AWS CodeDeploy

CodeDeploy is a deployment service from AWS which can automate application deployments to Amazon EC2 instances, on-premises instances or Lambda functions. This does a onetime deployment, for scheduling of deployment you may have to use AWS CodePipeline also.

Application: A CodeDeploy application can be defined from AWS CodeDeploy web console.

Revision: Represents the code need to be deployed on EC2 instance.

Appspec file**:** This contains the instruction to CodeDeploy, like copying of files, executing the scripts etc during the code deployment process. It is present in the root directory of unzipped code with name appspec.yml.

Deployment Group: Represent set of machines of Lambda function where code has to be deployed.

Deployment: The process of deployment.

Setup in Brief:

I have used two EC2 instance of AMZ2 Linux. First one is the web server we will be configuring, also called CodeDeploy agent. Second EC2 machine is supposed to use by developer where the codes are programmed. The names of the resources in the experiment are arbitrary and may name the resources your own.

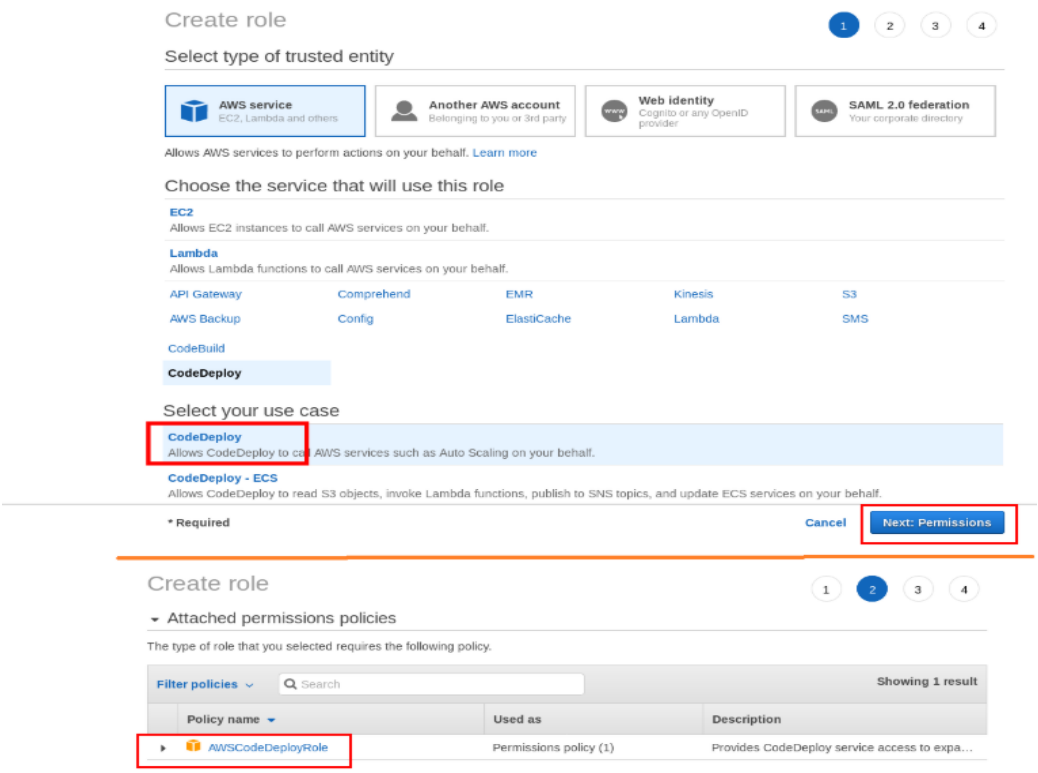
1. Create IAM Roles for EC2-S3-CodeDeploy access.
2. Create IAM user account for developer
3. Install and prepare the CodeDeploy agent on webserver.
4. Create the code from Developer machine
5. Create Codedeploy Application and Push the code to S3 bucket from Developer machine
6. Create Deployment Group to include web server
7. Create Deployment to push the code to the webserver
8. Test the website configuration

Steps in Detail

1- Create IAM Roles for EC2-S3-CodeDeploy access.

a - Create IAM Role for EC2 instance to access S3. Select EC2 as AWS service and assign *AmazonS3FullAccess* permission. Use any arbitrary name for the Role. I have used a name *s3-ec2-full*. This Role must be attached the EC2 instance (webserver) later.

b- Create another IAM Role for CodeDeploy access. Select CodeDeploy as AWS Service and assign *AWSCodeDeployRole* permission like below. I have assigned a name *cdrole*. This role must be used while the CodeDeploy deployment is configured in a later stage.



2-Create IAM user account for developer

a- Use the existing desktop/laptop or Launch a new EC2 instance. This is used by the Developer for the code creation and manual pushing of code to S3 bucket.

b- Create an IAM user and assign programming access. He should be given AmazonS3FullAccess and AWSCodeDeployFullAccess permissions

c - execute aws configure command on developer's machine and install the access/secret keys.

3- Install and prepare the CodeDeploy agent on webserver

a- Launch the EC2 instance. This is used for deploying webserver with CodeDeploy.

b-Create a Tag for the instance. The deployment group member ship for the EC2 instance is decided by this Tag. I have used AppName Tag with value SampleApp.

c- open the port 80 for Security Group since it is a web server.

d- Attach the Role s3-ec2-full to this instance.

e - SSH to the Instance and su to root and execute the command below. This will download the CodeDeply agent software and install. Make sure the you don't change the directory during the process.

```
# yum update
# yum install ruby -y
# yum install wget -y
# wget https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install
# chmod +x install
# ./install auto
# service codedeploy-agent status
```

4 - Create the code from Developer machine

Note: you may copy the contents from this document to create code. scripts should be given execute permissions.

a- SSH to developer machine. I have su to root and a created a directory /root/deploy_dir

b- Make sure that zip file of the of the code and its extracted directory is kept inside the directory /root/deploy_dir. my application name is sampleapp.

c-Let us visit the code now. The output shows a s*ampleapp* directory which is extracted from the code sampleapp.zip

```
[root@ip-172-30-0-178 deploy_dir]# ls
sampleapp sampleapp.zip
```

d-Listing all file and directories in the code

```
[root@ip-172-30-0-178 deploy_dir]# ls -R
.:
sampleapp sampleapp.zip
./sampleapp:
appspec.yml index.html scripts
./sampleapp/scripts:
httpd_install.sh httpd_start.sh httpd_stop.sh
```

The code should contain a file appspec.yml*. The* files: *section says what are files to be copied in which directory of the destination machine. I want to copy index.html to /var/www/html.* BeforeInstall: *section says what action must be done before install application in my case before copying the file I wanted httpd rpm package has to be installed*.

```
[root@ip-172-30-0-178 deploy_dir]# cat sampleapp/appspec.yml
version: 0.0
os: linux
files:
 - source: /index.html
   destination: /var/www/html/
hooks:
 BeforeInstall:
  - location: scripts/httpd_install.sh
    timeout: 300
    runas: root
  - location: scripts/httpd_start.sh
    timeout: 300
    runas: root
 ApplicationStop:
  - location: scripts/httpd_stop.sh
    timeout: 300
    runas: root
```

e- Let us see the contents of script files

```
[root@ip-172-30-0-178 deploy_dir]# cat sampleapp/scripts/httpd_install.sh
#!/bin/bash
yum install -y httpd

[root@ip-172-30-0-178 deploy_dir]# cat sampleapp/scripts/httpd_start.sh
#!/bin/bash
systemctl start httpd
systemctl enable httpd

[root@ip-172-30-0-178 deploy_dir]# cat sampleapp/scripts/httpd_stop.sh
#!/bin/bash
systemctl stop httpd
systemctl disable httpd
```

f-contents of/root/deploy_dir/sampleapp/index.html

```
[root@ip-172–30–0–178 sampleapp]# cat index.html
<html>
<h2> Sample App Version 1 </h2>
</html>
```

5 -Create Application & Push the code to S3 bucket

a- Create S3 bucket for uploading the code, I have named it as gir-sampleapp
b- Change directory to sampleapp developer machine and create a codedeploy application. Execute the command below

```
# aws deploy create-application --application-name sampleapp
```

c- Now upload the code to S3 by the executing the command below. Directory of execution is important.

```
# aws deploy push --application-name sampleapp --s3-location s3://gir-sampleapp/sampleapp.zip
```

d- Now browse the s3 bucket to see that sampleapp.zip is present.

6-Create Deployment Group to include webserver

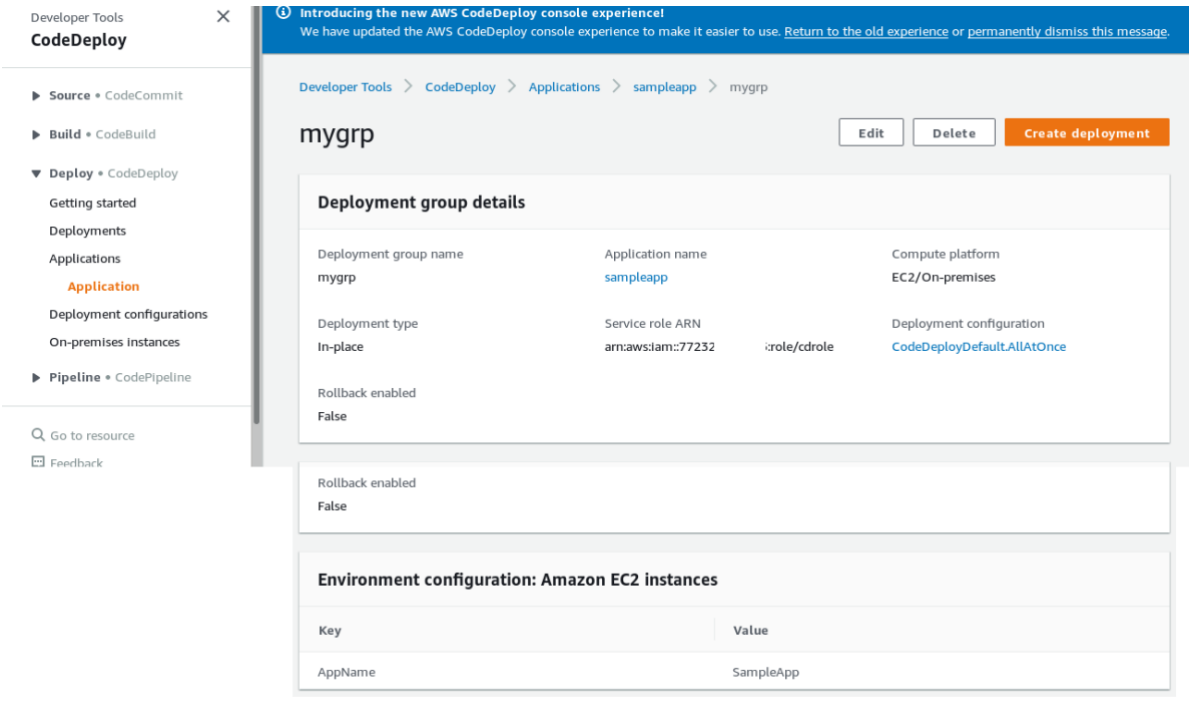
a- Login to Codedeply AWS web console

b- Select sampleapp and click *Create Deployment Group* from *Deployment Groups* tab.

c- Enter the values like below and leave the other parameters default

Enter a deployment group name: mygrp
Choose a service role: cdrole
Deployment type: in-place
Environment configuration: choose Amazon EC2 instances
Key as *AppName* Value as *SampleApp*
Load balancer: uncheck Enable load balancing

Click *Create Deployment Group* button to finish creation of deployment group

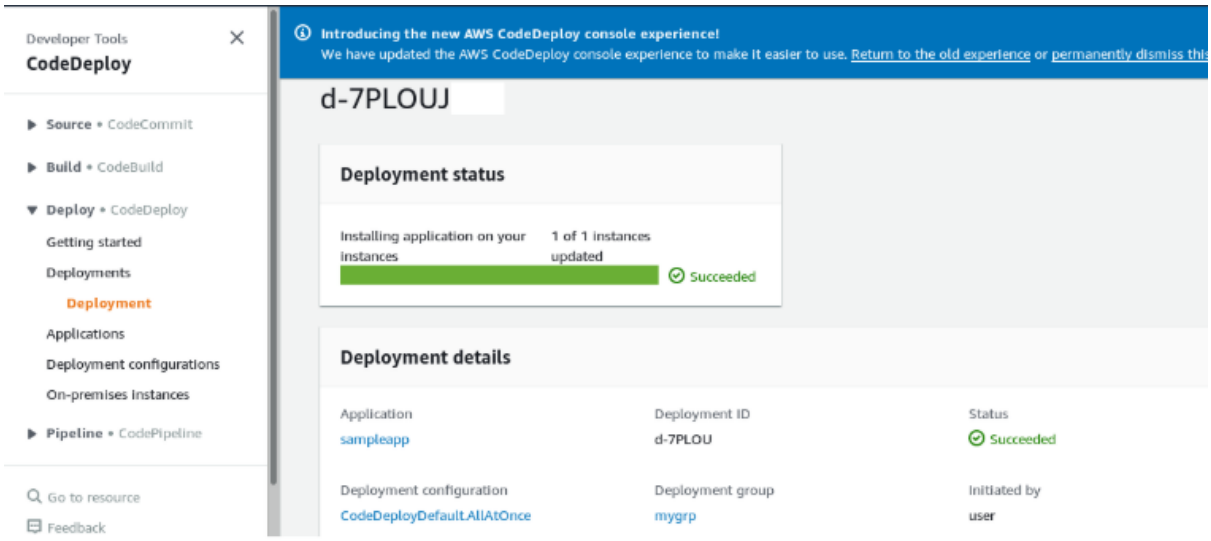


7-Create Deployment which pushes code to the webserver

In the sampleapp click *Create Deployment.* Enter values like below. Other parameter can can be kept default

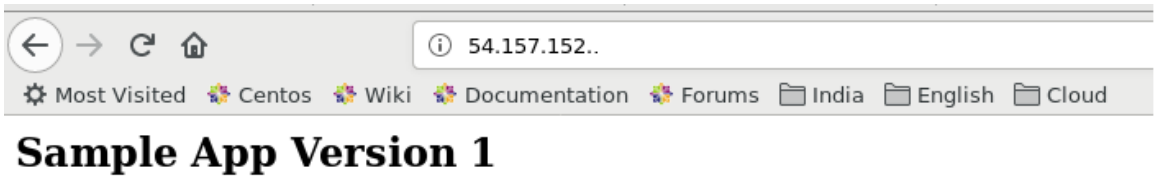
Deployment group : mygrp *Revision type:* My application is stored in Amazon S3 *Revision location* : s3://select_location_from_list

Click *Create Deployment* to finish



8-Testing the Configuration

Now access the public Ip address fo the webserver from the browser and see that it is working



AWS CodePipeline

AWS CodePipeline is a continuous delivery service for software releases. CodePipeline can automate the process of software deployment and releases. Here we will see how CodePipeline can be used to update the webserver with a new release version. In this experiment we make small change in index.html for the second release, rest all files are same for both versions.

We use AWS CodeDeploy along with AWS Codepipeline for this experiment. Initial version of the code in uploaded to the S3 bucket. Whenever new version is released, the only operation required by the developer is to upload the new version of the code to the same S3 bucket. CodePipeline does the rest. It detects a new upload in the S3 bucket with help of CloudWatch and trigger the deployment to the target server (webserver in our case) using CodeDeploy.

Note: Since CodeDeploy is most important part of CodePipeline configuration, follow the earlier process what we saw to implement the same.

Setup in brief:

1. Create IAM Roles for EC2-S3-CodeDeploy access.
2. Create IAM user account for developer
3. Install and prepare the CodeDeploy agent on webserver.
4. Develop Initial version of the code
5. Upload version 1 code to S3 bucket
6. Develop version 2 of the code
7. Create CodeDeploy Application and Deploy Version 1
8. Create a CodePipeline
9. Upload the version 2 of code to S3 bucket and test the setup

Step 1,2,3 was already completed with CodeDeploy, hence move on from the Step 4.

4- Develop Initial Version of the code

Creating the code is slightly different from the previous experiment.

a- SSH to Developer machine.

b- create a directory /root/deploy_dir. Create a subdirectory sampleapp

```
# mkdir /root/deploy_dir
# cd /root/deploy_dir
# mkdir sampleapp
```

c- Under sampleapp create files appspec.yml, index.html and scripts directory. Inside scripts create three scripts httpd_install.sh, httpd_start.sh and httpd_stop.sh.

d- Let us see the contents of the appspec.yml file.

```
[root@ip-172-30-0-178 deploy_dir]# cat sampleapp/appspec.yml
version: 0.0
os: linux
files:
 - source: /index.html
   destination: /var/www/html/
hooks:
 BeforeInstall:
  - location: scripts/httpd_install.sh
    timeout: 300
    runas: root
  - location: scripts/httpd_start.sh
    timeout: 300
    runas: root
 ApplicationStop:
  - location: scripts/httpd_stop.sh
    timeout: 300
```

e - Let us see the contents of all scripts

```
[root@ip-172-30-0-178 deploy_dir]# cat sampleapp/scripts/httpd_install.sh
#!/bin/bash
yum install -y httpd

[root@ip-172-30-0-178 deploy_dir]# cat sampleapp/scripts/httpd_start.sh
#!/bin/bash
systemctl start httpd
systemctl enable httpd

[root@ip-172-30-0-178 deploy_dir]# cat sampleapp/scripts/httpd_stop.sh
#!/bin/bash
systemctl stop httpd
systemctl disable httpd
```

f- Assign execute permission for all script files

```
# chmod 755 /root/deploy_dir/sampleapp/scripts/*
```

g- Create index.html file

```
[root@ip-172–30–0–178 sampleapp]# cat index.html
<html>
<h2> Sample App Version 1 </h2>
</html>
```

h - create zip file of the code. *Directory of execution is very important.*

```
# cd /root/deploy_dir/sampleapp
# zip  -r    ../sampleapp.zip   .
```

5-Upload the version 1 code to S3 bucket

a - create S3 bucket *gir-sampleapp* and enable **Versioning**. copy sampleapp.zip to the S3 bucket.

```
# aws s3 cp sampleapp.zip  s3://gir-sampleapp
```

6 - Develop version 2 of the code

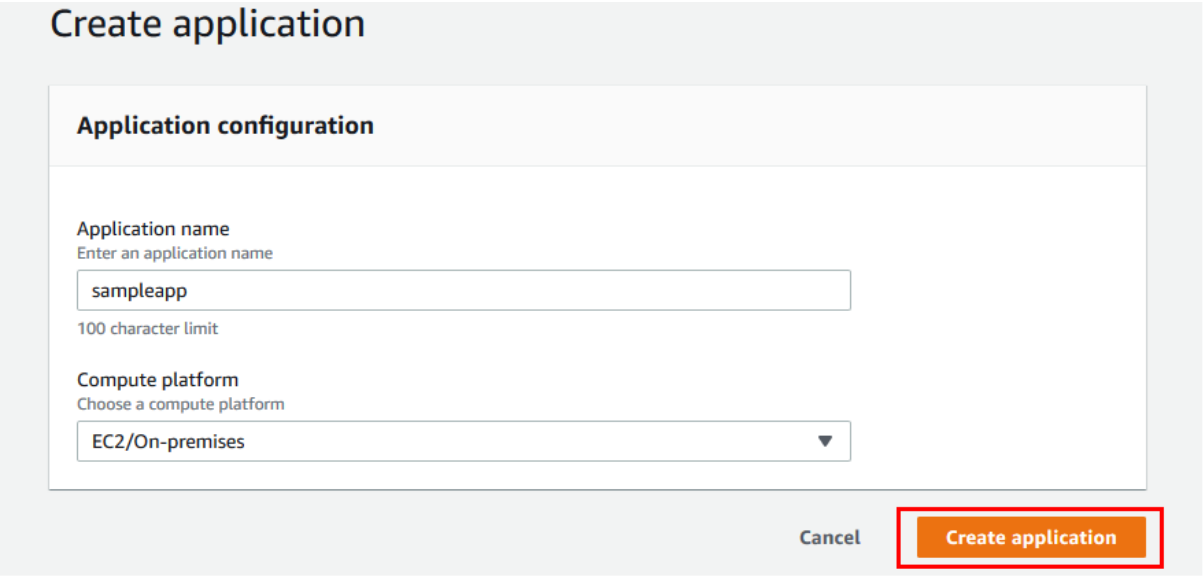
a - Now you have to develop the code for the second release in a different directory /root/deploy_dir2. We will create second version just by modifying the contents of index.html. In the file index.html the text "Version 1" is changed to "Version 2". Rest of the files and codes are same. For that follow the steps below

```
# mkdir /root/deploy_dir2
# cd /root/deploy_dir2
# cp -var /root/deploy_dir/sampleapp  .
# cat /root/deploy_dir2/sampleapp/index.html
<html>
<h2> Sample App Version 2 </h2>
</html>
# cd /root/deploy_dir2/sampleapp
# zip -r ../sampleapp.zip .
```

b- Output of "ls -R" on /root/deploy_dir2 is like below.

```
[root@ip-172-30-0-178 deploy_dir2]# ls -R
.:
sampleapp sampleapp.zip
./sampleapp:
appspec.yml index.html scripts
./sampleapp/scripts:
httpd_install.sh httpd_start.sh httpd_stop.sh
```

7- Create CodeDeploy Application and Deploy code Version 1

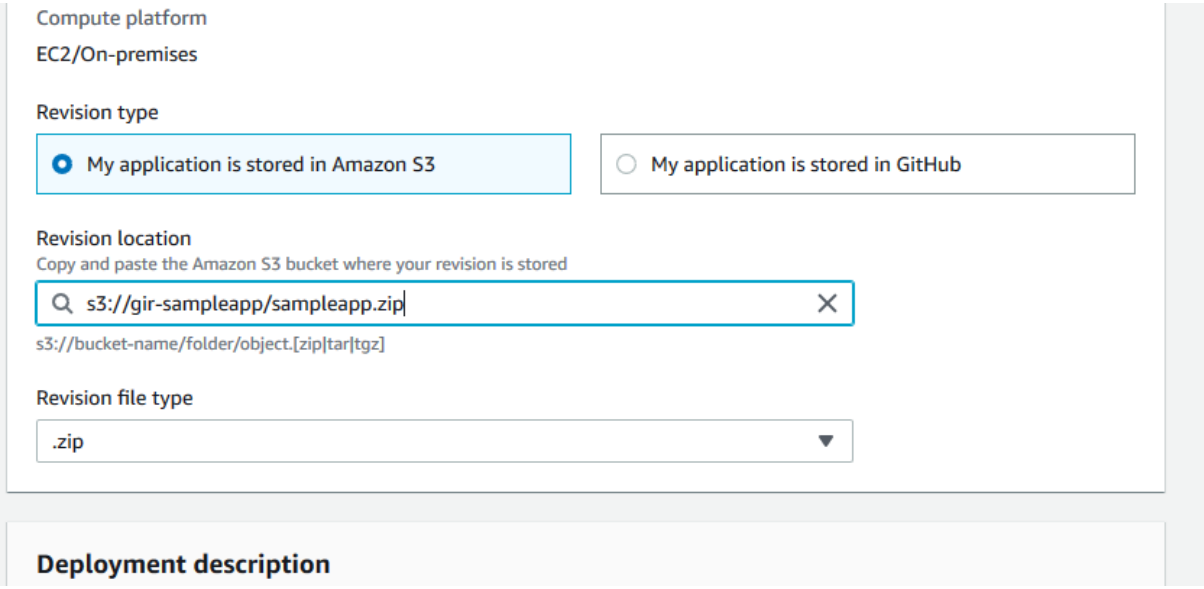


a - Create *Deployment Group* with following options. Leave other as default

Enter a deployment group name: mygrp
Choose a service role: cdrole
Deployment type: in-place
Environment configuration: choose Amazon EC2 instances
Key as *AppName* Value as *SampleApp*
Load balancer: uncheck Enable load balancing

b- Create *Deployment* with following options. Leave other as default.

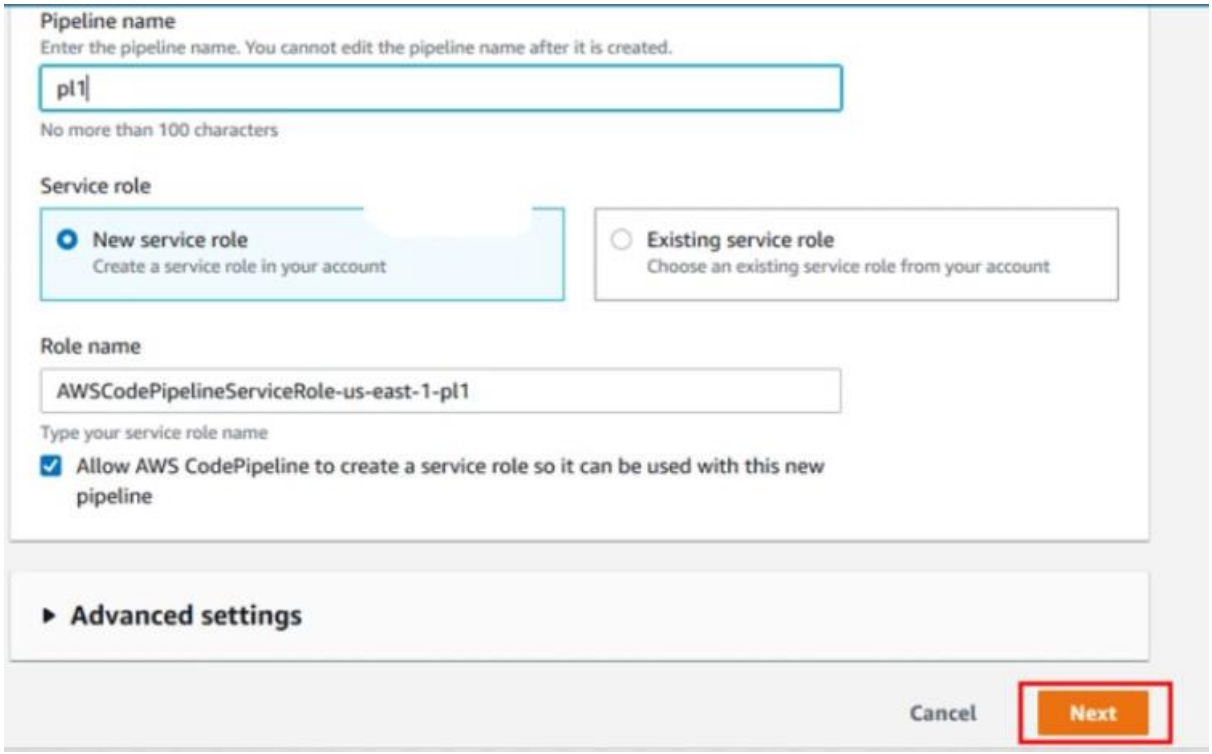
Deployment group : mygrp
Revision type: My application is stored in Amazon S3
Revision location : s3://gir-sampleapp/sampleapp.zip



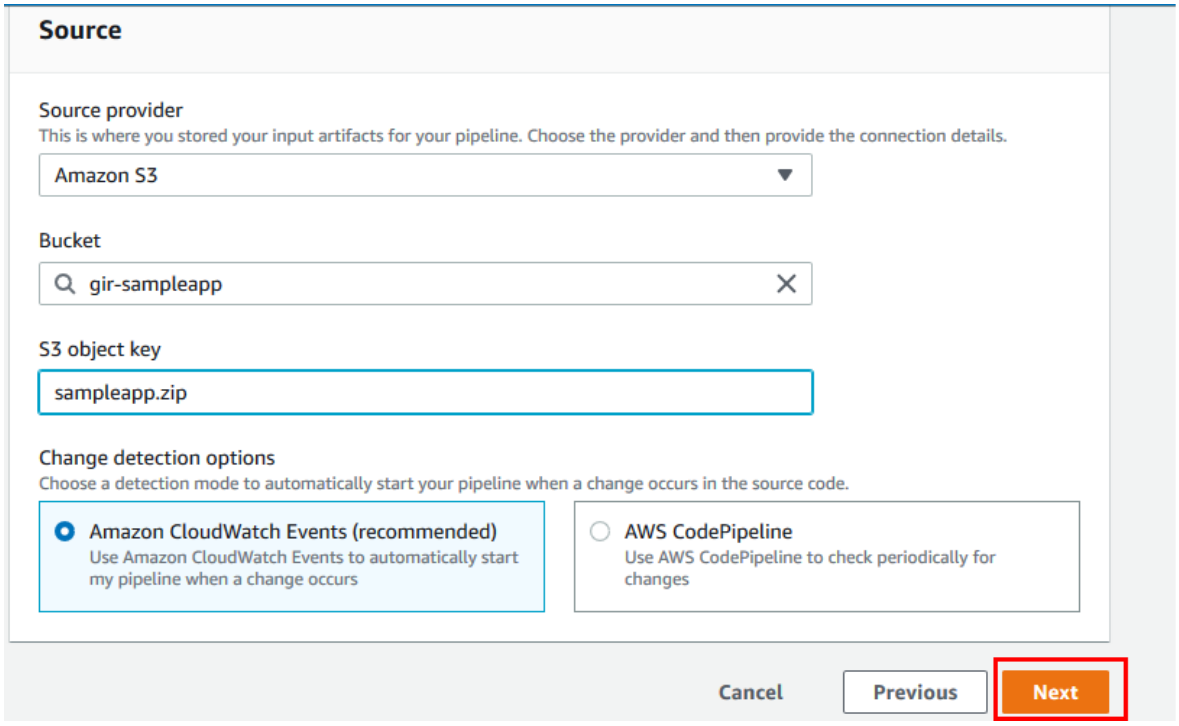
Complete the Deployment. Once the Deployment process is completed copy the public IP address of the webserver and paste to the web browser. You should see contents of index.html (version 1)

8 - Create AWS CodePipeline

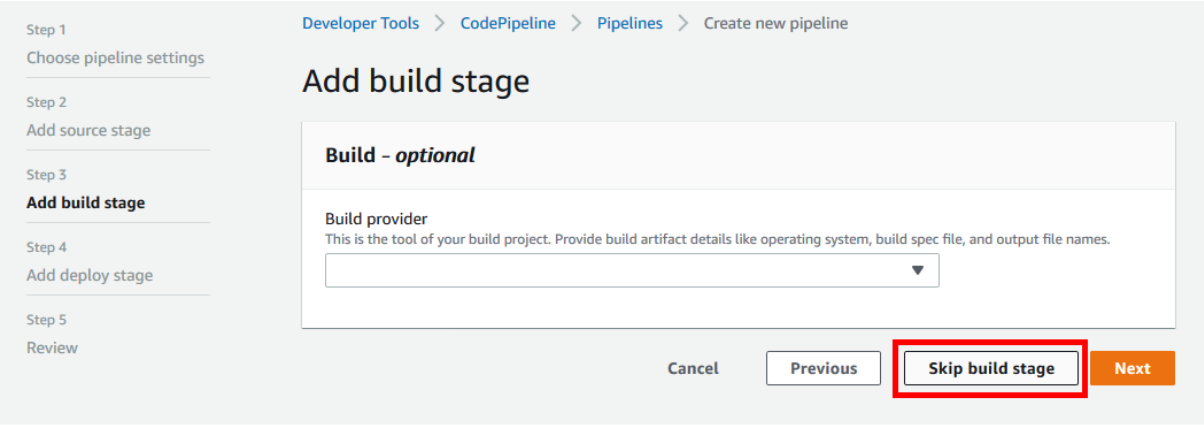
Click *Create Pipeline* and enter the values like in the screen shot



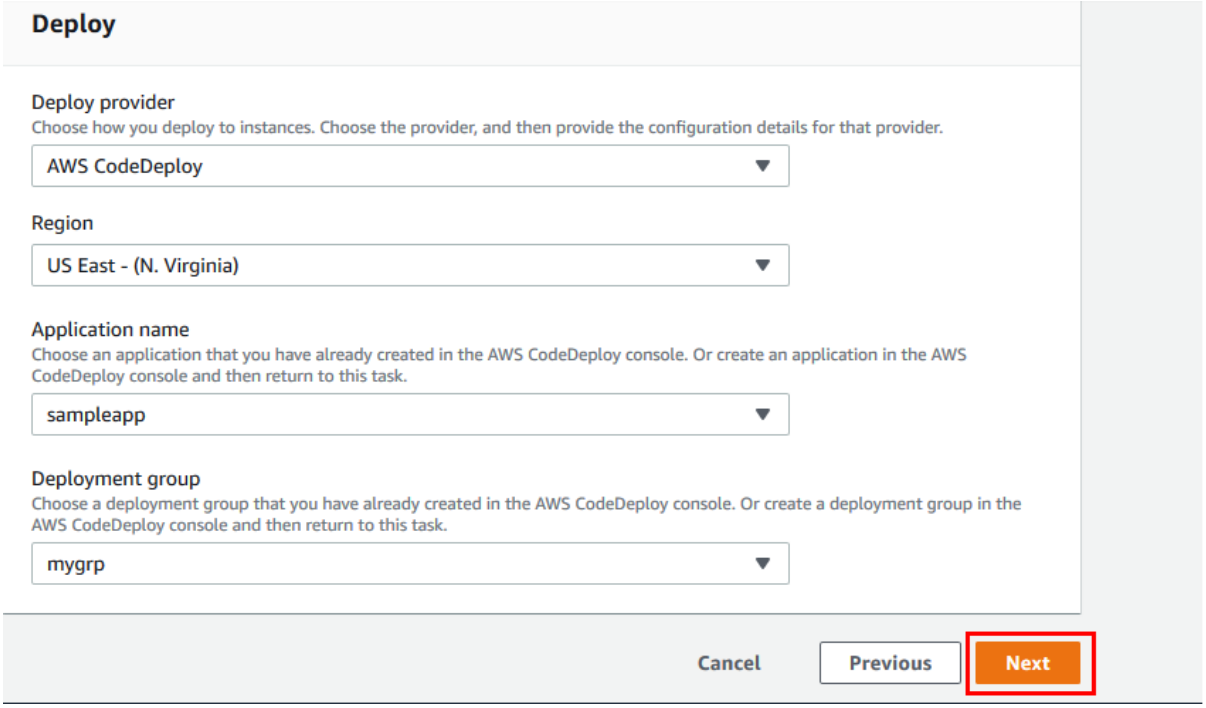
Enter the Source section and click next



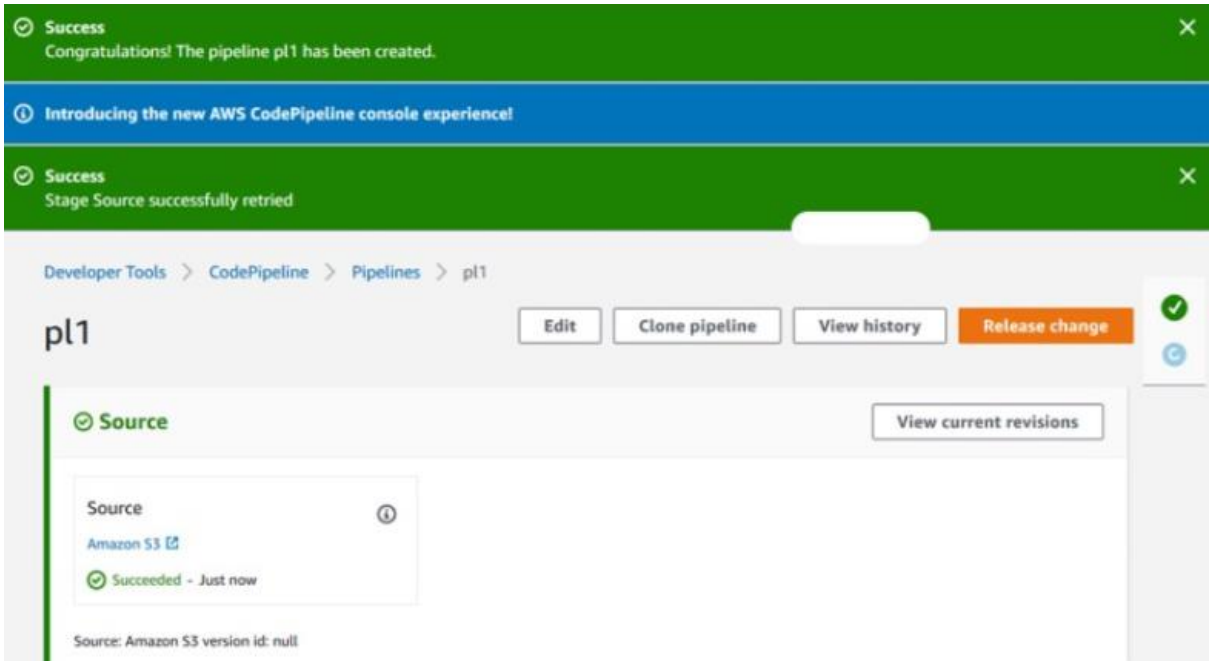
In the Add Build Stage, Click *Skip Build Stage* button and then confirm the skip.



In the Deploy stage Enter the value like below and Click Next.



Once AWS CodePipeline is created it looks like below.



9 -Upload the version 2 of code to S3 bucket and test the setup

a - Now it the time to test AWS CodePipeline. SSH to the Developer machine to upload the Version 2 of the code.

```
# cd /root/deploy_dir2
# aws s3 cp sampleapp.zip s3://gir-sampleapp
```

b - Wait for a few minutes for CloudWatch to detect the new upload to S3 bucket. Refresh the webbrowser. Now you should see contents of index.html ("Version 2" Text")

c- Let us see the history of execution in AWS CodePipeline Console.