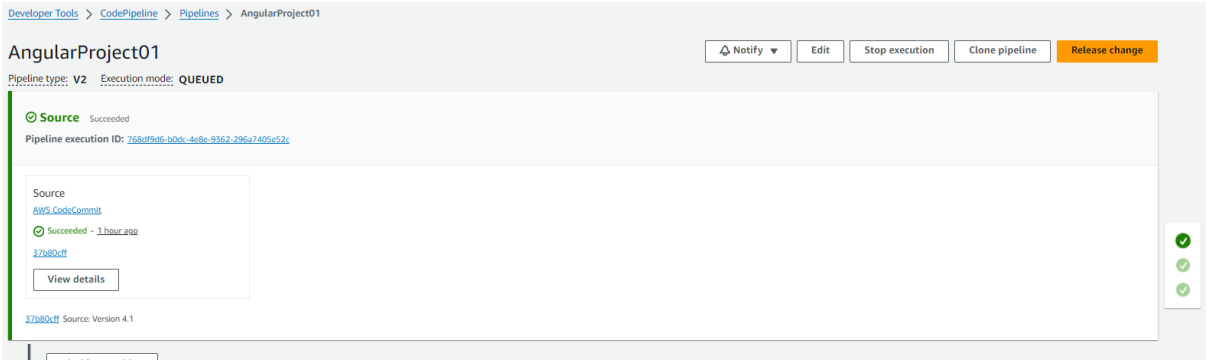
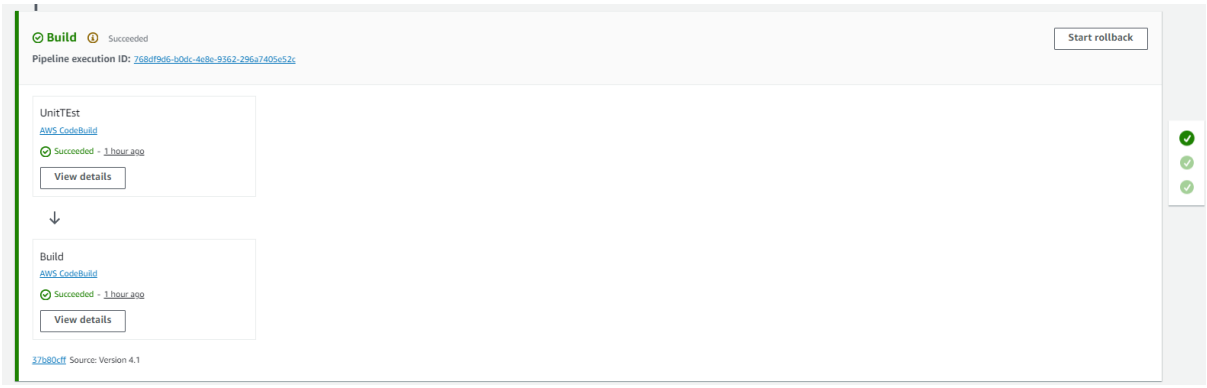
# 😊 Configuring EC2 instance for AWS Code Deploy

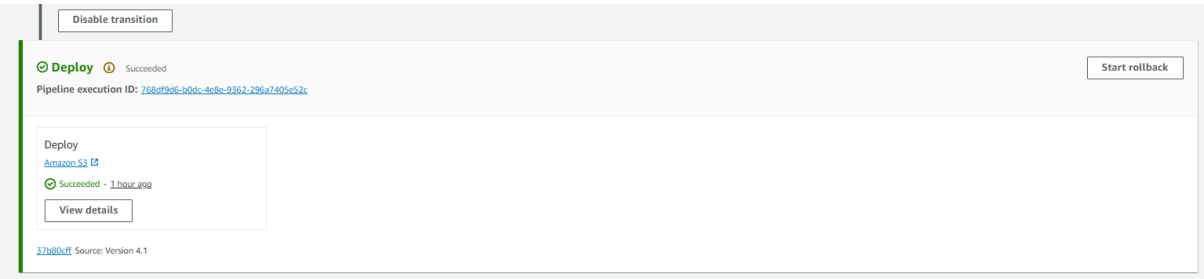
This tutorial explains how to set up AWS CodeDeploy and configure an EC2 instance to automate the deployment of an Angular project. Creating an IAM role, starting and configuring an EC2 instance, installing the required software, and configuring Nginx as a web server are the steps involved in the procedure. Along with setting up a CodeDeploy application and deployment group, it also covers connecting these with a continuous integration and delivery pipeline and generating an appspec.yml file for deployment management. The last phase involves starting the pipeline, pushing code modifications to a repository, and effectively deploying the application to the EC2 instance. The main goal is to make it possible for the Angular project to be deployed smoothly and automatically onto an EC2 server, ensuring that updates are applied effectively and without manual intervention.

## 😄 To begin with the Lab:

This lab has some prerequisites, and you should have your pipeline running from the previous labs because we are using the same Angular Project application. In case you have deleted your pipeline then recreate them. You can see below that our pipeline is up.

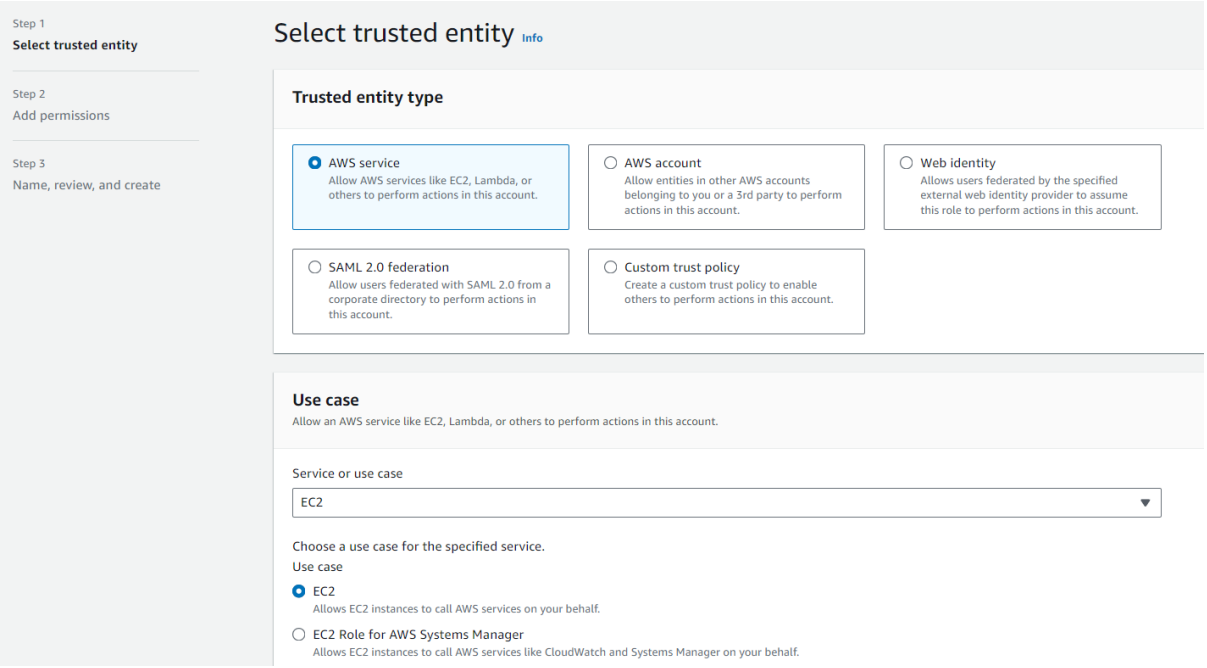


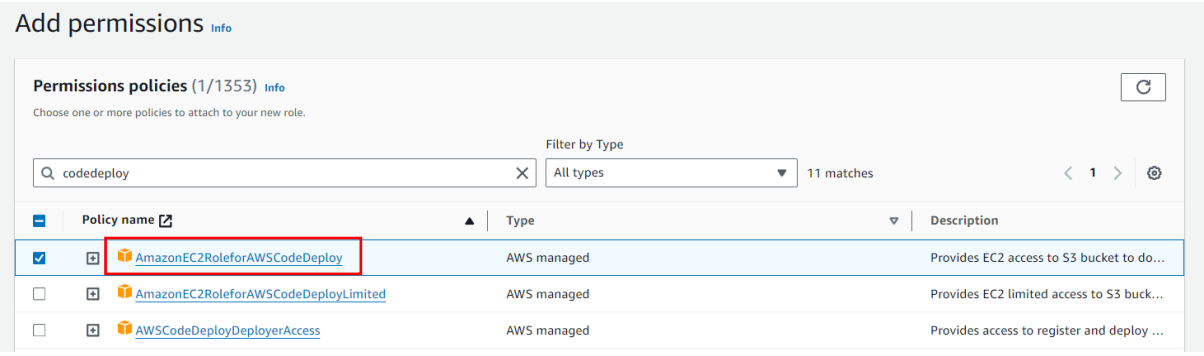



## 😁 Step 1: Creating an IAM Role

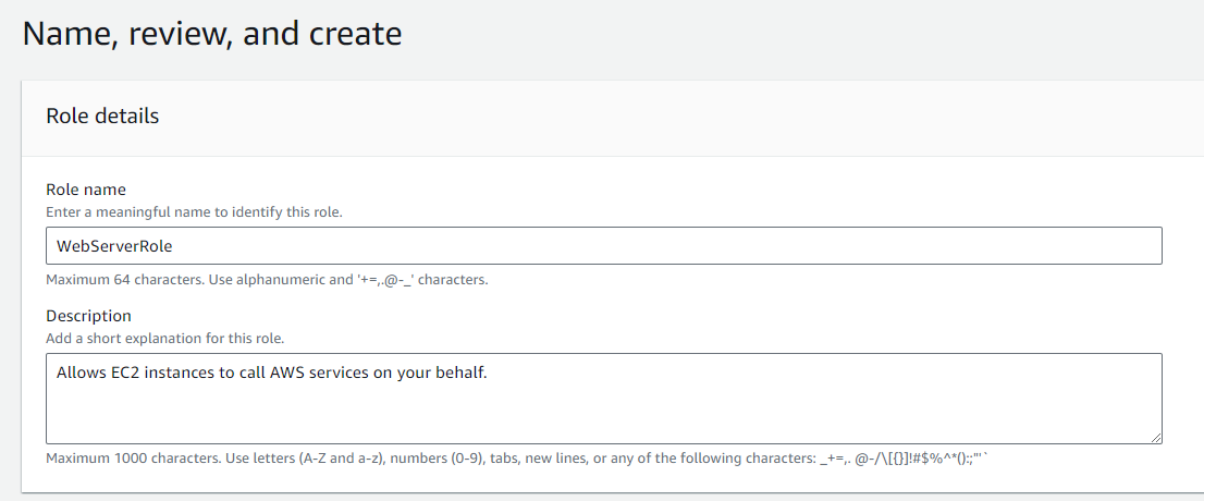
1. In your AWS Console go to IAM and then go to roles and click on create roles.
2. In step 1 you have to choose AWS service as your trusted entity and in the use case choose EC2.



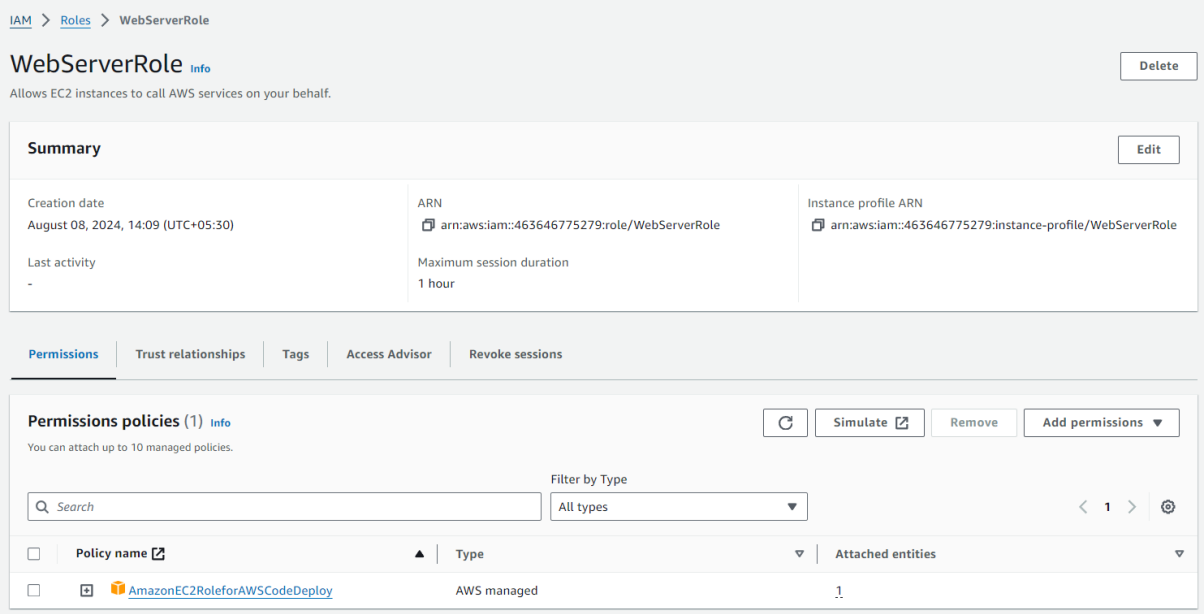

3. Then you need to search for code deploy in the permission policies and choose the same policy as shown below.




4. After that name your role and click on create role.

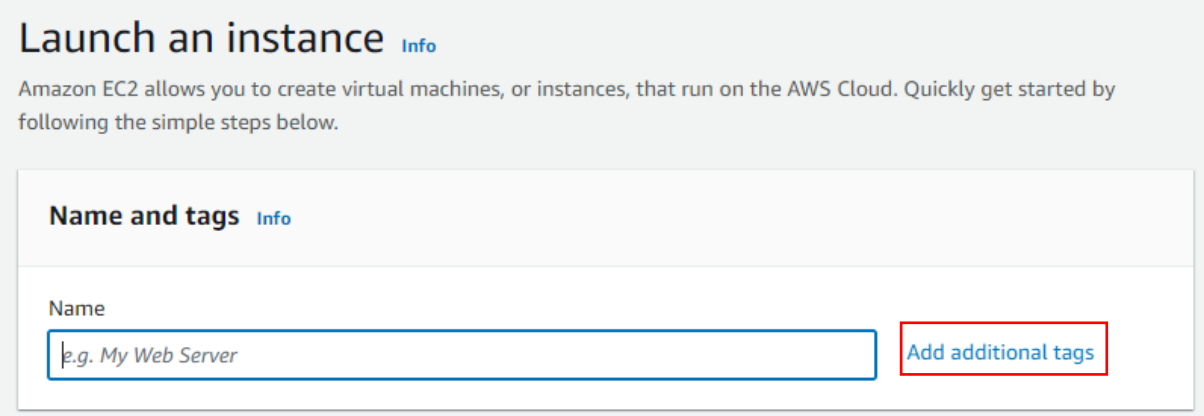



5. Below you can see that our role has been created successfully.

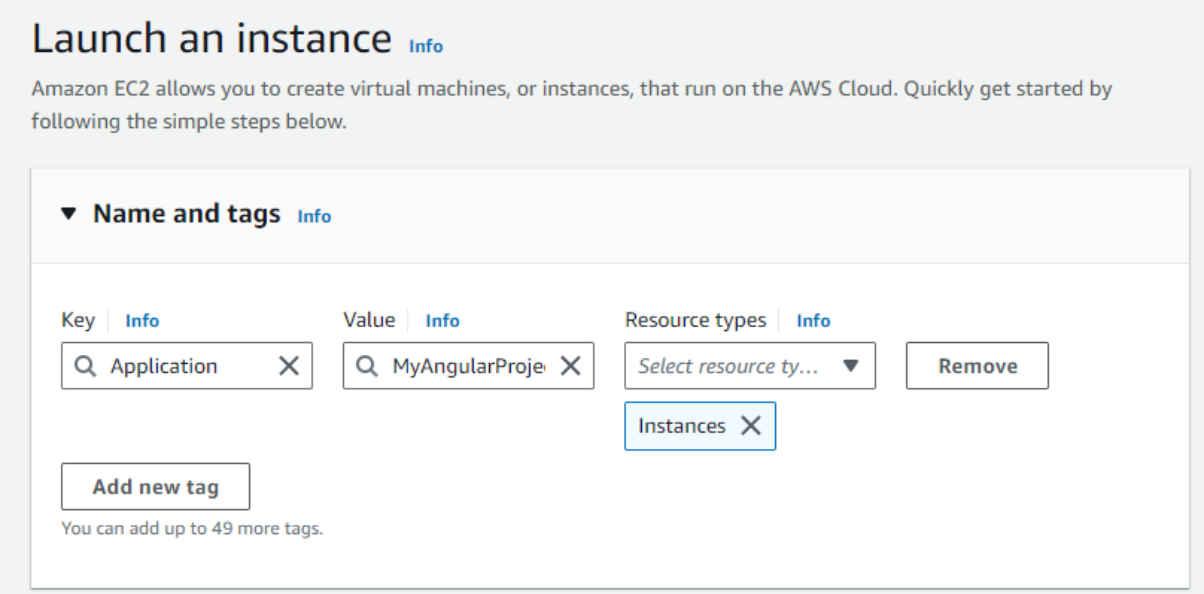



## 😁 Step 2: Configuring EC2 instance for Code Deploy

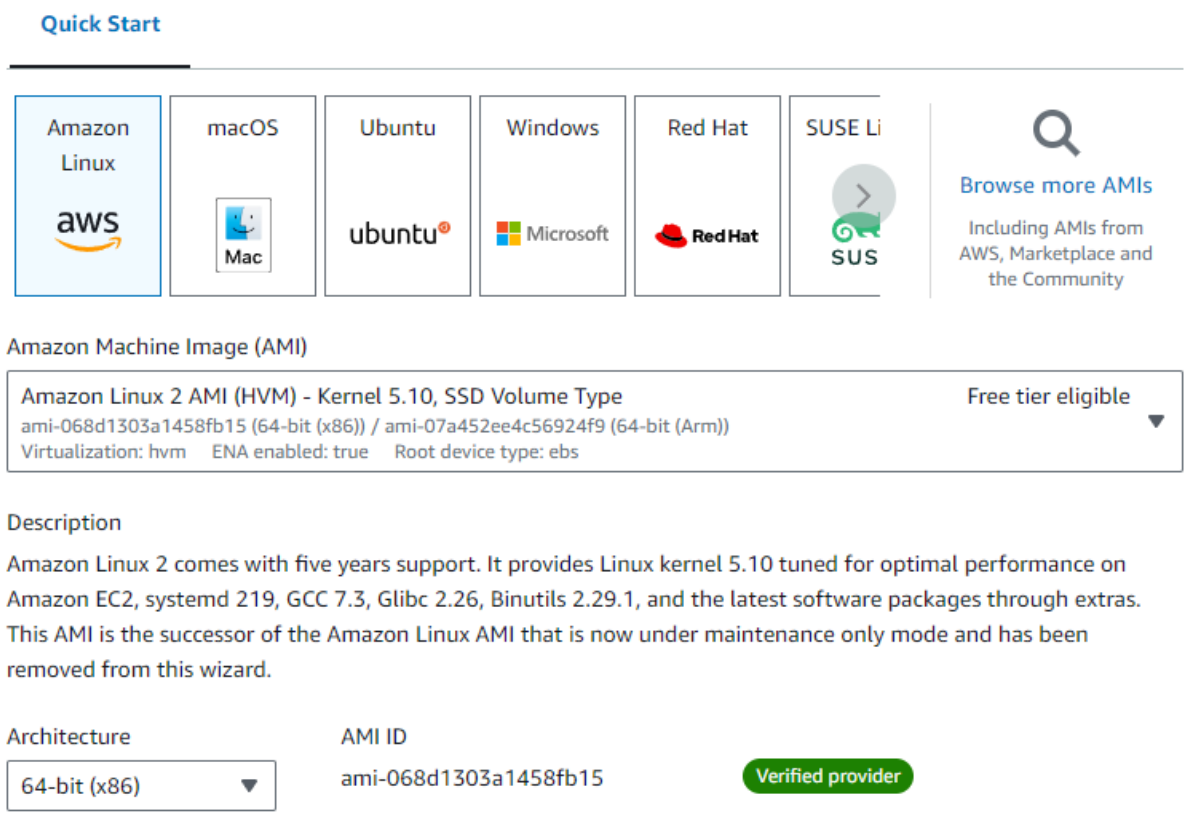
1. Now navigate to EC2 and click on Launch Instance to create one. Below you can see that here we have to give a name to our instance, but this time we will provide this instance a tag for AWS Code Deploy. To give it a tag you need to click on Add additional tags which has been highlighted in the snapshot.



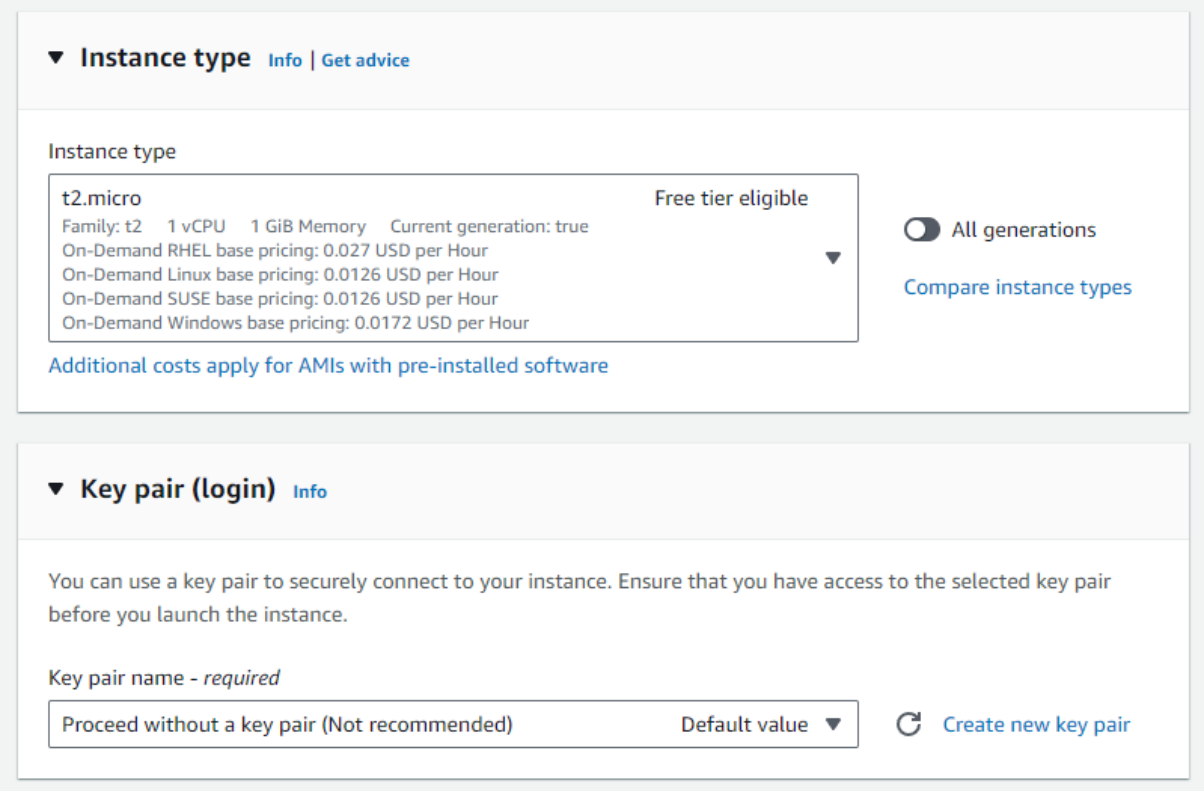

2. Below you can see that now you have the ability to give a key and a value. Give the same key and value if you wish to. Then scroll down.



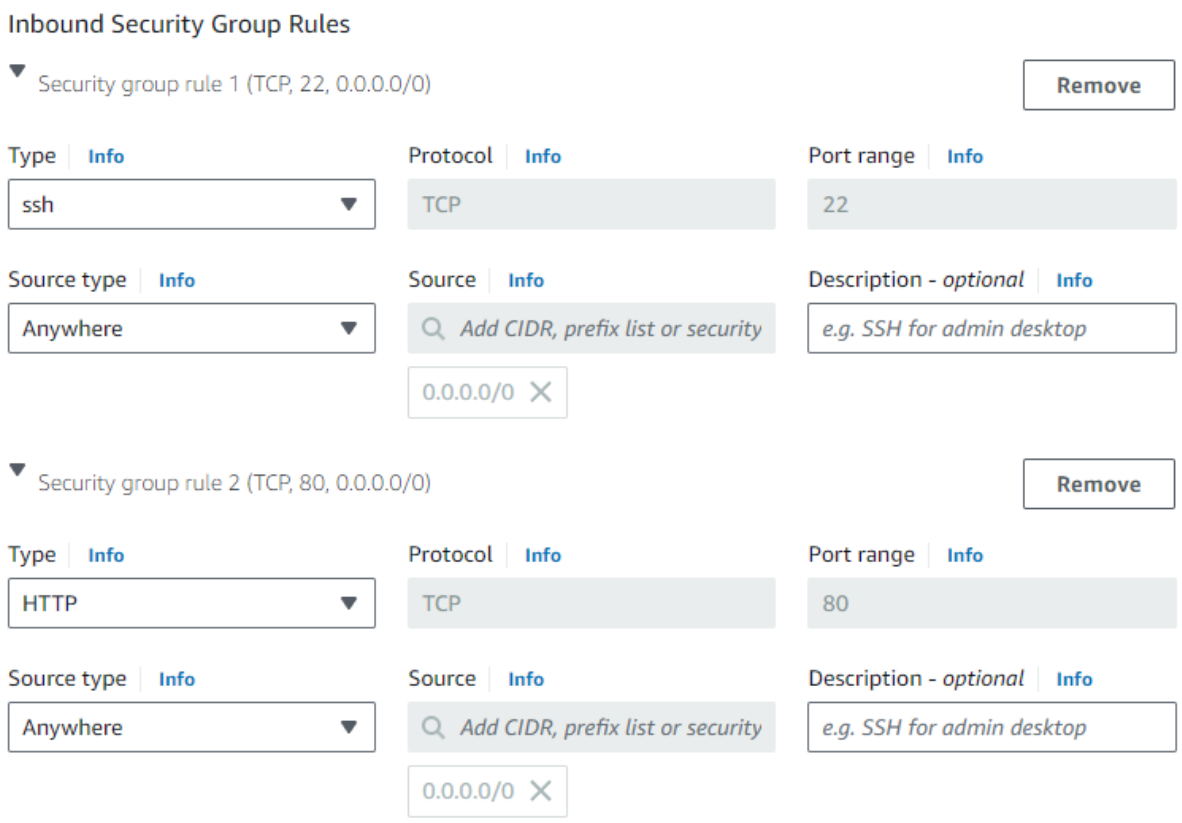

3. Then in the application and OS section you need to choose Amazon Linux as your OS and in AMI choose Amazon Linux 2.




4. Then in the instance type choose t2.micro and for the key pair choose proceed without a key pair or if you want to then you can create a key pair.

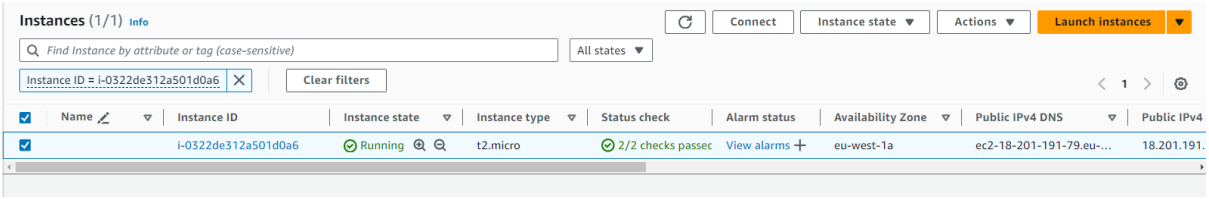


5. Now in the networking section you just need to create a new security group and in the inbound security group rules add a rule for port 80 HTTP.

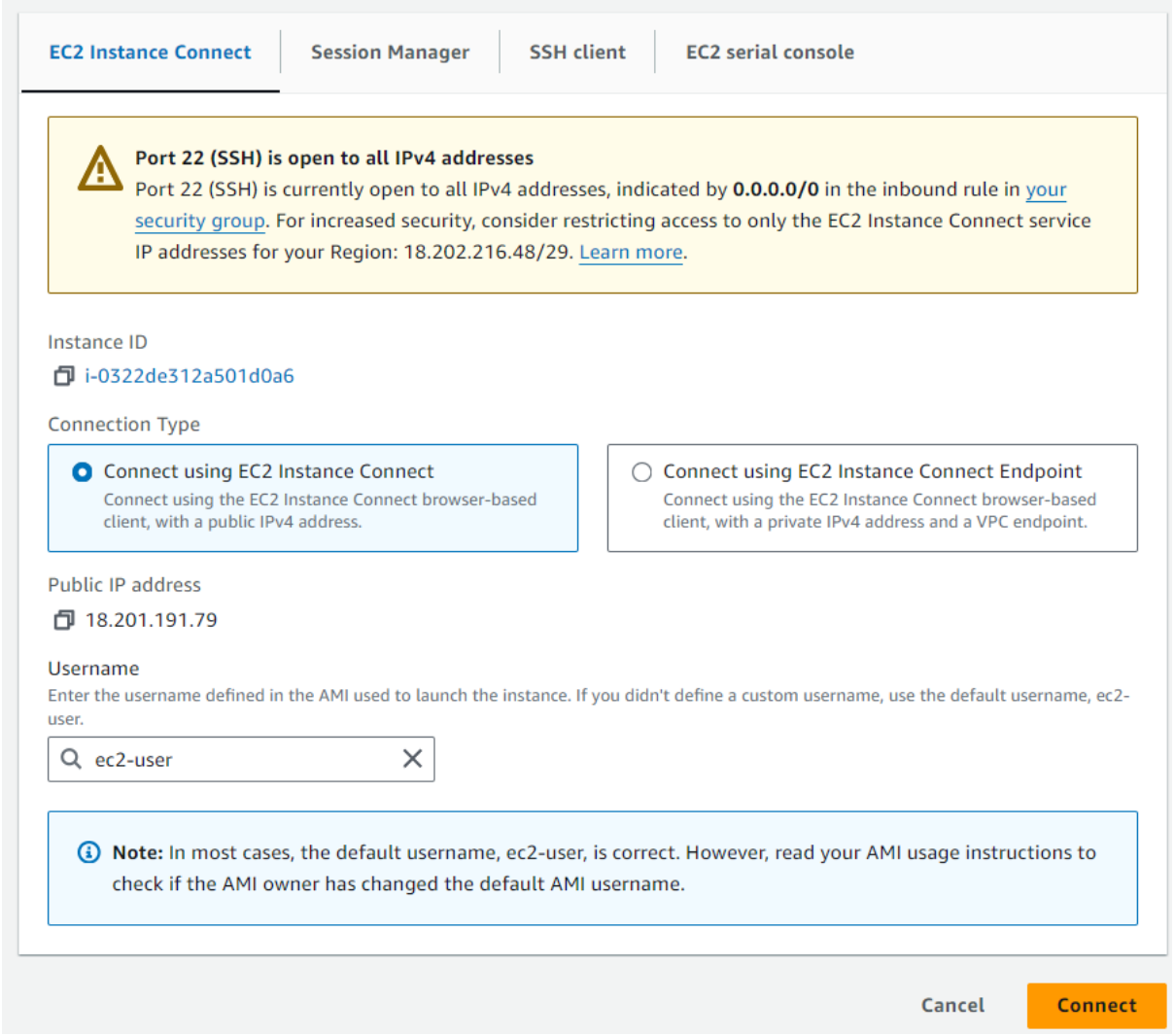



6. After that you need to expand the advanced details and choose your IAM role, this is very important do not forget this. Then you can click on launch instance.

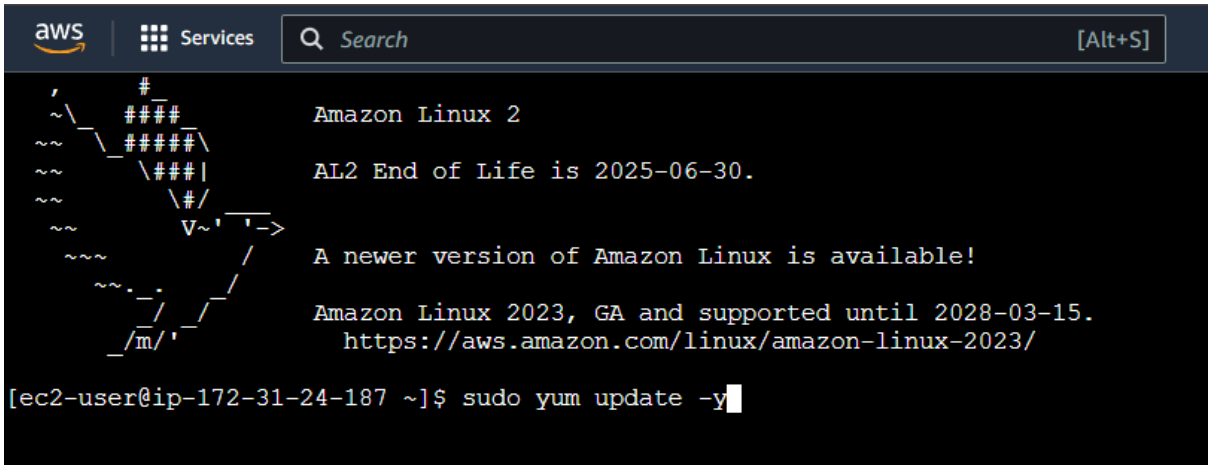

7. Once your instance is launched then you need to connect it. So, below you can see that our instance is up and running. Now we need to click on connect.




8. Choose EC2 instance connect click on connect.




9. Here you can see that we have connected with our instance. Firstly, we will update the packages on the instance using the 'sudo yum update' command with a '-y' option to approve all updates automatically.




10. Next, we will use the 'yum install' command again with the 'sudo' and the '-y' option to install the 'ruby' and 'wget' packages, which are necessary for the Code Deploy agent installation.
    **sudo yum install -y ruby wget**
11. Next, we will download the Code Deploy agent's installation package using the 'wget' command. It gets the package URL as input.
12. Now AWS provides this package from a different S3 bucket in each AWS region. So, its URL changes according to the AWS region used. But it has the same structure for each. So, the bucket name starts with 'aws-codedeploy' followed by a hyphen. Then comes the AWS region code, which is 'eu-west-1' for the Ireland region we use. And then, you provide the 's3' subdomain under this AWS region. And in this S3 bucket, the package is placed under the 'latest' folder with the 'install' file name. Please don't miss the region codes in the URL. They are crucial.

    **wget https://aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com/latest/install**


13. Let's use the 'ls' command with the '-la' option to list all files in the current folder in the long format. The install file was downloaded to the current directory with its owner as the 'ec2-user', the user we logged in as. But as you see, the package isn't executable yet. It only has read and write permissions. Therefore, let's use the 'chmod' command and provide executing permissions with '+x' for the 'install' file in the current directory where the file was downloaded.

```
[ec2-user@ip-172-31-24-187 ~]$ ls -la
total 32
drwx------ 3 ec2-user ec2-user    89 Aug  8 09:07 .
drwxr-xr-x 3 root     root        22 Aug  8 08:53 ..
-rw-r--r-- 1 ec2-user ec2-user    18 Jul 15  2020 .bash_logout
-rw-r--r-- 1 ec2-user ec2-user   193 Jul 15  2020 .bash_profile
-rw-r--r-- 1 ec2-user ec2-user   231 Jul 15  2020 .bashrc
-rw-rw-r-- 1 ec2-user ec2-user 19045 Feb 21 19:20 install
drwx------ 2 ec2-user ec2-user    29 Aug  8 08:53 .ssh
[ec2-user@ip-172-31-24-187 ~]$ chmod +x ./install
[ec2-user@ip-172-31-24-187 ~]$ ls -la
total 32
drwx------ 3 ec2-user ec2-user    89 Aug  8 09:07 .
drwxr-xr-x 3 root     root        22 Aug  8 08:53 ..
-rw-r--r-- 1 ec2-user ec2-user    18 Jul 15  2020 .bash_logout
-rw-r--r-- 1 ec2-user ec2-user   193 Jul 15  2020 .bash_profile
-rw-r--r-- 1 ec2-user ec2-user   231 Jul 15  2020 .bashrc
-rwxrwxr-x 1 ec2-user ec2-user 19045 Feb 21 19:20 install
drwx------ 2 ec2-user ec2-user    29 Aug  8 08:53 .ssh
[ec2-user@ip-172-31-24-187 ~]$
```

14. And finally, we can install the Code Deploy agent by using 'sudo' to get superuser privileges and executing the 'install' file by providing its path. You should also provide the 'auto' subcommand for automatic installation.

    **sudo ./install auto**

15. Now we will check whether our code deploy agent is running or not, for that we will run this command.

    **Sudo service codedeploy-agent status**

```
aws | Services | Search [Alt+S]
[ec2-user@ip-172-31-24-187 ~]$ sudo service codedeploy-agent status
The AWS CodeDeploy agent is running as PID 3906
[ec2-user@ip-172-31-24-187 ~]$
```

16. Now we need to install a web server application on our instance and for this lab we will install Nginx. So, to install it we need to run this command.

    **sudo amazon-linux-extras install -y nginx1**

17. Once Nginx is installed, we will check its status for that we need to run this command and if our Nginx is inactive then we can run the to start this service.

    sudo service nginx status
    sudo service nginx start

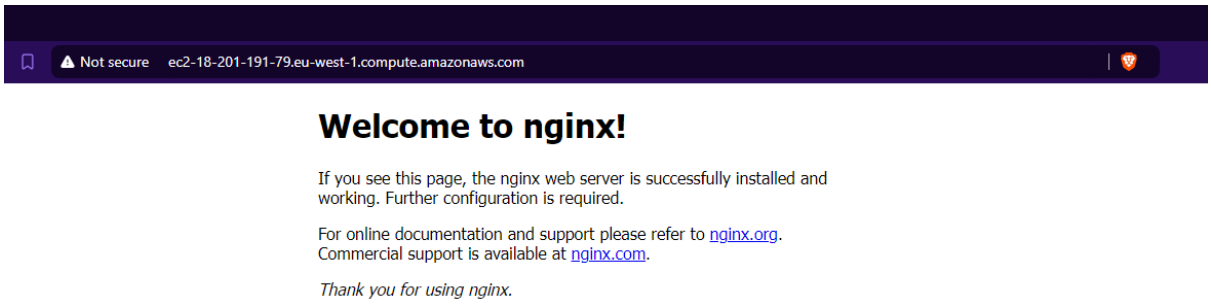
```
[ec2-user@ip-172-31-24-187 ~]$ sudo service nginx status
Redirecting to /bin/systemctl status nginx.service
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
[ec2-user@ip-172-31-24-187 ~]$ sudo service nginx start
Redirecting to /bin/systemctl start nginx.service
[ec2-user@ip-172-31-24-187 ~]$ sudo service nginx status
Redirecting to /bin/systemctl status nginx.service
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2024-08-08 09:21:05 UTC; 4s ago
  Process: 4191 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 4187 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 4186 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
 Main PID: 4193 (nginx)
   CGroup: /system.slice/nginx.service
           ├─4193 nginx: master process /usr/sbin/nginx
           └─4194 nginx: worker process

Aug 08 09:21:05 ip-172-31-24-187.eu-west-1.compute.internal systemd[1]: Starting The nginx HTTP and reverse proxy server...
Aug 08 09:21:05 ip-172-31-24-187.eu-west-1.compute.internal nginx[4187]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Aug 08 09:21:05 ip-172-31-24-187.eu-west-1.compute.internal nginx[4187]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Aug 08 09:21:05 ip-172-31-24-187.eu-west-1.compute.internal systemd[1]: Started The nginx HTTP and reverse proxy server.
[ec2-user@ip-172-31-24-187 ~]$
```

18. As you see, the Nginx service is disabled. So, we must also ensure that Nginx starts on EC2 instance reboots. You run the 'chkconfig' command with 'sudo' for this, provide the 'nginx' service, and then the 'on' parameter to enable it.

    **sudo chkconfig nginx on**

```
[ec2-user@ip-172-31-24-187 ~]$
[ec2-user@ip-172-31-24-187 ~]$ sudo chkconfig nginx on
Note: Forwarding request to 'systemctl enable nginx.service'.
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /usr/lib/systemd/system/nginx.service.
[ec2-user@ip-172-31-24-187 ~]$ sudo service nginx status
Redirecting to /bin/systemctl status nginx.service
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2024-08-08 09:21:05 UTC; 2min 34s ago
 Main PID: 4193 (nginx)
   CGroup: /system.slice/nginx.service
           ├─4193 nginx: master process /usr/sbin/nginx
           └─4194 nginx: worker process

Aug 08 09:21:05 ip-172-31-24-187.eu-west-1.compute.internal systemd[1]: Starting The nginx HTTP and reverse proxy server...
Aug 08 09:21:05 ip-172-31-24-187.eu-west-1.compute.internal nginx[4187]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Aug 08 09:21:05 ip-172-31-24-187.eu-west-1.compute.internal nginx[4187]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Aug 08 09:21:05 ip-172-31-24-187.eu-west-1.compute.internal systemd[1]: Started The nginx HTTP and reverse proxy server.
[ec2-user@ip-172-31-24-187 ~]$
```

19. Now go to EC2 and select your instance then copy the IPv4 Public DNS name of your instance and paste it into a new tab or browser, here you can see that our Nginx service is up and running.




20. Now, we need to specify where we will deploy our application on the EC2 instance. So, let's create a new folder using the 'mkdir' command with 'sudo' and the '-p' option to create all missing folders recursively. Let's create it under the '/var/www' folders with a name like 'my-angular-project' in kebab-case.

    **sudo mkdir -p /var/www/my-angular-project**

21. Next, we must tell Nginx to load its default server from this folder. So, let's open the Nginx configuration file

    **sudo nano /etc/nginx/nginx.conf**

22. Now, let's find the root directive under the server configuration. We must replace it with the path of the new folder we created for our sample application.

```
GNU nano 2.9.8

    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /usr/share/nginx/html;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        error_page 404 /404.html;
        location = /404.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
    }
```

23. Below you can see that we have made the changes.

```
GNU nano 2.9.8

    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /var/www/my-angular-project;
```

24. Now we need to restart our Nginx service.

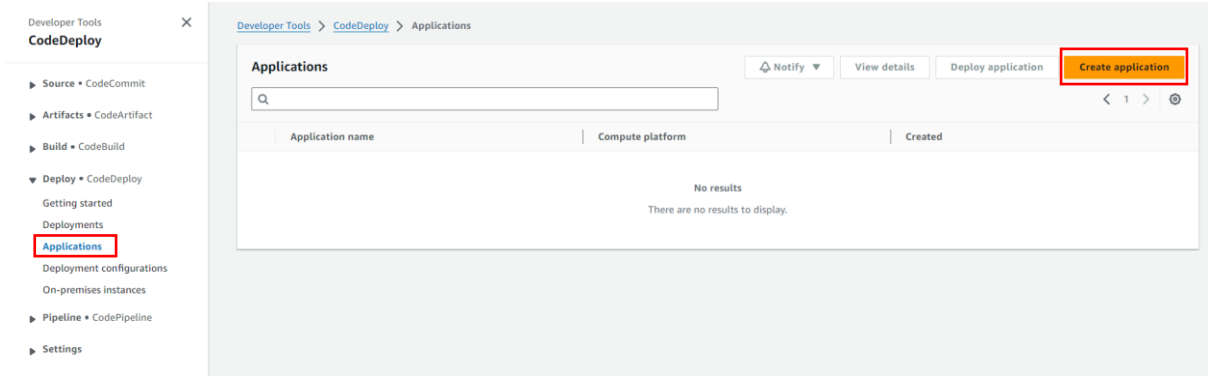
    **sudo service nginx restart**

25. After restarting it go back to the website tab and refresh the page you will see that it responds with the '403 Forbidden' error. Why is that? Well, don't worry, it is expected

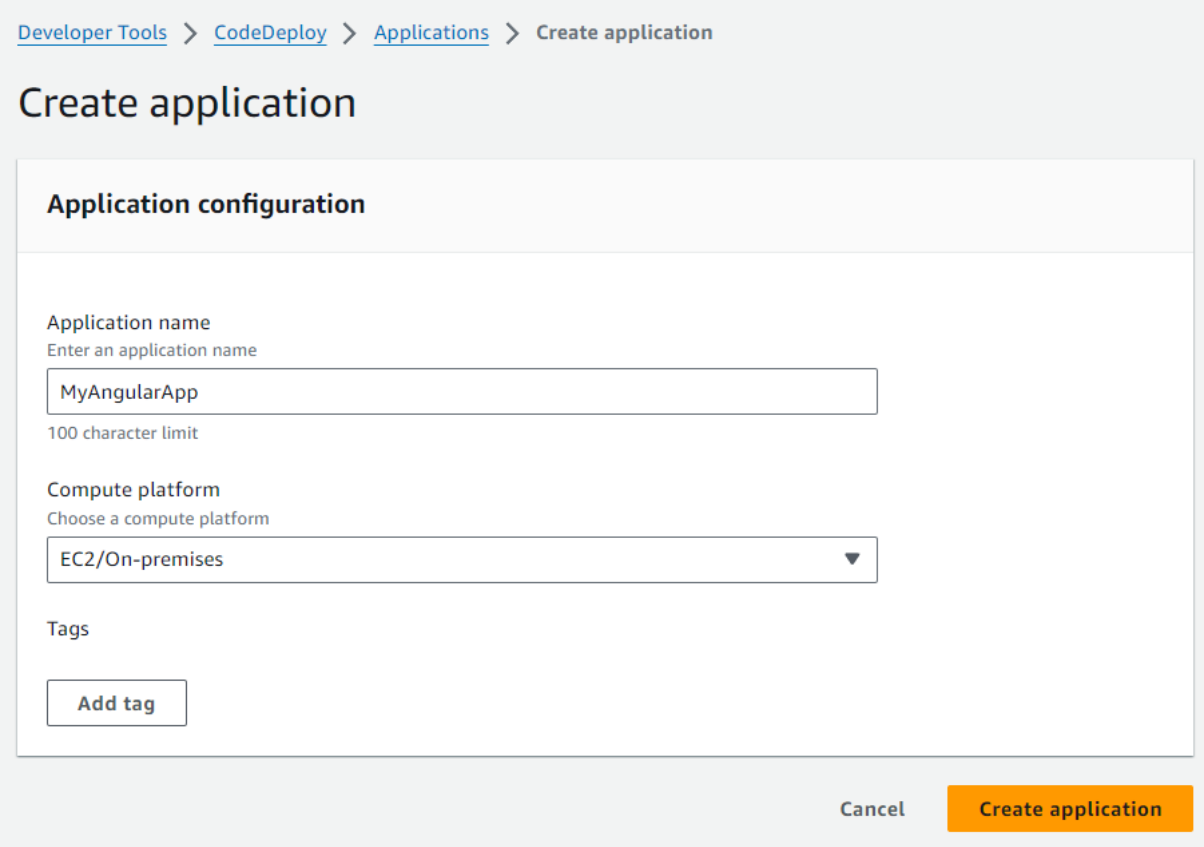
because we haven't deployed our web application yet. The deployment folder we created on the EC2 instance is still empty. Besides, it must be empty because CodeDeploy raises errors if it finds files in the deployment folder that it didn't deploy before.


## 😁 Step 3: Creating Code Deploy

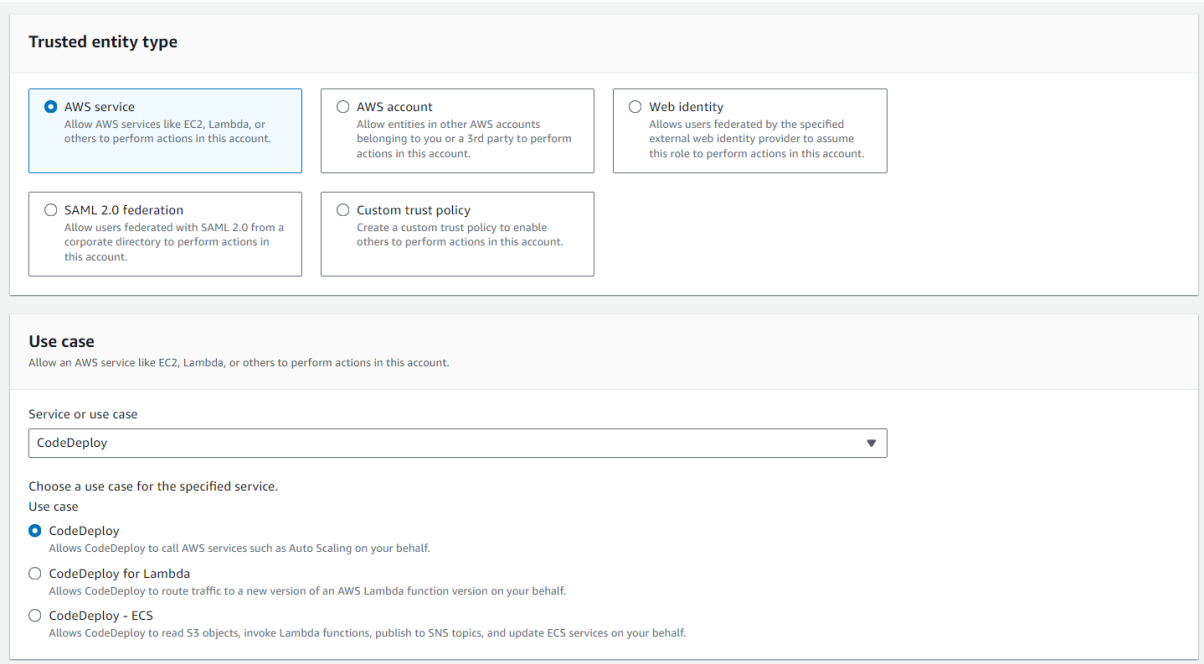
1. So, in this lecture, we will create a Code Deploy application, a Code Deploy service role for EC2 deployments, and a deployment group for this tag. Therefore, our EC2 instance will be selected for new deployments.
2. Now search for code deploy and then come to applications, then you need to click on Create application.



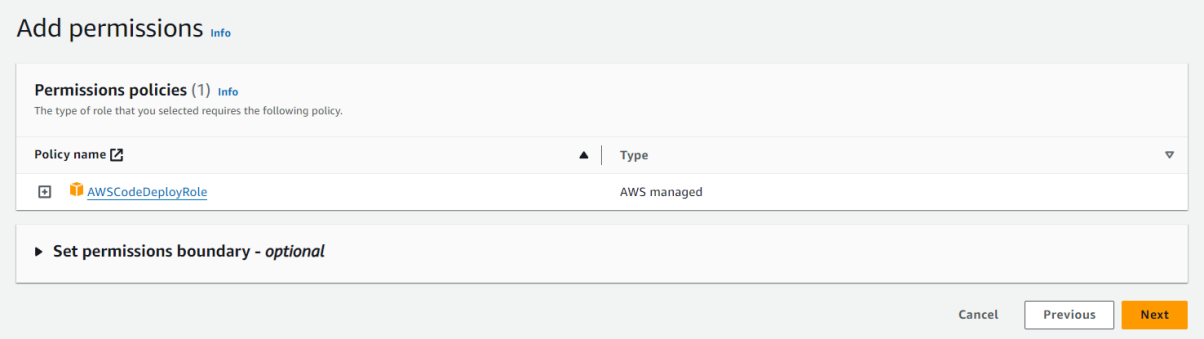

3. Now give your application a name then in the compute platform choose EC2/On-premises. Then click on Create Application.



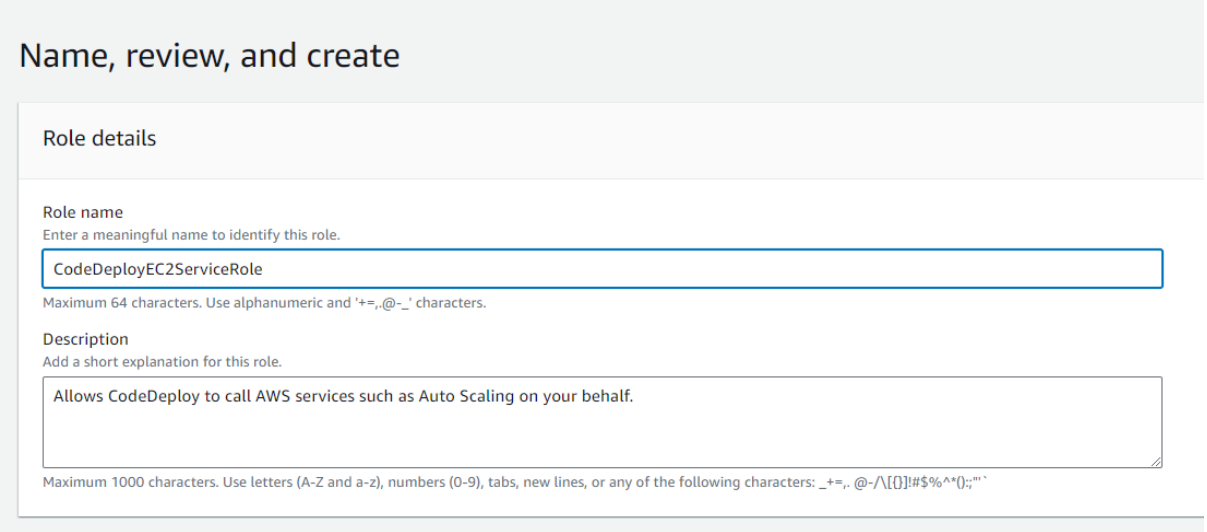

4. Once your application is created you need to create deployment groups. But before that, we need to create an IAM role. So, go to IAM and then to roles then click on Create roles.
5. Here in the use case you need to search for Code Deploy and choose it as shown below and click on next.



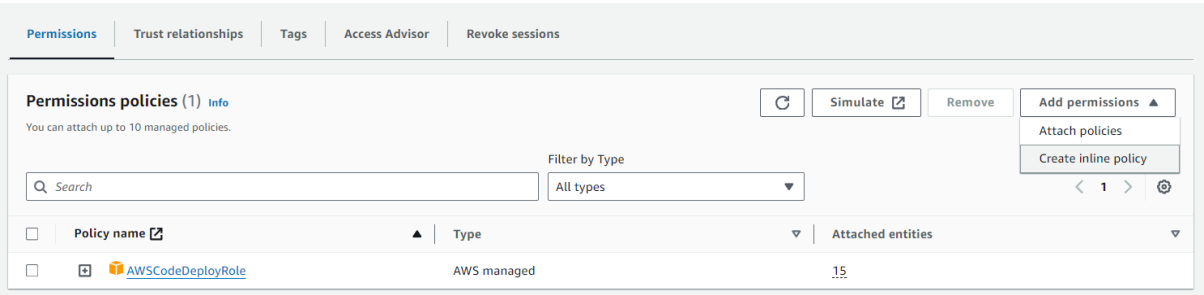

6. Then you will see that it has automatically picked up the permission policy, now click on next.




7. In the end give your role a name and then create it. Once it is created then you need to open your role to add some more permissions.




8. Now you need to click on Add permission and choose to create an inline policy.




9. Then quickly switch to JSON to edit the policy and here you need to have the same code as shown below.

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Sid": "Statement1",
                        "Effect": "Allow",
                        "Action":       ["ec2:RunInstances",       "ec2:CreateTags",
"iam:PassRole"],
```

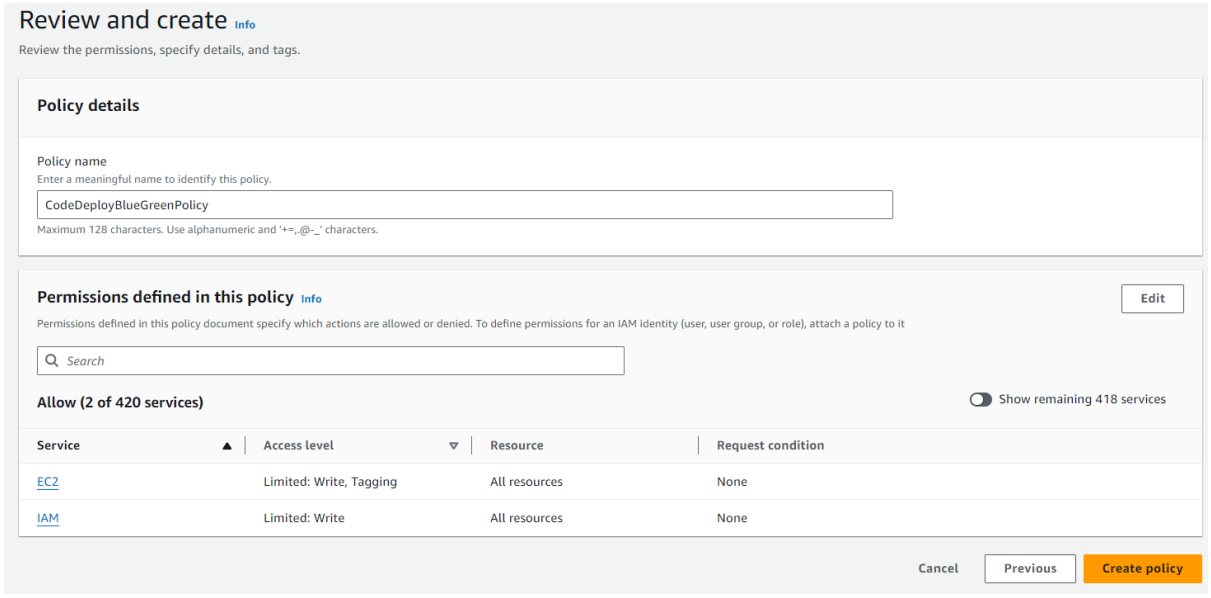
"Resource": "*"

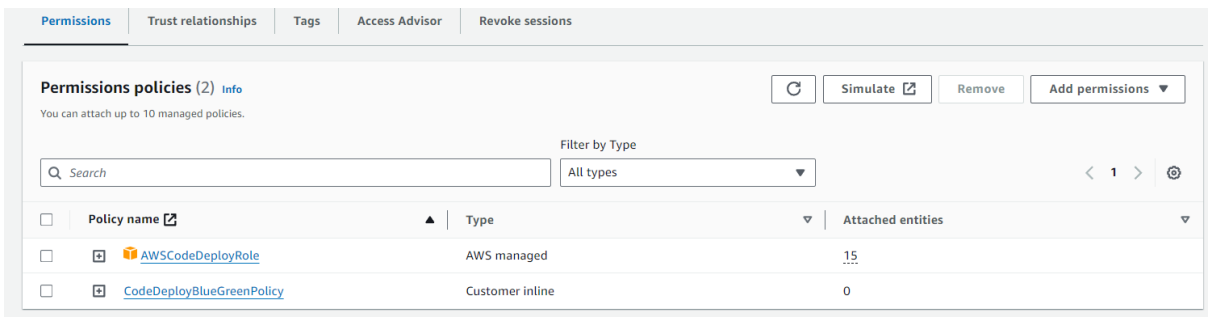
}

]

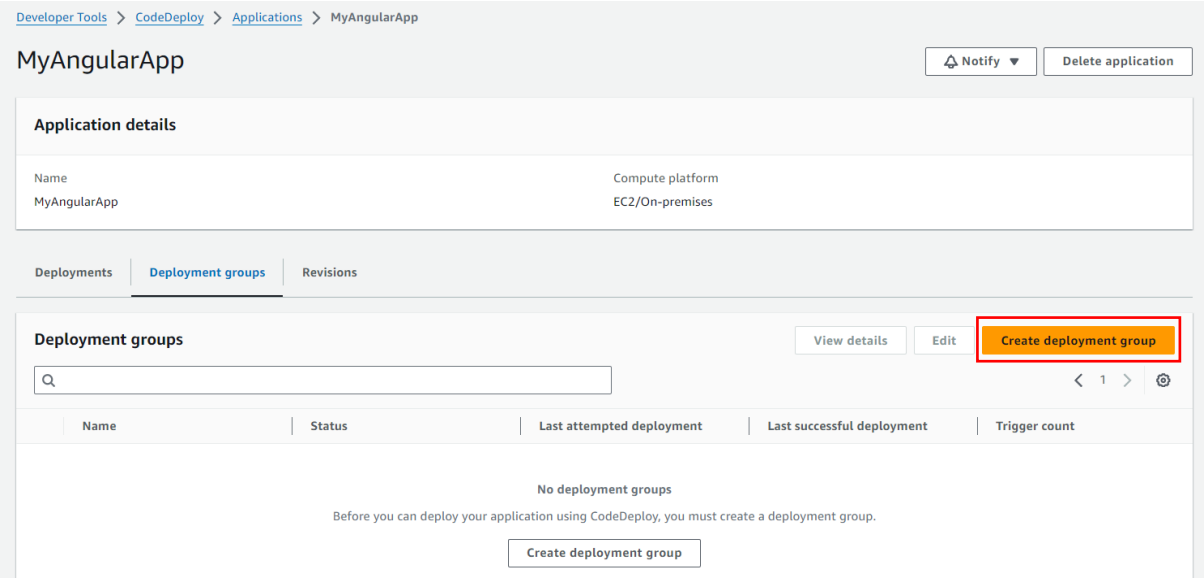
}


10. After that click on next give a name to your role and just create it.

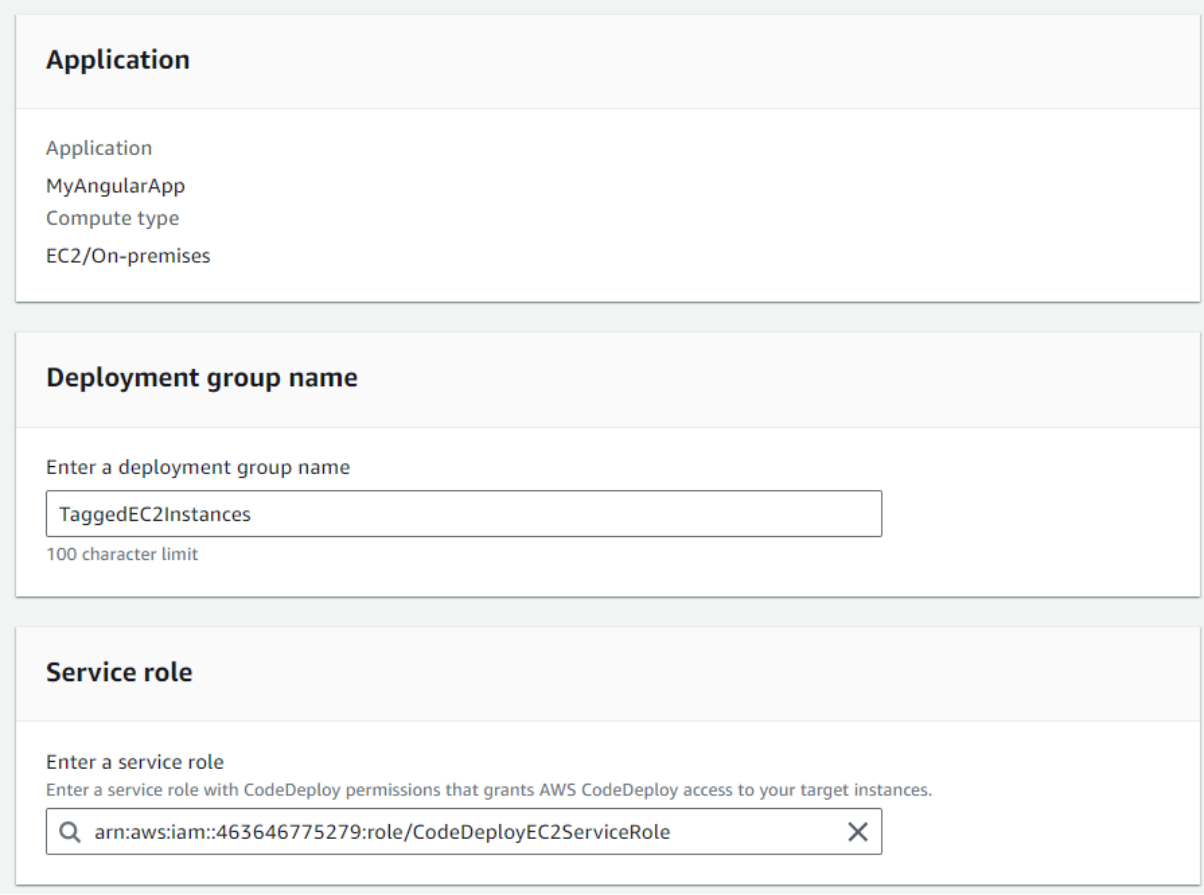


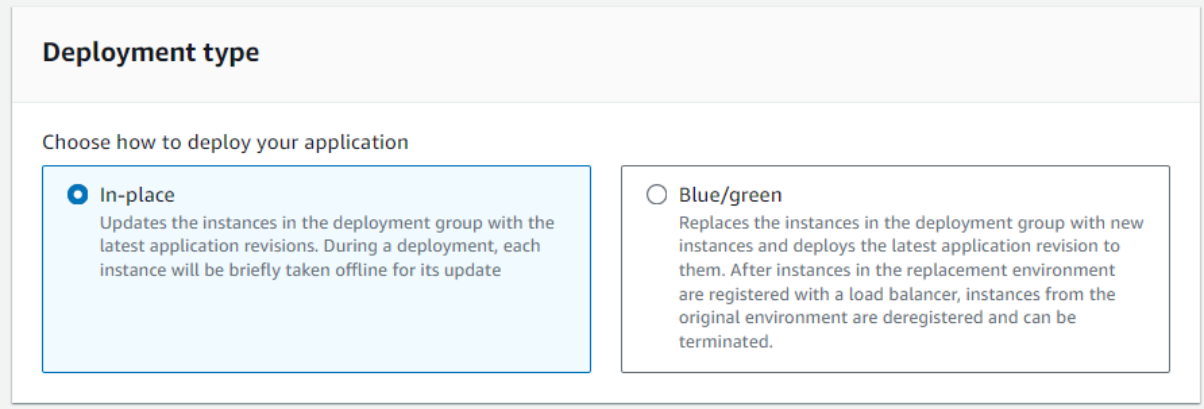
11. Now come back to your application in code deploy and click on Create deployment group.



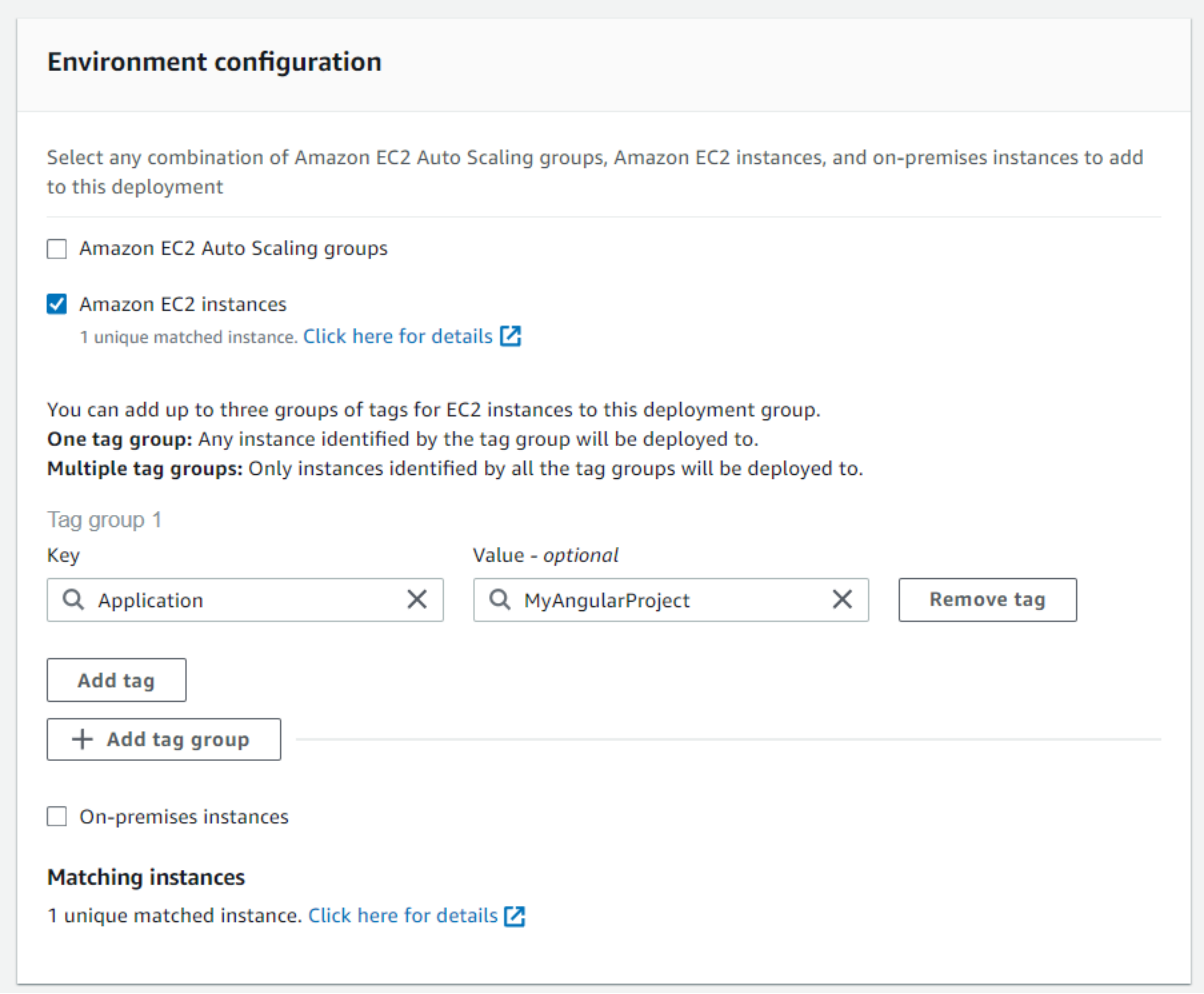

12. Here you can see that it has picked up our application and we need to provide it with a name then choose our service role which we created earlier.



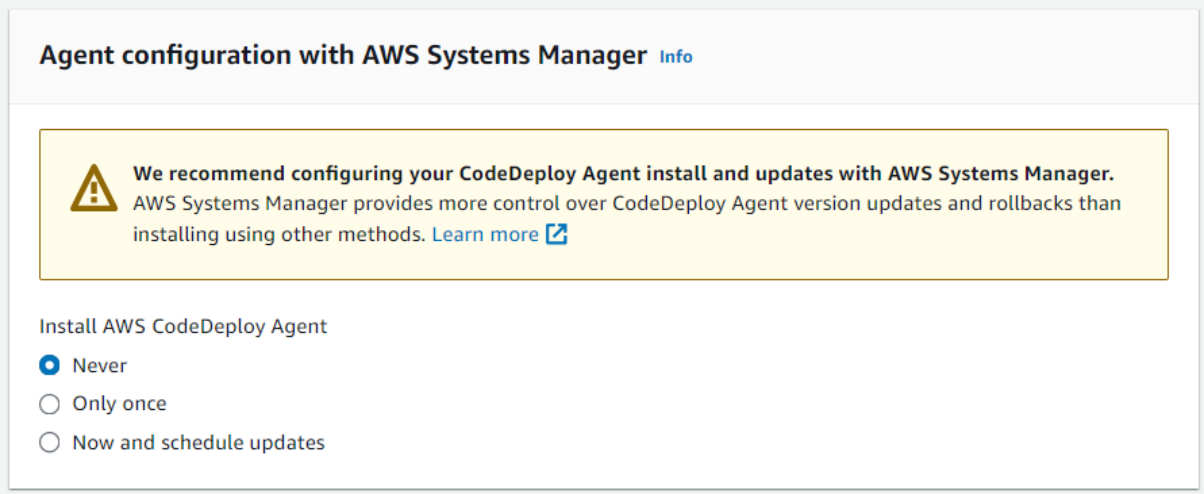

13. Then in the deployment type you need to choose In-place and move forward.



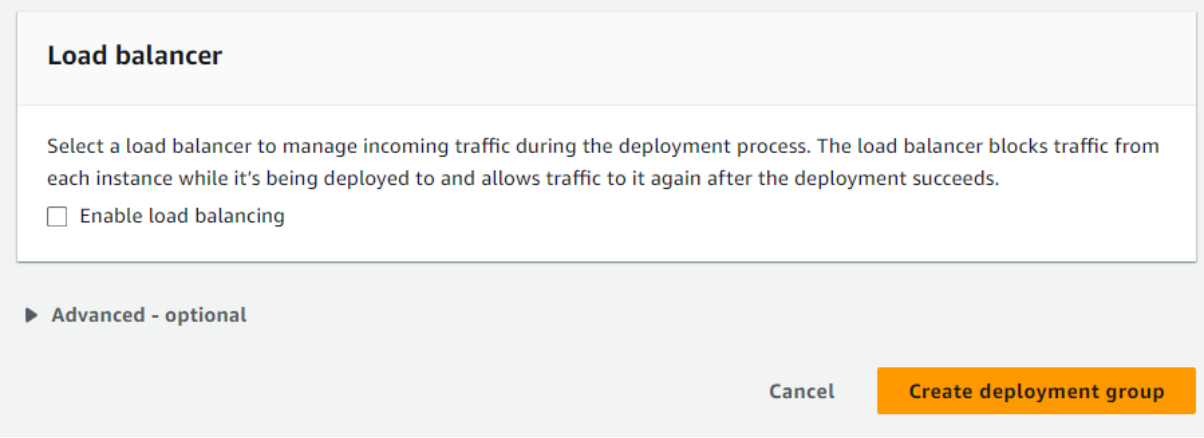

14. Now in the environment configuration you need to choose Amazon EC2 instances and then you need to choose the tag that you'd given while creating your instance.




15. Now scroll down and choose never to Install AWS Code Deploy Agent because we have already installed it on our instance.

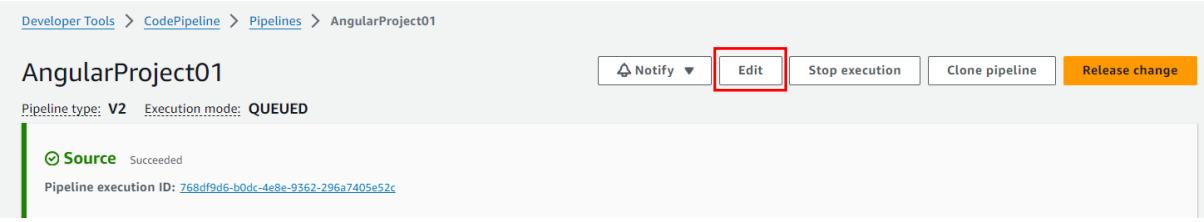



16. Then scroll down to the bottom disable load balancing and now click on Create Deployment Group.

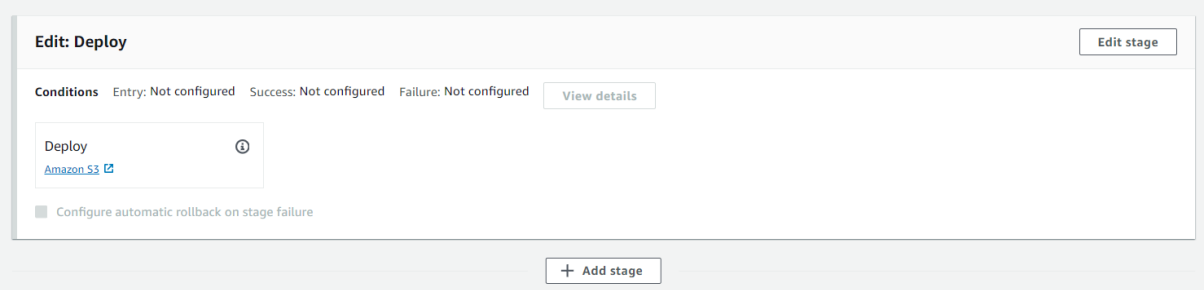



## 😁 Step 4: Adding a codedeploy deploy action to our Pipeline

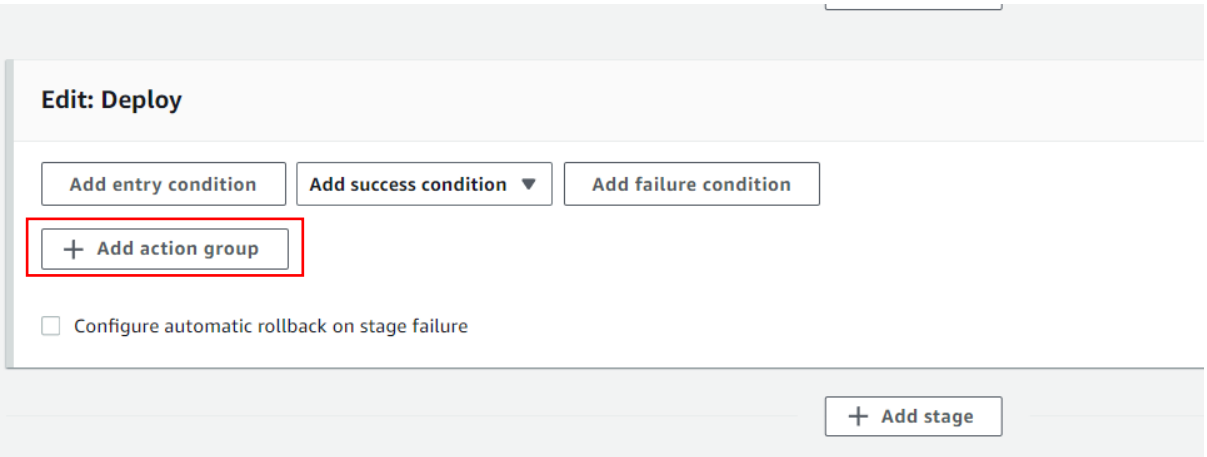
1. Now navigate to code pipeline open your pipeline and click on Edit. Here we are going to remove the S3 deploy action from the pipeline.



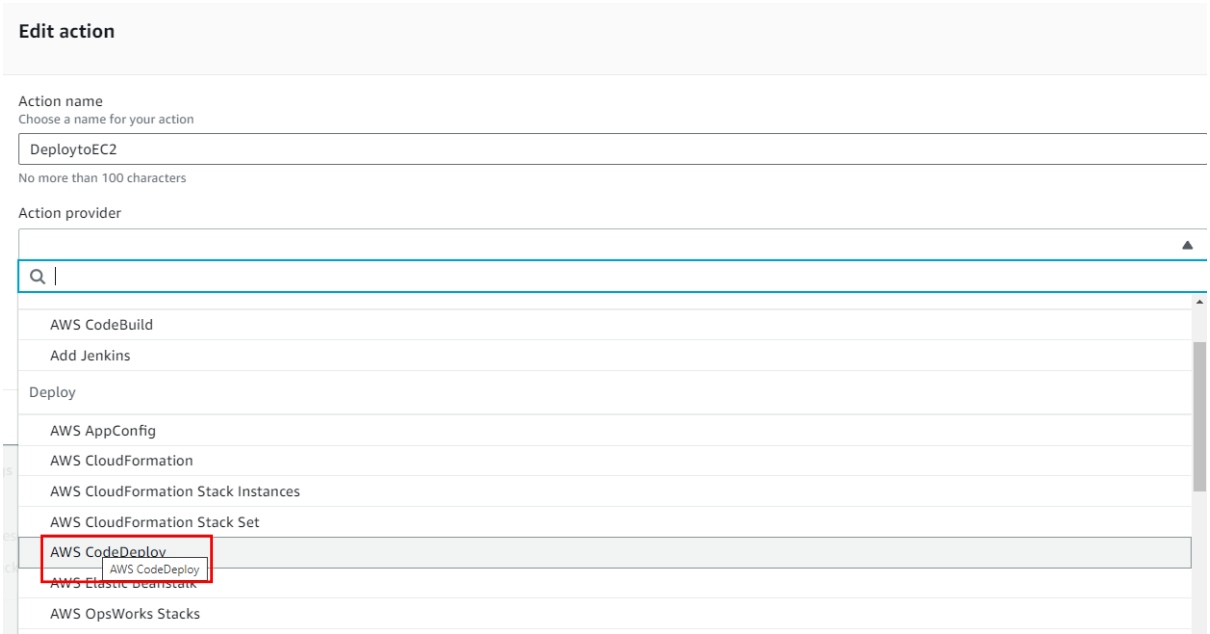

2. Now scroll down to the bottom, click on edit deploy stage and remove the current deploy action.



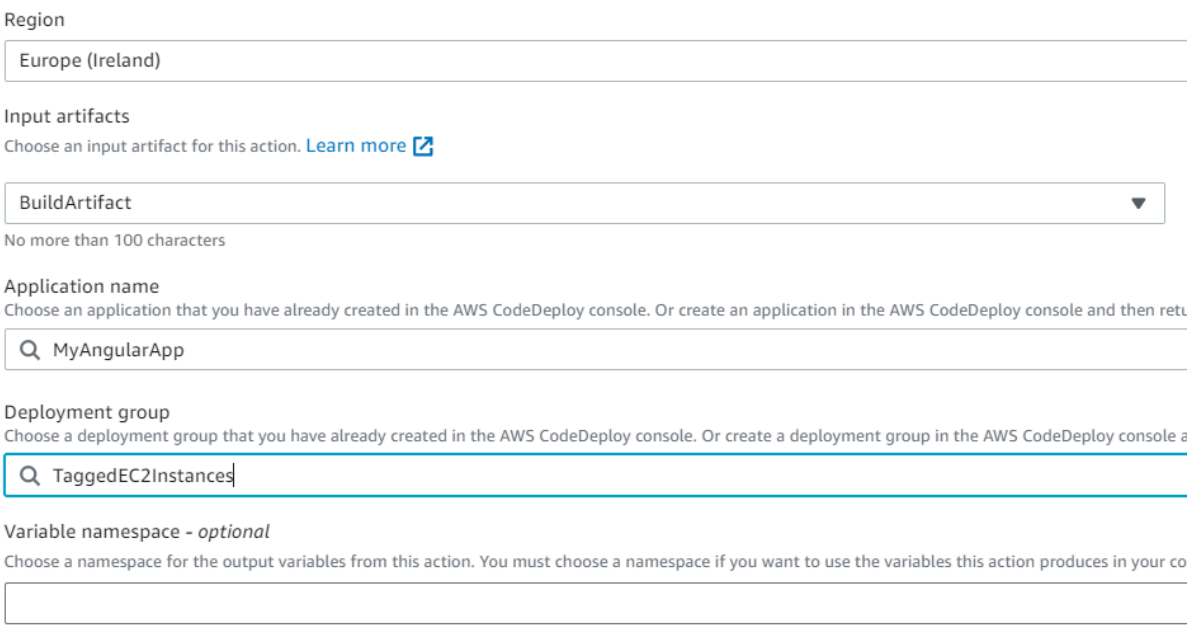

3. Below you can see that we have removed the deploy action from our deploy stage of pipeline. Now we need to click on add action group.



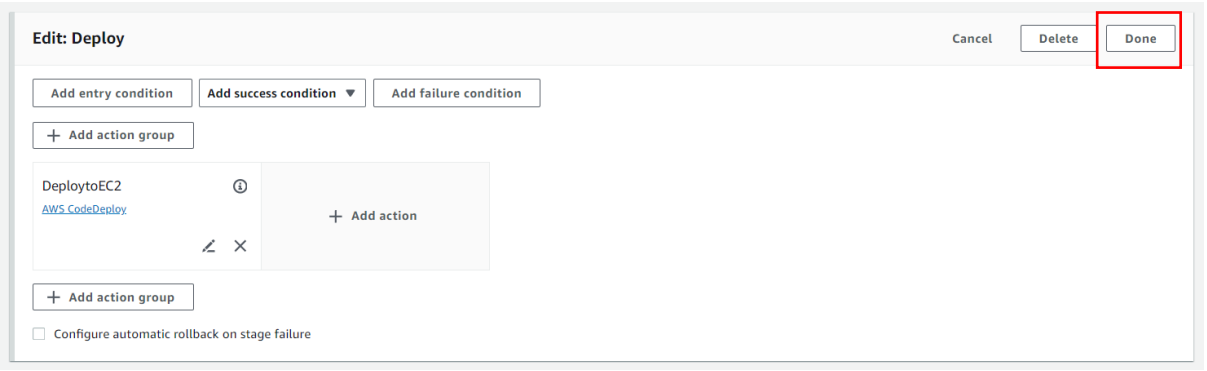

4. Then we need give our action a name and choose the action provider as AWS Code Deploy from the deploy section as you can see below.



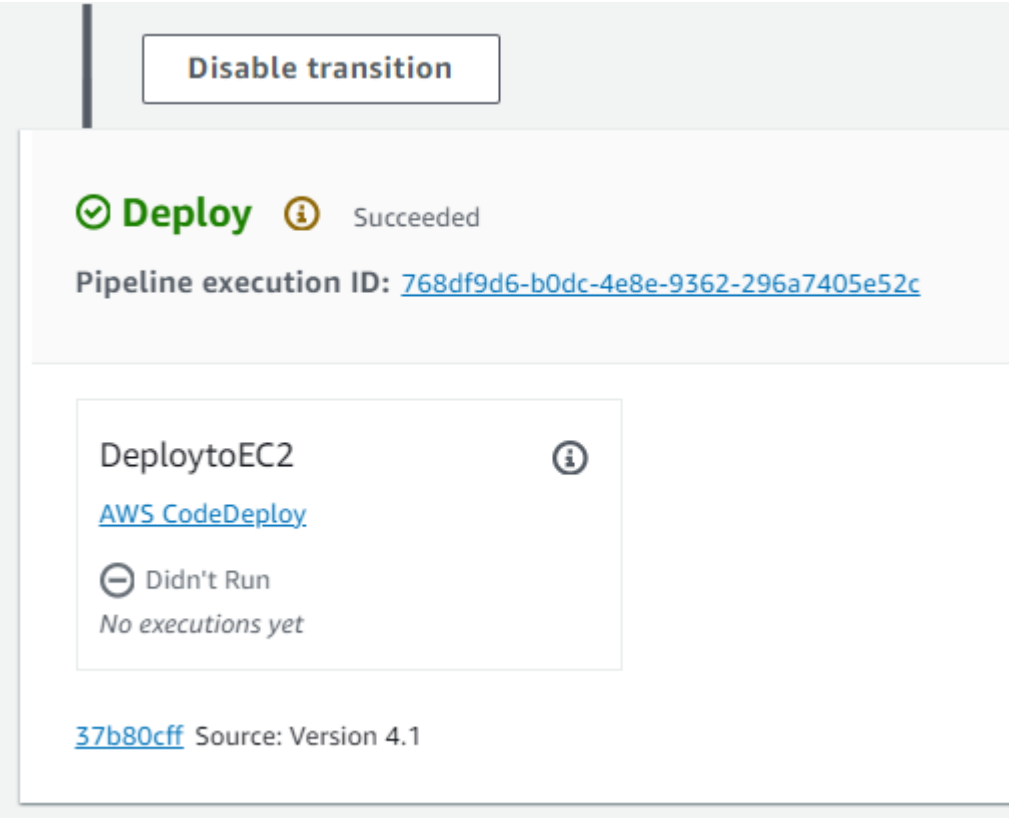

5. After that choose the same region, you were working on and then in the input artifacts choose the build artifact. Now you need to choose the application name and then the deployment group, click on done.




6. Below you can see that our new deploy action has been added now we need to click on done again. Then scroll up to the top of the page and click on save to save the changes in your pipeline.



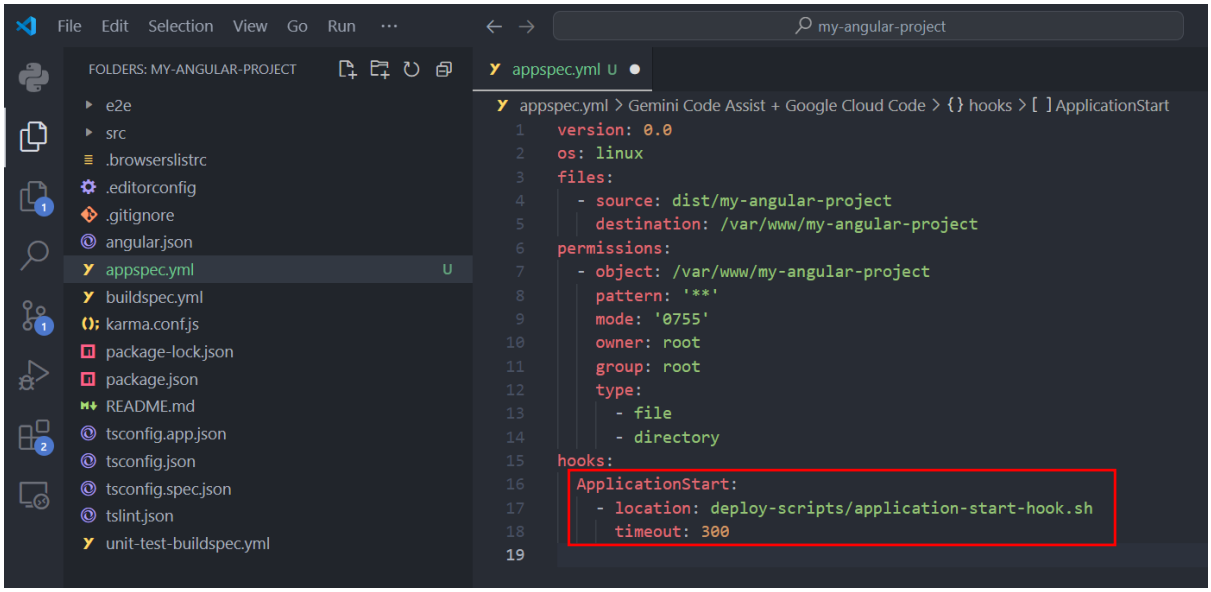

7. Here you can see that our deployment action has been changed.



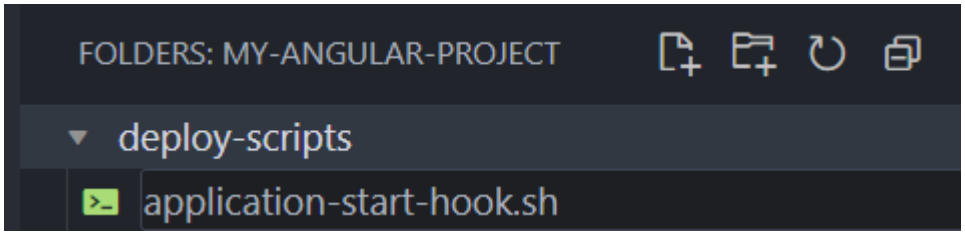

## 😁 Step 5: Creating an Appspec file for deployments to EC2

1. Now open the Angular Project in VS Code then in the root folder you need to create the appspec.yml file and paste the code mentioned below. Please check the indentation of the code after pasting it.

```
version: 0.0
os: linux
files:
  - source: dist/my-angular-project
    destination: /var/www/my-angular-project
permissions:
  - object: /var/www/my-angular-project
    pattern: '**'
    mode: '0755'
    owner: root
    group: root
    type:
      - file
      - directory
hooks:
  ApplicationStart:
    - location: deploy-scripts/application-start-hook.sh
      timeout: 300
```



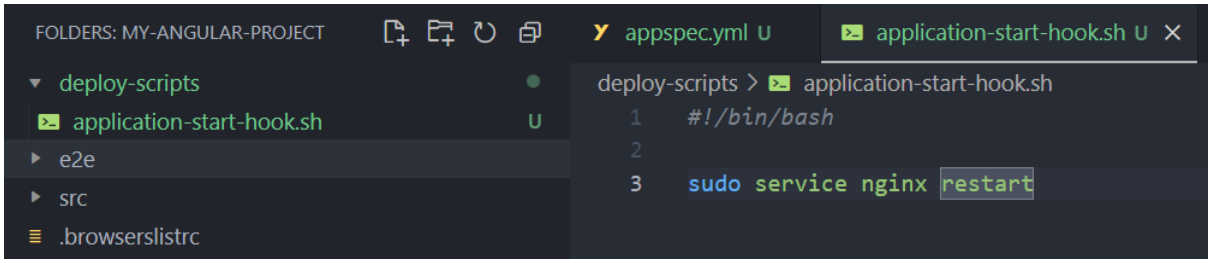

2. Now from the above snapshot you can see that in the application we have given a location here, so we need to specify also where this location comes from.
3. So, for that first you are going to create a folder named deploy-scripts then in this folder you need to create a file with the same name as shown below.




4. Then in the application-start-hook.sh file you need to write the same code as shown below.

**#!/bin/bash**

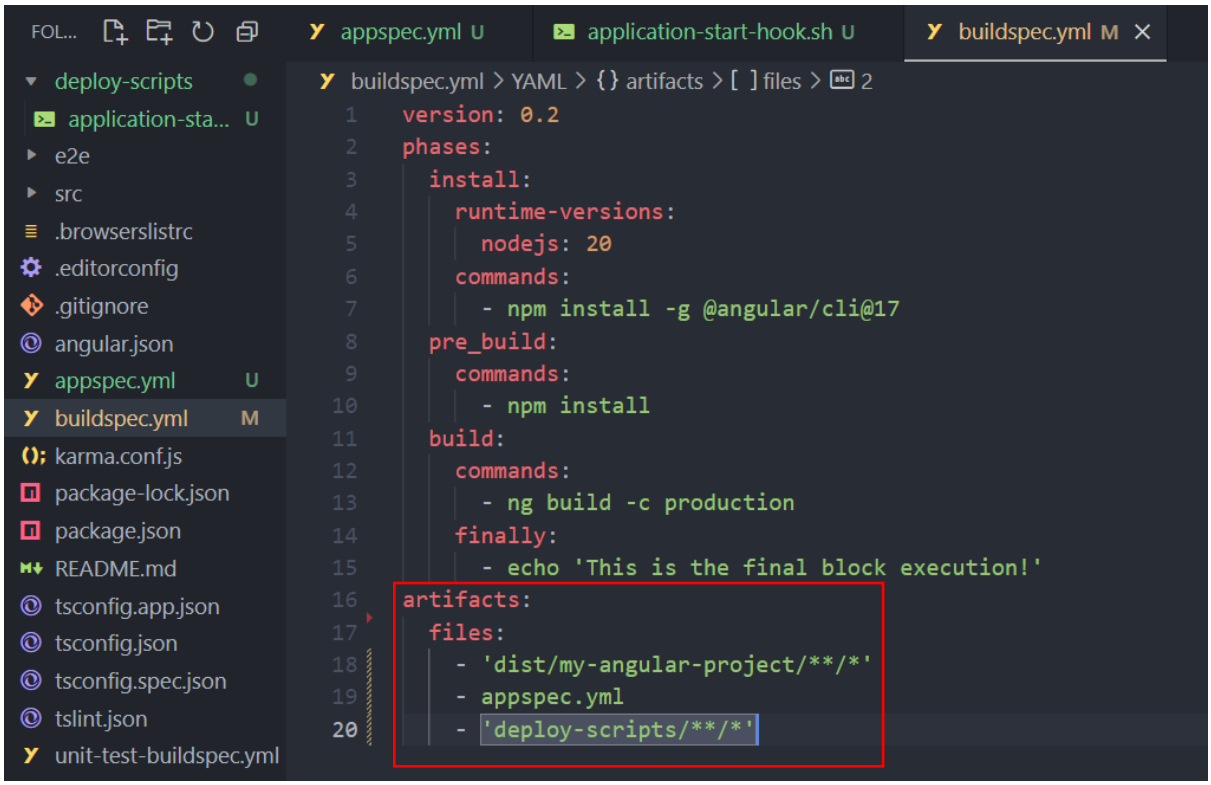
**sudo service nginx restart**




5. After that you need to open your buildspec.yml file. In this file you need to change the highlighted section. Copy the artifacts section from down below and paste in your buildspec file, just remember to check the indentation. Now save everything.

**artifacts:**
**files:**
**- 'dist/my-angular-project/**/*'**

**- appspec.yml**
**- 'deploy-scripts/**/*'**



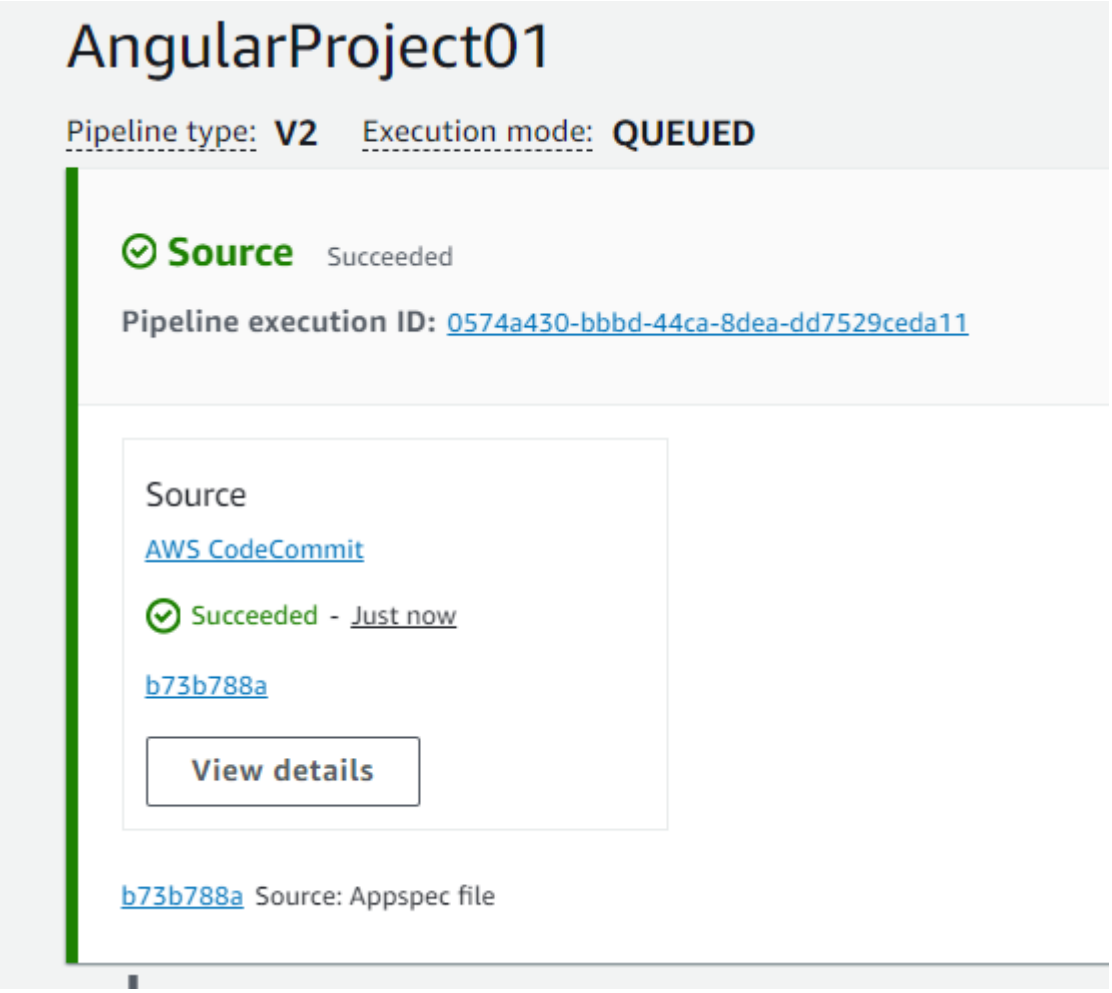

6. After making changes and saving everything we need to run some commands and push our code to the repository.

**git status**
**git add .**
**git commit -m “Appspec file”**
**git push origin master**

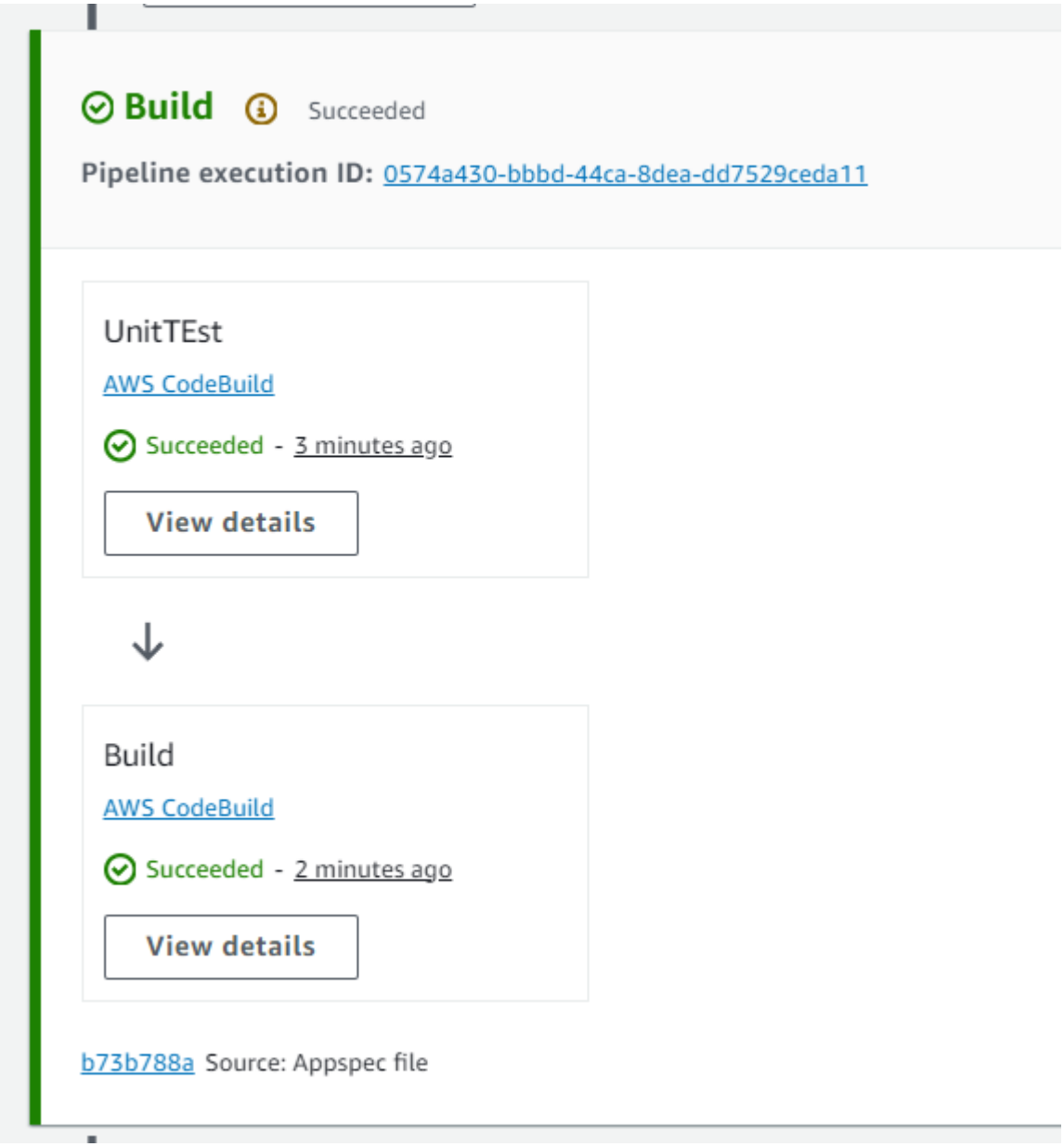
```
PS C:\Users\PULKIT\Downloads\my-angular-project> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   buildspec.yml

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        appspec.yml
        deploy-scripts/

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\PULKIT\Downloads\my-angular-project> git add .
warning: in the working copy of 'deploy-scripts/application-start-hook.sh', LF will be replaced by CRLF the next time Git tou
ches it
PS C:\Users\PULKIT\Downloads\my-angular-project> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   appspec.yml
        modified:   buildspec.yml
        new file:   deploy-scripts/application-start-hook.sh

PS C:\Users\PULKIT\Downloads\my-angular-project>
```

```
PS C:\Users\PULKIT\Downloads\my-angular-project> git commit -m "Appspec file"
[master b73b788] Appspec file
 3 files changed, 25 insertions(+), 2 deletions(-)
 create mode 100644 appspec.yml
 create mode 100644 deploy-scripts/application-start-hook.sh
PS C:\Users\PULKIT\Downloads\my-angular-project> git push origin master
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 778 bytes | 778.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DemoAngularRepo
   37b80cf..b73b788  master -> master
PS C:\Users\PULKIT\Downloads\my-angular-project>
```

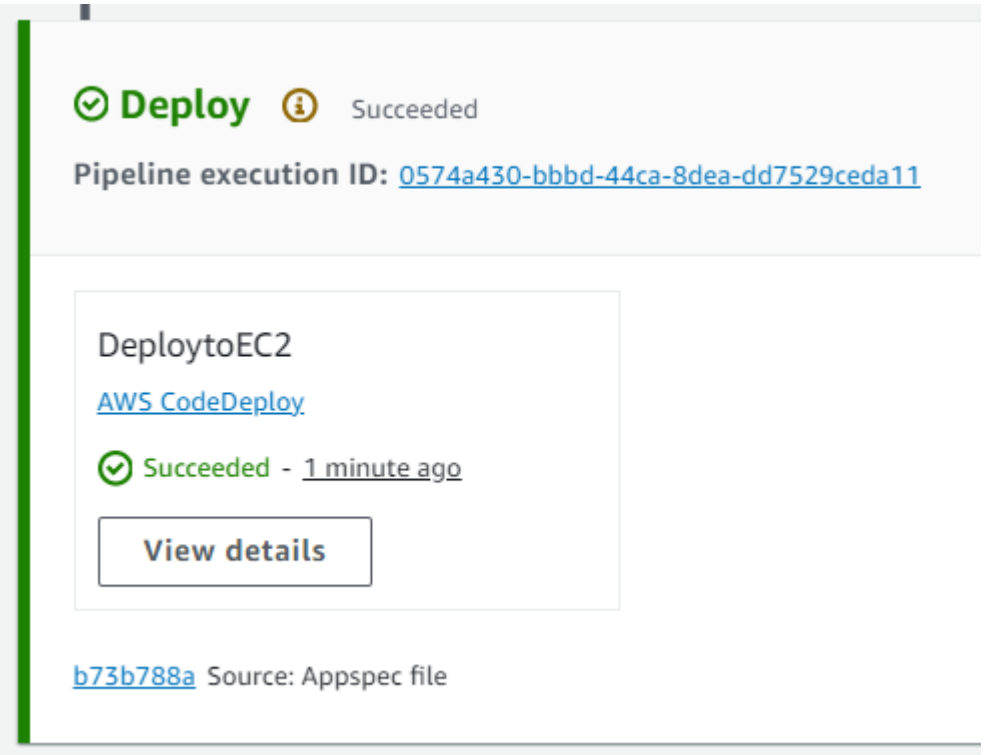
7. Once everything is pushed to the repository go to your pipeline here you will see that your pipeline has been started.




8. Below you can see that our build stage also gets executed.




9. Also, our deploy stage is completed. Now we need to go back to EC2 and




10. Here you can see that our website is working successfully.