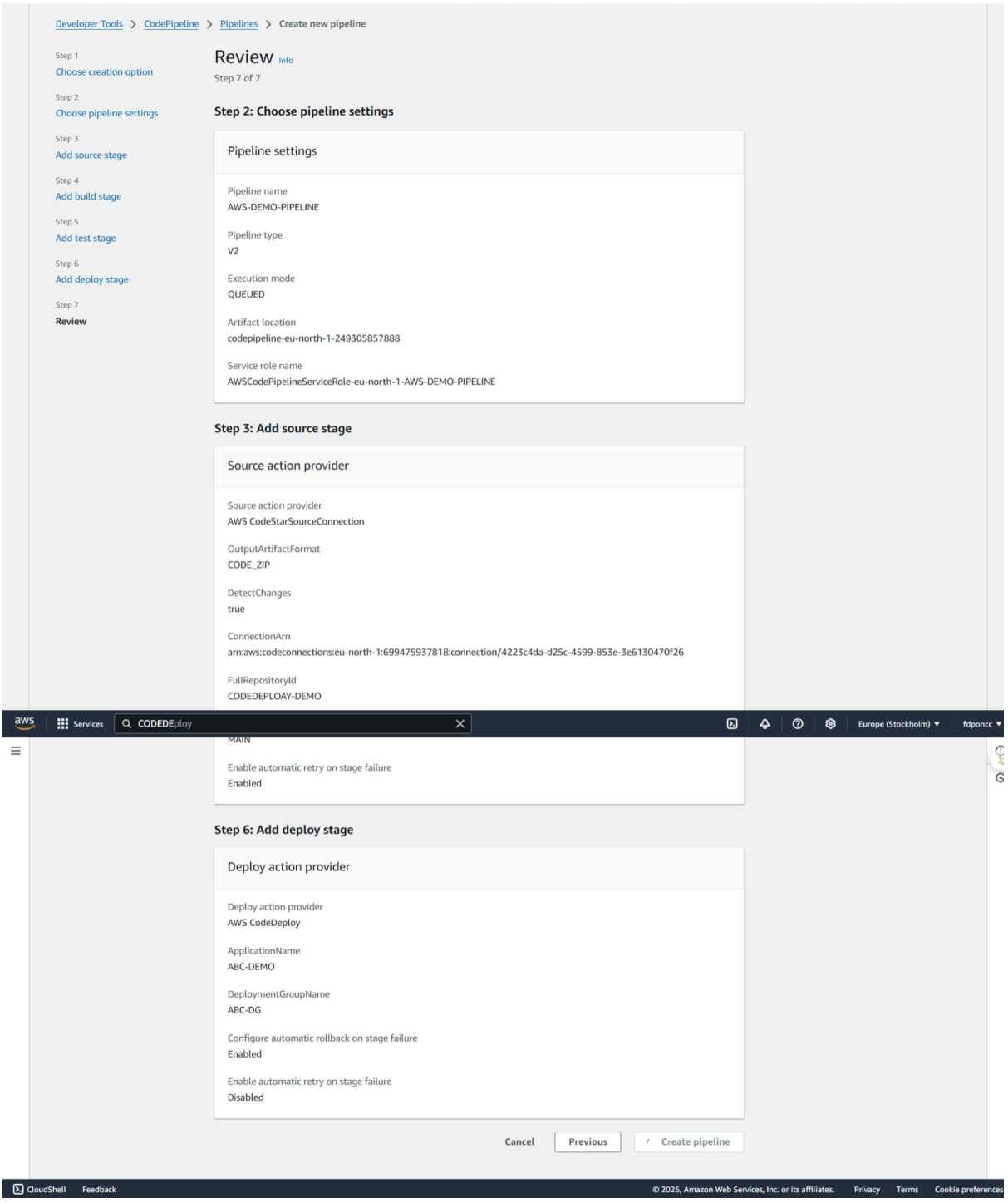

Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose pipeline settings
Step 3
Add source stage
Step 4
Add build stage
Step 5
Add test stage
Step 6
Add deploy stage
Step 7
Review
Add source stage Info
Step 3 of 7
Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (via GitHub App)
Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.
arn:aws:codeconnections:eu-north-1:699475937818:connection/42
or
Connect to GitHub
Repository name
Choose a repository in your GitHub account.
CODEDEPLOAY-DEMO
An unspecified error occurred. Check your network connectivity, and then check to see if there are any issues with the service at the Service Health Dashboard. (Click here to retry)
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
aws
Services
Search
[Alt+S]
Europe (Stockholm)
fdponcc
Not Found (Click here to retry)
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. Learn more
Enable automatic retry on stage failure
Webhook events
Webhook - optional
Start your pipeline on push and pull request events.
Webhook event filters - optional
Starts your pipeline on a specific event.
Remove filters
Cancel
Previous
Next
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



NEW CONSOLE CREATE APPLICATION AND DEPLOYEMENT GROUP




aws
Services
Search
[Alt+S]
Europe (Stockholm)
fdponcc
Developer Tools
CodeDeploy
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Applications
Application
Settings
Deployment configurations
On-premises instances
Pipeline • CodePipeline
Settings
Go to resource
Feedback
ABC-DEMO
Notify
Delete application
Application details
Name
ABC-DEMO
Compute platform
EC2/On-premises
Deployments
Deployment groups
Revisions
Deployment groups
View details
Edit
Create deployment group
1
Name
Status
Last attempted deploy...
Last successful deploym...
Trigger count
No deployment groups
Before you can deploy your application using CodeDeploy, you must create a deployment group.
Create deployment group
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose pipeline settings
Step 3
Add source stage
Step 4
Add build stage
Step 5
Add test stage
Step 6
Add deploy stage
Step 7
Review
Review Info
Step 7 of 7
Step 2: Choose pipeline settings
Pipeline settings
Pipeline name
AWS-DEMO-PIPELINE
Pipeline type
V2
Execution mode
QUEUED
Artifact location
codepipeline-eu-north-1-249305857888
Service role name
AWSCodePipelineServiceRole-eu-north-1-AWS-DEMO-PIPELINE
Step 3: Add source stage
Source action provider
Source action provider
AWS CodeStarSourceConnection
OutputArtifactFormat
CODE_ZIP
DetectChanges
true
ConnectionArn
arn:aws:codeconnections:eu-north-1:699475937818:connection/4223c4da-d25c-4599-853e-3e6130470f26
FullRepositoryId
CODEDEPLOAY-DEMO
aws
Services
CODEDEploy
Europe (Stockholm)
fdponcc
MAIN
Enable automatic retry on stage failure
Enabled
Step 6: Add deploy stage
Deploy action provider
Deploy action provider
AWS CodeDeploy
ApplicationName
ABC-DEMO
DeploymentGroupName
ABC-DG
Configure automatic rollback on stage failure
Enabled
Enable automatic retry on stage failure
Disabled
Cancel
Previous
Create pipeline
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences