

CODE DEPLOY AND CODE PIPELINE PROJECT

Balashundaram A

OBJECTIVE:

Complete CI/CD with AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline

Step 1: IAM Roles for EC2-S3-CodeDeploy access

a) S3 full access

The screenshot shows a list of AWS IAM policies filtered by 's3'. The 'AmazonS3FullAccess' policy is selected, indicated by a checked checkbox. Other policies listed include AmazonDMSRedshiftS3Role, AmazonS3ObjectLambdaExecutionRolePolicy, AmazonS3OutpostsFullAccess, AmazonS3OutpostsReadOnlyAccess, AmazonS3ReadOnlyAccess, IVSRecordToS3, and QuickSightAccessForS3StorageManagementAnalyticsReadOnly. All policies have 'None' listed under 'Used as'.

Policy name	Used as
AmazonDMSRedshiftS3Role	None
AmazonS3FullAccess	None
AmazonS3ObjectLambdaExecutionRolePolicy	None
AmazonS3OutpostsFullAccess	None
AmazonS3OutpostsReadOnlyAccess	None
AmazonS3ReadOnlyAccess	None
IVSRecordToS3	None
QuickSightAccessForS3StorageManagementAnalyticsReadOnly	None

b) Code-Deploy role

The screenshot shows the 'Attached permissions policies' step of the 'Create role' wizard. It displays a single result for the 'AWSCodeDeployRole' policy, which provides CodeDeploy service access. The wizard has four steps, with step 2 highlighted.

Policy name	Used as	Description
AWSCodeDeployRole	None	Provides CodeDeploy service access to expand...

Step 2: Server Deployment

- 1) Creating an IAM user with full **Administrative access** and login.
- 2) Creating an Amazon Linux EC2 server 1 for the developer to create and change the codes.
- 3) Creating another Linux server 2 for Code Deploy operations.

```
root@ip-172-31-5-82:~  
└─ login as: ec2-user  
└─ Authenticating with public key "imported-ssh-key"  
  
      _\   _ /_ )_ Amazon Linux 2 AMI  
     _\_\_|\_ |  
  
https://aws.amazon.com/amazon-linux-2/  
11 package(s) needed for security, out of 35 available  
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-31-5-82 ~]$ sudo su -  
[root@ip-172-31-5-82 ~]# aws configure  
AWS Access Key ID [None]: AKIA6KKT2J4XQ6KIKBDN  
AWS Secret Access Key [None]: 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  
Default region name [None]: ap-south-1  
Default output format [None]: json  
[root@ip-172-31-5-82 ~]# aws s3 ls  
2021-09-23 11:11:08 createee  
[root@ip-172-31-5-82 ~]#
```

```
root@ip-172-31-43-214:~  
└─ login as: ec2-user  
└─ Authenticating with public key "imported-ssh-key"  
  
      _\   _ /_ )_ Amazon Linux 2 AMI  
     _\_\_|\_ |  
  
https://aws.amazon.com/amazon-linux-2/  
11 package(s) needed for security, out of 35 available  
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-31-43-214 ~]$ sudo su -  
[root@ip-172-31-43-214 ~]#
```

In the Server 2, the following commands are executed to install CodeDeploy

- `yum install ruby -y`
-> to install ruby
- `yum install wget -y`
-> to install the wget package
- `wget https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install`
-> wget command is used to download the files directly from the web.
- `chmod +x install`
-> changing permissions
- `./install auto`
- `service codedeploy-agent status`

```
[root@ip-172-31-43-214 ~]# service codedeploy-agent status
The AWS CodeDeploy agent is running as PID 4326
[root@ip-172-31-43-214 ~]#
```

Step 3: Creating the files required for the webpage from Developer machine

- 1) **Index file** - vi index.html
 - 2) **Yaml file** - vi abc.yml

This file helps to deploy the source code into webserver automatically.

- 3) Creating httpd install , start and stop files which is getting called in yaml file.

```
vi httpd_install.sh
```

```
#!/bin/bash
```

```
yum install -y httpd
```

vi httpd_start.sh

```
#!/bin/bash
```

```
systemctl start httpd
```

vi httpd_stop.sh

```
#!/bin/bash
```

```
systemctl stop httpd
```

Step 4: Creating Codedeploy Application and Pushing the code to S3 bucket from Server 1

- a) Creating a Bucket with **Public** policy and **Bucket Versioning** as Enabled.
- b) Creating an application in the code deploy using CLI,

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

```
[root@ip-172-31-5-82 scripts]# aws deploy create-application --application-name sampleapp
{
    "applicationId": "be74e27b-61e6-4241-b1c6-a493d9cab9de"
}
[root@ip-172-31-5-82 scripts]#
```

- c) Uploading the code to S3 by executing the command below,

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

- d) Now in the s3 bucket sampleapp.zip is present.

Step 5: Creating a **Deployment Group** to include Server 2 using AWS Management Console.

Step 6: Creating a **Deployment** which pushes code to the Server 2.

Step 7: Creating a **Pipeline** enabling automatic deployment the moment when the new version reaches the S3 repository.



And DONE! Now whenever the Developer changes the code and pushes it to S3 repository the Pipeline recognizes it and triggers services like Code Commit and Code Deploy resulting in changes in the webpage.

Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishers/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name
mytopic
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

Display name - optional
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. [Info](#)

My Topic
Maximum 100 characters, including hyphens (-) and underscores (_).

Encryption - optional
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

Access policy - optional
This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic. [Info](#)

Choose method
 Basic
Use simple criteria to define a basic access policy.
 Advanced
Use a JSON object to define an advanced access policy.

Define who can publish messages to the topic

Only the topic owner
Only the owner of the topic can publish to the topic

Everyone
Anybody can publish

Only the specified AWS accounts
Only the specified AWS account IDs can publish to the topic

Define who can subscribe to this topic

Only the topic owner
Only the owner of the topic can subscribe to the topic

Everyone
Any AWS account can subscribe to the topic

Only the specified AWS accounts
Only the specified AWS account IDs can subscribe to the topic

Only requesters with certain endpoints

JSON preview

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "SNS:Publish",
        "SNS:RemovePermission",
        "SNS:SetTopicAttributes",
        "SNS:DeleteTopic",
        "SNS>ListSubscriptionsByTopic"
      ]
    }
  ]
}
```

Enabling the SNS Notification service to alert us for the deployment success, fail, start, stop etc.