

😊 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.

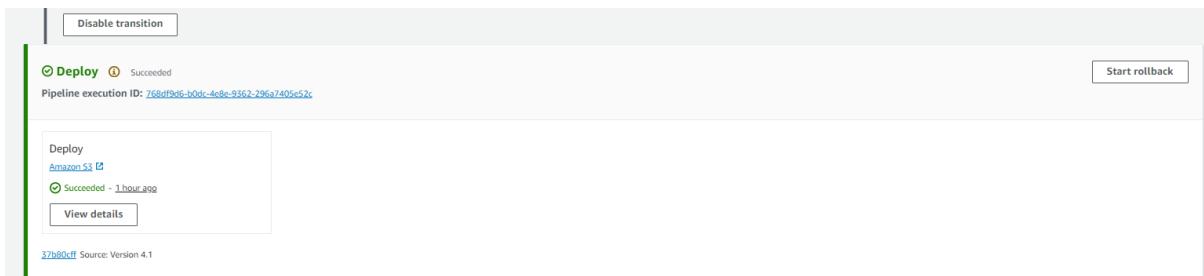
😊 To begin with the Lab:

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.

The screenshot shows the AWS CodePipeline console with two stages visible:

- Source Stage:** Pipeline type: V2, Execution mode: QUEUED. Step: AWS CodeCommit, Status: Succeeded, Last updated: 37h80cff. A "View details" button is present.
- Build Stage:** Pipeline execution ID: 768df9d6-b0dc-4e8e-9362-296a7405e52c. Step: AWS CodeBuild, Status: Succeeded, Last updated: 37h80cff. A "View details" button is present.

On the right side of each stage, there are three green checkmarks indicating successful execution. A "Start rollback" button is located at the top right of the Build stage.



😊 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.

Step 1 Select trusted entity

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.

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

Service or use case
EC2

Choose a use case for the specified service.
Use case

- EC2 Allows EC2 instances to call AWS services on your behalf.
- EC2 Role for AWS Systems Manager Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.

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

Add permissions

Permissions policies (1/1353)

Choose one or more policies to attach to your new role.

Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonEC2RoleforAWSCodeDeploy	AWS managed	Provides EC2 access to S3 bucket to do...
<input type="checkbox"/> AmazonEC2RoleforAWSCodeDeployLimited	AWS managed	Provides EC2 limited access to S3 buck...
<input type="checkbox"/> AWSCodeDeployDeployerAccess	AWS managed	Provides access to register and deploy ...

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

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
WebServerRole
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.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=., @-/[\{\}!\$%^*();":`

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

IAM > Roles > WebServerRole

WebServerRole [Info](#) [Delete](#)

Allows EC2 instances to call AWS services on your behalf.

Summary		Edit
Creation date	ARN	Instance profile ARN
August 08, 2024, 14:09 (UTC+05:30)	arn:aws:iam::463646775279:role/WebServerRole	arn:aws:iam::463646775279:instance-profile/WebServerRole
Last activity	Maximum session duration	
-	1 hour	

[Permissions](#) [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)

Permissions policies (1) [Info](#) [C](#) [Simulate](#) [Remove](#) [Add permissions](#)

You can attach up to 10 managed policies.

Policy name		Type	Attached entities
<input type="checkbox"/>	AmazonEC2RoleforAWSCodeDeploy	AWS managed	1

😊 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.

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

e.g. My Web Server

Add additional tags

- 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.

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

▼ Name and tags Info

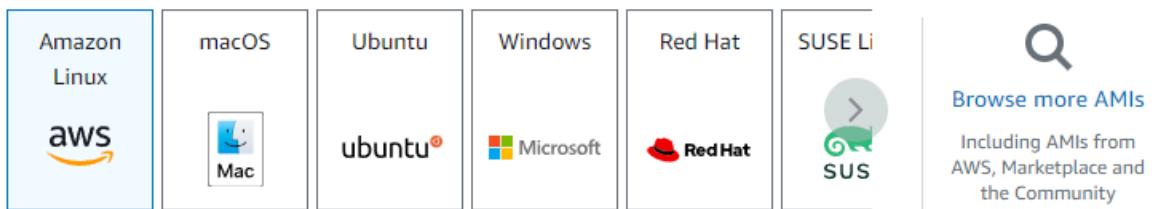
Key | Info Value | Info Resource types | Info

Application MyAngularProjec

You can add up to 49 more tags.

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

Quick Start



Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type ami-068d1303a1458fb15 (64-bit (x86)) / ami-07a452ee4c56924f9 (64-bit (Arm)) Virtualization: hvm ENA enabled: true Root device type: ebs	Free tier eligible
---	--------------------

Description

Amazon Linux 2 comes with five years support. It provides Linux kernel 5.10 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.

Architecture

64-bit (x86) ▾

AMI ID

ami-068d1303a1458fb15

Verified provider

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.

The screenshot shows two steps of an AWS instance creation wizard:

- Instance type:** The user has selected the "t2.micro" instance type. A tooltip provides detailed information about the instance: Family: t2, 1 vCPU, 1 GiB Memory, Current generation: true, On-Demand RHEL base pricing: 0.027 USD per Hour, On-Demand Linux base pricing: 0.0126 USD per Hour, On-Demand SUSE base pricing: 0.0126 USD per Hour, and On-Demand Windows base pricing: 0.0172 USD per Hour. There are buttons for "Free tier eligible" and "All generations", and a link to "Compare instance types". A note at the bottom states: "Additional costs apply for AMIs with pre-installed software".
- Key pair (login):** A note says: "You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance." A dropdown menu for "Key pair name" offers "Proceed without a key pair (Not recommended)" and "Default value". A "Create new key pair" button is also present.

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.

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0) Remove

Type Info ssh	Protocol Info TCP	Port range Info 22
Source type Info Anywhere	Source Info <input type="text"/> Add CIDR, prefix list or security 0.0.0.0/0 X	Description - optional Info e.g. SSH for admin desktop

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0) Remove

Type Info HTTP	Protocol Info TCP	Port range Info 80
Source type Info Anywhere	Source Info <input type="text"/> Add CIDR, prefix list or security 0.0.0.0/0 X	Description - optional Info e.g. SSH for admin desktop

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.

▼ Advanced details [Info](#)

Domain join directory Info Select ▼	Create new directory 
IAM instance profile Info WebServerRole arn:aws:iam::463646775279:instance-profile/WebServerRole ▼	Create new IAM profile 
Hostname type Info IP name ▼	

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.



Instances (1/1) Info									
<input type="text"/> Find Instance by attribute or tag (case-sensitive)		<input type="button"/> Connect		Instance state ▼		Actions ▼		<input type="button"/> Launch instances ▼	
<input checked="" type="checkbox"/>	Name ▼	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm status	Availability Zone ▼	Public IPv4 DNS ▼	Public IPv4
<input checked="" type="checkbox"/>	i-0322de312a501d0a6	Running View details Logs	t2.micro	2/2 checks passed View alarms + Create alarm	eu-west-1a	ec2-18-201-191-79.eu...	18.201.191...		

8. Choose EC2 instance connect click on connect.

- [EC2 Instance Connect](#)
- [Session Manager](#)
- [SSH client](#)
- [EC2 serial console](#)

⚠ **Port 22 (SSH) is open to all IPv4 addresses**

Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: **18.202.216.48/29**. [Learn more](#).

Instance ID
i-0322de312a501d0a6

Connection Type
 Connect using EC2 Instance Connect
 Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

Connect using EC2 Instance Connect Endpoint
 Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address
18.201.191.79

Username
 Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.
ec2-user

ℹ **Note:** In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Cancel
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.

```

Amazon Linux 2
AL2 End of Life is 2025-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-172-31-24-187 ~]$ sudo yum update -y

```

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>

```

[ec2-user@ip-172-31-24-187 ~]$ wget https://aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com/latest/install
--2024-08-08 09:07:08-- https://aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com/latest/install
Resolving aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com (aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com) ... 3.5.67.182, 3.5.69.16, 3.5.70.38, ...
Connecting to aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com (aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com)|3.5.67.182|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19045 (19K) []
Saving to: 'install'

100%[=====] 19,045      --.-K/s   in 0s

2024-08-08 09:07:08 (76.0 MB/s) - `install' saved [19045/19045]

```

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.

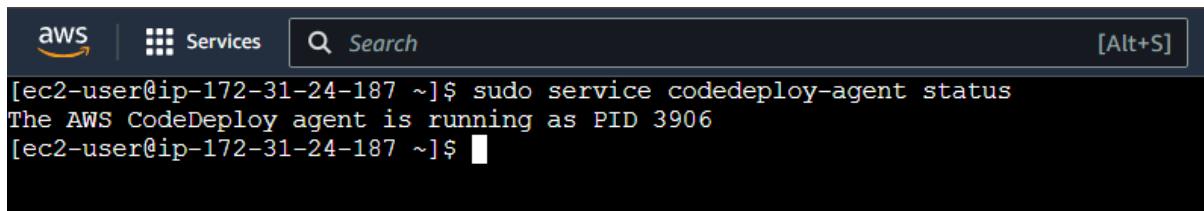
```
[ec2-user@ip-172-31-24-187 ~]$ ls -la
total 32
drwx----- 3 ec2-user ec2-user 89 Aug 8 09:07 .
drwxr-xr-x 3 root root 22 Aug 8 08:53 ..
-rw-r--r-- 1 ec2-user ec2-user 18 Jul 15 2020 .bash_logout
-rw-r--r-- 1 ec2-user ec2-user 193 Jul 15 2020 .bash_profile
-rw-r--r-- 1 ec2-user ec2-user 231 Jul 15 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 ~]$
```

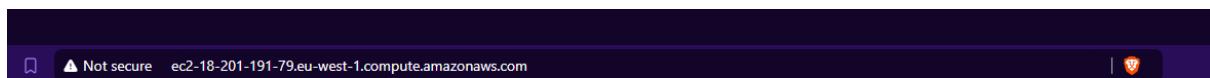
18. As you see, the Nginx service is disabled. So, we must also ensure that Nginx starts on EC2 instance reboots. You run the 'chkconfig' command with 'sudo' for this, provide the 'nginx' service, and then the 'on' parameter to enable it.

sudo chkconfig nginx on

```
[ec2-user@ip-172-31-24-187 ~]$ sudo chkconfig nginx on
[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.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

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;

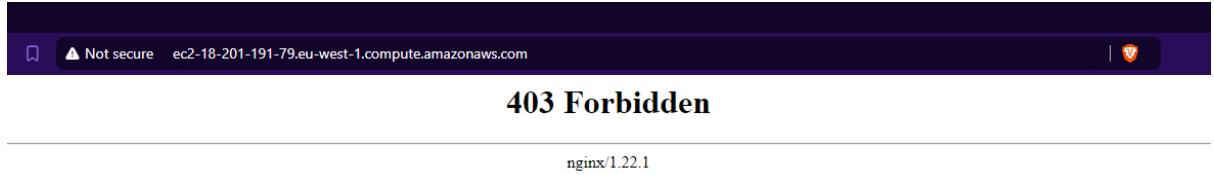
server {
    listen      80;
    listen      [::]:80;
    server_name ;
    root      /var/www/my-angular-project;
```

24. Now we need to restart our Nginx service.

```
sudo service nginx restart
```

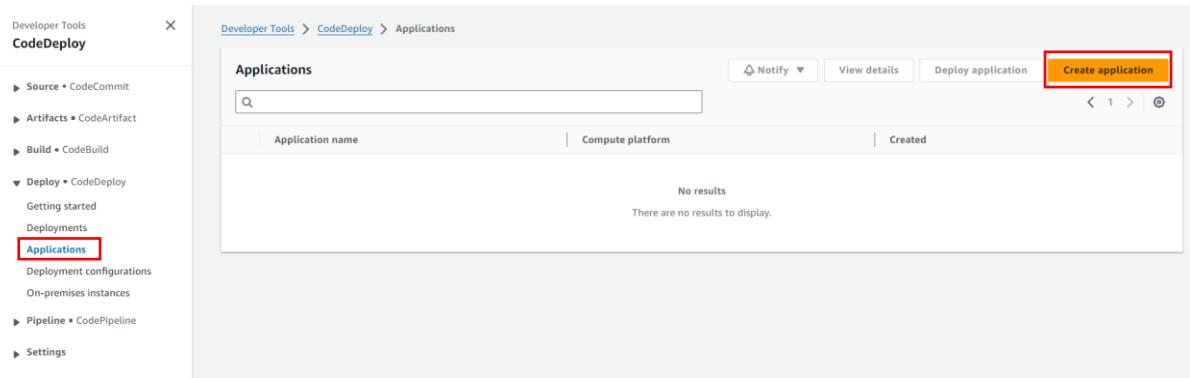
25. After restarting it go back to the website tab and refresh the page you will see that it responds with the '403 Forbidden' error. Why is that? Well, don't worry, it is expected

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.

Create application

Application configuration

Application name

Enter an application name

100 character limit

Compute platform

Choose a compute platform

Tags

Cancel

Create application

- 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.
- Here in the use case you need to search for Code Deploy and choose it as shown below and click on next.

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.

Service or use case

Choose a use case for the specified service.

Use case

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 SNS topics, and update ECS services on your behalf.

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

Add permissions [Info](#)

Permissions policies (1) [Info](#)
The type of role that you selected requires the following policy.

Policy name Edit	Type
<input checked="" type="checkbox"/> AWSCodeDeployRole	AWS managed

► Set permissions boundary - *optional*

Cancel [Previous](#) [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.

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
 CodeDeployEC2ServiceRole

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

Description
Add a short explanation for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=_, @_/\[\]!#\$%^&*();~`

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

[Permissions](#) [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)

Permissions policies (1) [Info](#)
You can attach up to 10 managed policies.

Policy name Edit	Type	Attached entities
<input checked="" type="checkbox"/> AWSCodeDeployRole	AWS managed	15

Filter by Type
 Search

[C](#) [Simulate](#) [Remove](#) [Add permissions ▲](#)
[Attach policies](#) [Create 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"],
            "Resource": "*"
        }
    ]
}
```

```

        "Resource": "*"
    }
}

```

Policy editor

```

1 ▼ {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Sid": "Statement1",
6             "Effect": "Allow",
7             "Action": ["ec2:RunInstances", "ec2:CreateTags", "iam:PassRole"],
8             "Resource": "*"
9         }
10    ]
11 }

```

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

Review and create [Info](#)
Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and '+,-,@,_' characters.

Permissions defined in this policy [Info](#)
Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Service	Access level	Resource	Request condition
EC2	Limited: Write, Tagging	All resources	None
IAM	Limited: Write	All resources	None

Permissions **Trust relationships** **Tags** **Access Advisor** **Revoke sessions**

Permissions policies (2) [Info](#)
You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AWSCodeDeployRole	AWS managed	15
CodeDeployBlueGreenPolicy	Customer inline	0

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

Developer Tools > CodeDeploy > Applications > MyAngularApp

MyAngularApp

Application details

Name: MyAngularApp Compute platform: EC2/On-premises

Deployments Deployment groups Revisions

Deployment groups

No deployment groups

Before you can deploy your application using CodeDeploy, you must create a deployment group.

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.

Application

Application: MyAngularApp
Compute type: EC2/On-premises

Deployment group name

Enter a deployment group name
TaggedEC2Instances
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.
arn:aws:iam::463646775279:role/CodeDeployEC2ServiceRole

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

Deployment type

Choose how to deploy your application

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

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.

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.

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

1 unique matched instance. [Click here for details](#) 

You can add up to three groups of tags for EC2 instances to this deployment group.

One tag group: Any instance identified by the tag group will be deployed to.

Multiple tag groups: Only instances identified by all the tag groups will be deployed to.

Tag group 1

Key

Application

Value - optional

MyAngularProject

X

[Remove tag](#)

[Add tag](#)

[+ Add tag group](#)

On-premises instances

Matching instances

1 unique matched instance. [Click here for details](#) 

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

Agent configuration with AWS Systems Manager [Info](#)



We recommend configuring your CodeDeploy Agent install and updates with AWS Systems Manager. AWS Systems Manager provides more control over CodeDeploy Agent version updates and rollbacks than installing using other methods. [Learn more](#)

Install AWS CodeDeploy Agent

- Never
- Only once
- Now and schedule updates

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

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

► Advanced - optional

[Cancel](#)

[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.

Developer Tools > CodePipeline > Pipelines > AngularProject01

AngularProject01

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded

Pipeline execution ID: [768df9d6-b0dc-4e8e-9362-296a7405e52c](#)

Notify Edit Stop execution Clone pipeline Release change

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

Edit: Deploy

Conditions Entry: Not configured Success: Not configured Failure: Not configured View details

Deploy Amazon S3 ⓘ

Configure automatic rollback on stage failure

+ Add stage

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.

Edit: Deploy

Add entry condition Add success condition Add failure condition

+ Add action group

Configure automatic rollback on stage failure

+ Add stage

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.

Edit action

Action name
Choose a name for your action
DeploytoEC2
No more than 100 characters

Action provider

Q |

- AWS AppBuild
- Add Jenkins
- Deploy
- AWS AppConfig
- AWS CloudFormation
- AWS CloudFormation Stack Instances
- AWS CloudFormation Stack Set
- AWS CodeDeploy**
- AWS CodeDeploy
- AWS Elastic Load Balancing
- AWS OpsWorks Stacks

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.

Region

Input artifacts

Choose an input artifact for this action. [Learn more](#)

▼

No more than 100 characters

Application name

Choose an application that you have already created in the AWS CodeDeploy console. Or create an application in the AWS CodeDeploy console and then return to this step.

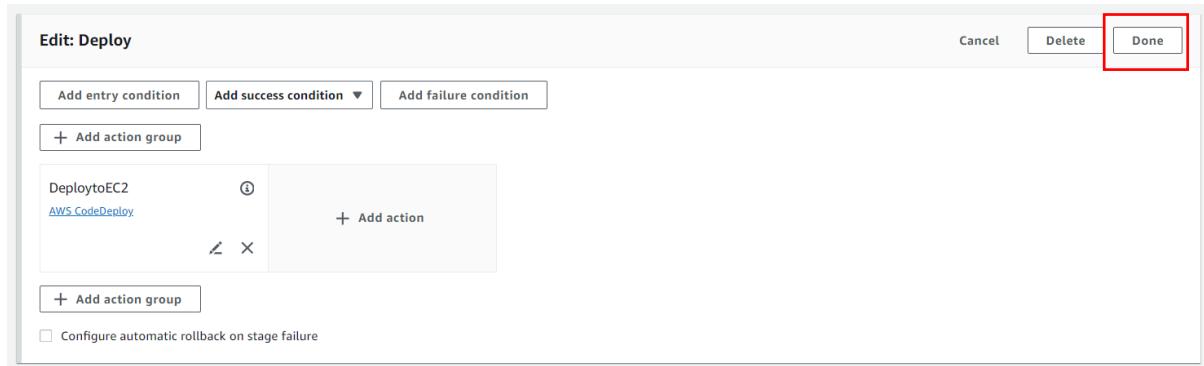
Deployment group

Choose a deployment group that you have already created in the AWS CodeDeploy console. Or create a deployment group in the AWS CodeDeploy console and then return to this step.

Variable namespace - *optional*

Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your code.

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.

Disable transition

Deploy ⓘ Succeeded

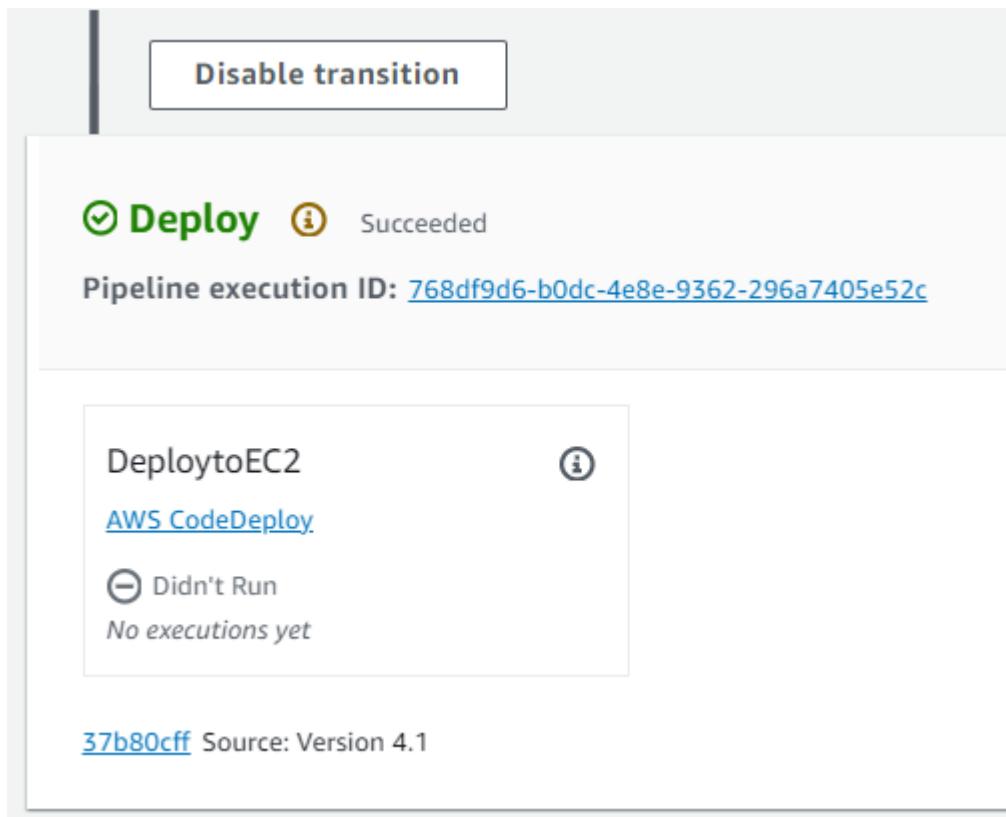
Pipeline execution ID: [768df9d6-b0dc-4e8e-9362-296a7405e52c](#)

DeploytoEC2 ⓘ

[AWS CodeDeploy](#)

⊖ Didn't Run
No executions yet

[37b80cff](#) Source: Version 4.1



😊 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.

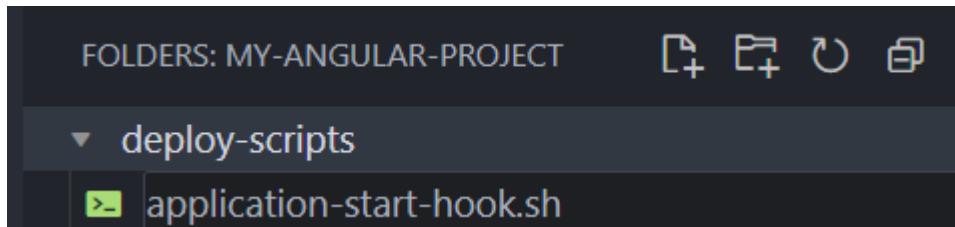
```
version: 0.0
os: linux
files:
  - source: dist/my-angular-project
    destination: /var/www/my-angular-project
permissions:
  - object: /var/www/my-angular-project
    pattern: '**'
    mode: '0755'
    owner: root
    group: root
    type:
      - file
      - directory
hooks:
  ApplicationStart: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

```

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
```

```

#!/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/**/*'

```

FOL...  Deploy Scripts  application-start-hook.sh  buildspec.yml
      deploy-scripts  application-start-hook.sh  buildspec.yml
      e2e
      src
      .browserslistrc
      .editorconfig
      .gitignore
      angular.json
      appspec.yml  buildspec.yml
      karma.conf.js
      package-lock.json
      package.json
      README.md
      tsconfig.app.json
      tsconfig.json
      tsconfig.spec.json
      tslint.json
      unit-test-buildspec.yml

Y buildspec.yml > YAML > {} artifacts > [ ] files > 2
  1   version: 0.2
  2   phases:
  3     install:
  4       runtime-versions:
  5         nodejs: 20
  6       commands:
  7         - npm install -g @angular/cli@17
  8     pre_build:
  9       commands:
 10         - npm install
 11     build:
 12       commands:
 13         - ng build -c production
 14     finally:
 15       - echo 'This is the final block execution!'
 16   artifacts:
 17     files:
 18       - 'dist/my-angular-project/**/*'
 19       - appspec.yml
 20       - '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 touches 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> █
```

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

AngularProject01

Pipeline type: V2 Execution mode: QUEUED

⌚ Source Succeeded

Pipeline execution ID: [0574a430-bbbd-44ca-8dea-dd7529ceda11](#)

Source
[AWS CodeCommit](#)

⌚ Succeeded - Just now

[b73b788a](#)

[View details](#)

[b73b788a](#) Source: Appspec file

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

Build Succeeded

Pipeline execution ID: [0574a430-bbbd-44ca-8dea-dd7529ced11](#)

UnitTEst

[AWS CodeBuild](#)

Succeeded - 3 minutes ago

[View details](#)

↓

Build

[AWS CodeBuild](#)

Succeeded - 2 minutes ago

[View details](#)

[b73b788a](#) Source: Appspec file

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

Deploy Succeeded

Pipeline execution ID: [0574a430-bbbd-44ca-8dea-dd7529ced11](#)

DeploytoEC2

[AWS CodeDeploy](#)

Succeeded - 1 minute ago

[View details](#)

[b73b788a](#) Source: Appspec file

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

