**AWS CodeDeploy & CodePipeline**

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 we have to use AWS CodePipeline also.

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.

We use AWS CodeDeploy along with AWS Codepipeline for this experiment. Initial version of the code is 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.

**Setup in Brief:**

I have used two EC2 instance of Amazon Linux. First one is the web server we will be configuring it, also install CodeDeploy agent. Second EC2 machine it 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 by 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
9. Create a CodePipeline
10. Upload the version 2 of code to S3 bucket and test the setup

**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. 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, we can use any specific Tag name that is easily identifiable.

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 - Launch 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.*

1. SSH to developer machine.  I have su to root and a created a directory /root/deploy\_dir and I have created another directory sampleapp inside deploy\_dir
2. *The code should contain a file*appspec.yml*. This*file*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.*

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

c - Then create index.html file

vi index.html

<html>

<h2> Sample App Version 1 </h2>

</html>

d – create another directory script under this file place the following script in separate file

vi httpd\_install.sh

#!/bin/bash

yum install -y httpd

vi httpd\_start.sh

#!/bin/bash

systemctl start httpd

systemctl enable httpd

vi httpd\_stop.sh

#!/bin/bash

systemctl stop httpd

systemctl disable httpd

e - At last give executable permission to the files created, if is not given it is not going to work anymore.

**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

Note - The destination of your s3 bucket should be specified

**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

At last 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 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 of the webserver from the browser and see that it is working

It should show a result of Sample App Version 1

**9) Create AWS CodePipeline**

Click *Create Pipeline* and enter the pipeline name in my case it is my pipeline.

In service role click new service role it will automatically create a new role based on your pipeline name for connection between codedeploy and codepipeline.

Click next in source stage choose s3, choose our bucket and s3 object key.

Now choose AWS cloudwatch events to monitor

Skip add build stage in deploy stage choose codedeploy, choose our application name and deployment group.

Create pipeline, now it will show an error. Go to s3 and enable bucket versioning.

**10) Code version 2**

Now change the index.html file from Sample App Version 1 to Sample App Version 2 and push the code to s3, now it will deploy automatically.

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

It should show a result of Sample App Version 2.