AWS CODEDEPLOY AND CODEPIPELINE

INTRODUCTION:

This project will demonstrate how to use AWS CodePipeline and AWS CodeDeploy to automate the build, test, and deployment of a simple web application hosted on EC2 instances. AWS CodePipeline is a fully managed continuous integration and continuous delivery service that helps you automate the software release process. AWS CodeDeploy is a fully managed deployment service that automates the deployment of applications to Amazon EC2 instances, on-premises servers, or AWS Lambda functions.

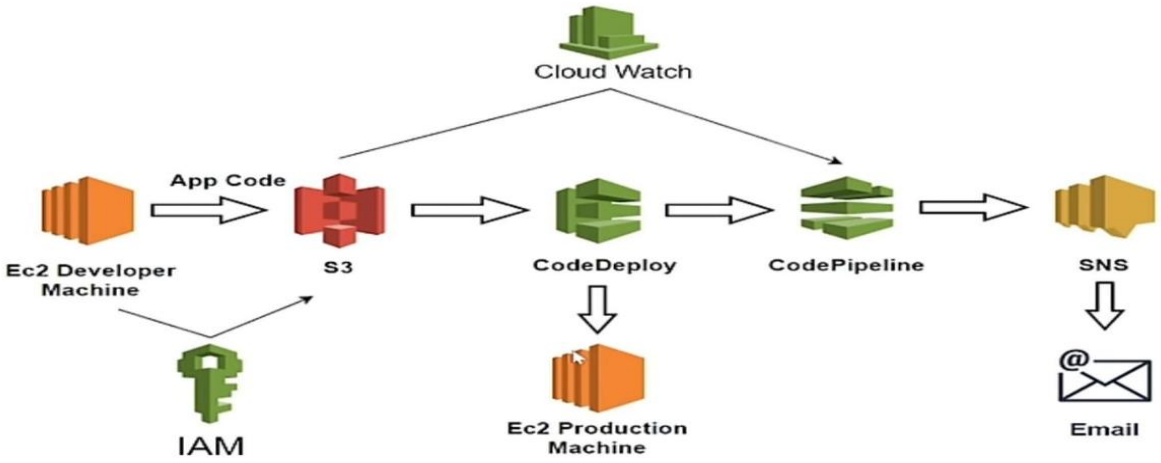
In this project, we will create a simple web application and then use CodePipeline to automate the deployment process.

The code will be deployed to EC2 instances using CodeDeploy. Once the pipeline is set up, we can make changes to the code and the pipeline will automatically rebuild, test, and deploy the code. This will help us to reduce the time it takes to get new features and bug fixes into production.

We will also show how to use CodePipeline to create different environments for development, testing, and production. This will help us to ensure that our code is thoroughly tested before it is deployed to production.

Finally, we will show how to use CodePipeline to roll back a deployment if something goes wrong. This will help us to minimize the impact of deployment failures.

This project will be a valuable resource for anyone who wants to learn how to use AWS CodePipeline and CodeDeploy to automate their software delivery process.



OBJECTIVES:

Objectives: Unleashing Efficiency and Reliability with AWS CodeDeploy and CodePipeline

- ✓ **CodeDeploy Objectives: Streamlined Deployments for Seamless Application Updates**

- **Automate Repetitive Tasks:** Free your development team from the manual burden of application deployments. CodeDeploy automates routine tasks like deploying code packages, configuring environments, and managing configurations across various compute services. This frees up valuable developer time for innovation and feature development.

- **Minimize Downtime - Keep Your Applications Online and Available:** CodeDeploy understands the critical nature of application uptime. It offers deployment strategies like rolling updates, blue/green deployments, and in-place deployments. These strategies minimize or eliminate downtime during application updates, ensuring a seamless user experience.

- **Tailored Deployment Approaches - Matching the Risk to Your Needs:** Not all applications require the same level of caution during updates. CodeDeploy empowers you with a flexible deployment toolbox. Choose from canary deployments for controlled rollouts to a small subset of users, linear deployments for gradual rollouts across instances, or all-at-once deployments for rapid updates. These options allow you to balance risk and rollout speed based on your application's needs.

- **Rollback Safety Net - Peace of Mind with Automated Version Rollbacks:** Deployment failures happen. CodeDeploy provides a robust rollback mechanism. You can configure automatic rollbacks based on predefined failure conditions or initiate manual rollbacks if needed. This functionality ensures a quick recovery path in case of unforeseen issues during deployments.
- **Centralized Management Hub - Command and Control Over Your Deployments:** CodeDeploy offers a centralized dashboard to manage all your deployments across environments. Track deployment progress in real-time, monitor success metrics like deployment time and error rates, and analyze logs for troubleshooting. This central control center keeps you informed and allows you to make informed decisions regarding your application's health.

✓ CodePipeline Objectives: Automating the Journey from Code to Production

- **End-to-End Automation - Streamline the Software Delivery Pipeline:** CodePipeline orchestrates the entire software release process, from code commit to production deployment. This eliminates the need for manual steps and handoffs, leading to a faster and more reliable delivery process. Focus on innovation while CodePipeline handles the repetitive tasks.

- **Embrace CI/CD - Deliver Faster and More Frequently:** CodePipeline plays a pivotal role in facilitating Continuous Integration and Continuous Delivery (CI/CD) practices. It integrates seamlessly with other AWS services like CodeBuild for

building and testing your code, and CodeDeploy for deployment to various environments. Additionally, it allows integration with third-party tools for a comprehensive CI/CD pipeline.

- **Customization is Key - Building a Workflow Tailored to Your Needs:** Your development process is unique. CodePipeline empowers you to design custom workflows with distinct stages such as source code retrieval, build and test stages, and finally, deployment. You can define the sequence of these stages and integrate specific tools within each stage to ensure your code undergoes rigorous testing before going live.

- **Real-Time Visibility - Identify and Address Issues Proactively:** Stay informed at every step of the release process. CodePipeline provides real-time updates on the status of your pipelines. You can identify any issues or bottlenecks early, allowing for swift troubleshooting and intervention before they impact the deployment process.

- **Scalability for Growth - Handle Complex Workflows with Confidence:** As your application and development team grow, your CI/CD pipelines need to scale accordingly. CodePipeline is built to handle complex workflows with numerous stages and integrations. Additionally, its seamless integration with other AWS services and third-party tools enables you to create a robust and scalable CI/CD pipeline that adapts to your evolving needs.

By leveraging AWS CodeDeploy and CodePipeline together, you can achieve a highly efficient and reliable software development lifecycle. CodeDeploy automates routine deployments and minimizes downtime, while CodePipeline orchestrates the entire release process. This combination empowers you to deliver high-quality applications faster and more frequently.

AWS SERVICES USED:

1. **IAM (Identity and Access Management):**

Role: IAM is responsible for managing access to AWS services and resources securely.

Functionality:

- **IAM Roles:** Define permissions to grant the necessary access to AWS services for various components of your CI/CD pipeline, such as CodePipeline, CodeDeploy, S3 and EC2 instances.
- **IAM Policies:** Attach policies to roles to restrict or allow specific actions, ensuring that services only have the permissions they need.
- **User Authentication:** Manage user access, ensuring that developers and administrators have the right level of access to interact with the CI/CD pipeline.

2. EC2 (Elastic Compute Cloud)

Role: EC2 instances are the compute resources where your application will be deployed and run.

Functionality:

- **Application Hosting:** EC2 instances host your web application after it's deployed through the CI/CD pipeline.
- **Deployment Target:** CodeDeploy deploys your application code to EC2 instances, which are registered in deployment groups.
- **Auto Scaling:** EC2 can be configured to automatically scale based on traffic, ensuring high availability and reliability.

3. S3 (Simple Storage Service)

Role: S3 is used to store assets like application code, configuration files, and deployment artifacts.

Functionality:

- **Artifact Storage:** Store the build artifacts (e.g., zip files, executables) that are generated during the CI/CD process.
- **Source Files:** Can store application source code if integrated with CodePipeline.
- **Backup:** Use S3 to back up critical configuration files or data that may need to be retrieved during deployment.

4. CodeDeploy

Role: CodeDeploy automates the deployment of your application to EC2 instances, Lambda functions, or on-premises servers.

Functionality:

- **Deployment Strategies:** Supports different deployment strategies such as in-place (replacing existing versions) or blue/green (deploying to new instances before rerouting traffic).
- **Deployment Groups:** Organize EC2 instances or Lambda functions into groups for targeted deployments.
- **Rollbacks:** Automatically rolls back to a previous version if a deployment fails, minimizing downtime.

5. CodePipeline

Role: CodePipeline orchestrates the CI/CD process, automating the steps required to release your software.

Functionality:

- **Pipeline Stages:** Define stages such as Source, Build, Test, and Deploy. Each stage represents a step in your CI/CD process.
- **Integration:** Integrates with CodeCommit, GitHub, or S3 for source control; with CodeBuild or third-party build services; and with CodeDeploy for deployment.
- **Continuous Integration:** Automatically triggers the pipeline when changes are detected in the source repository, ensuring that the latest code is always tested and deployed.

6. Cloud Watch

Role: CloudWatch is used for monitoring and logging across various AWS services, including CodeDeploy and CodePipeline.

Functionality:

- **Metrics Monitoring:** CloudWatch monitors metrics related to your deployments, such as deployment success/failure rates, pipeline execution times, and resource utilization.

- **Logs:** CloudWatch Logs store detailed logs generated during CodeDeploy deployments and CodePipeline executions. These logs include information on deployment events, errors, and the output of deployment scripts.
- **Alarms:** CloudWatch Alarms can be set up based on specific metrics (e.g., failure rates, execution times) to trigger notifications or automated responses when thresholds are breached.

7. SNS (Simple Notification Service)

Role: SNS is used for sending notifications based on events occurring in your CodeDeploy and CodePipeline processes.

Functionality:

- **Notifications:** SNS sends notifications (e.g., email, SMS, or HTTP/HTTPS) to inform you or your team about important events like deployment failures, pipeline state changes (e.g., pipeline succeeded, failed, or stopped), or specific CloudWatch Alarms being triggered.

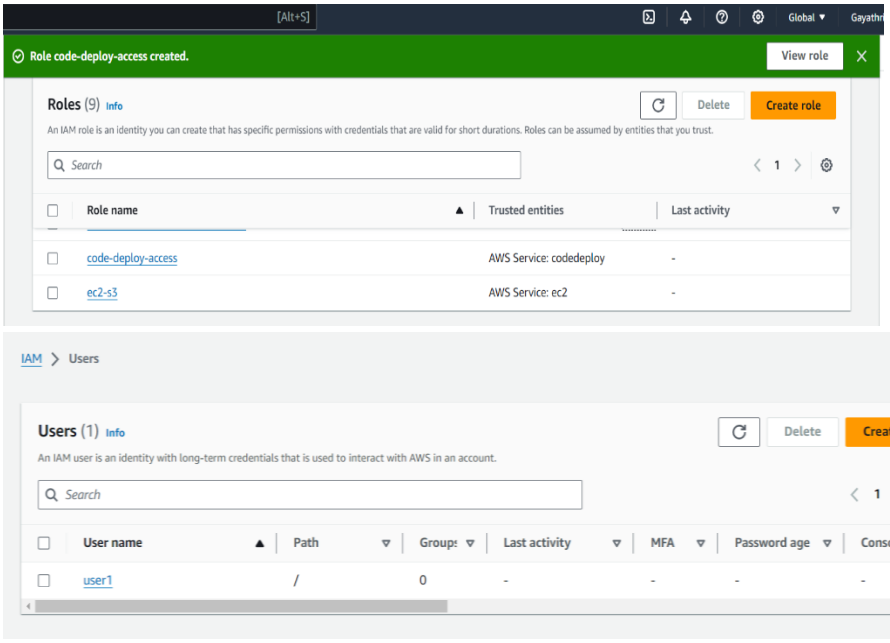
- **Integration with CloudWatch:** SNS can be linked with CloudWatch Alarms to automatically send notifications whenever an alarm is triggered. This ensures you're instantly informed of any critical issues during your CI/CD process.

- **Integration with CodeDeploy and CodePipeline:** In CodeDeploy, SNS can notify you about the start, success, or failure of a deployment. In CodePipeline, SNS can alert you when a pipeline stage starts or completes, helping you track the pipeline's progress.

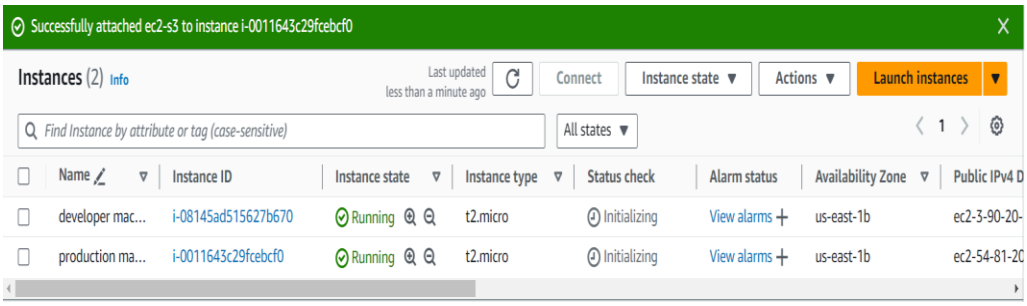
PROJECT IMPLEMENTATION:

To implement a project using AWS CodeDeploy and CodePipeline, one typically follow these steps:

1. **Create role and IAM user:**
   Create two IAM roles: "ec2-s3" and "code-deploy-access".
   Also, create an IAM user for accessing EC2 and S3 services.



2. **Create EC2 instance:**
   Create two EC2 instances: a) Production machine and b) Developer machine



3. **Connect to the instances using PuTTY.**

Connect to the developer machine as the "white" machine and the production machine as the "black" machine using PuTTY.

Also, attach the `ec2-s3` role to the production machine.

4. **Production machine:**
   Step1: For the code deployment process, we need the prerequisite of Ruby. Install it using the command '`yum install ruby -y`'.

```
[ec2-user@ip-172-31-22-238 ~]$ sudo -i
[root@ip-172-31-22-238 ~]# yum update
Last metadata expiration check: 0:06:10 ago on Wed Aug 21 06:22:11 2024.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-22-238 ~]# yum install ruby -y
Last metadata expiration check: 0:06:25 ago on Wed Aug 21 06:22:11 2024.
Dependencies resolved.
================================================================================
 Package                  Architecture        Version                  Repository
================================================================================
Installing:
 ruby3.2                  x86_64              3.2.2-180.amzn2023.0.3   amazonlinux
Installing dependencies:
 ruby3.2-default-gems     noarch              3.2.2-180.amzn2023.0.3   amazonlinux
 ruby3.2-libs             x86_64              3.2.2-180.amzn2023.0.3   amazonlinux
 ruby3.2-rubygem-io-console x86_64            0.6.0-180.amzn2023.0.3   amazonlinux
 ruby3.2-rubygem-json     x86_64              2.6.3-180.amzn2023.0.3   amazonlinux
 ruby3.2-rubygem-psych    x86_64              5.0.1-180.amzn2023.0.3   amazonlinux
```

Step 2: Install the CodeDeploy agent via the CLI using the command `wget https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install`.

```
root@ip-172-31-22-238:~
Package wget-1.21.3-1.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-22-238 ~]# wget https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install
--2024-08-21 06:30:25--  https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install
Resolving aws-codedeploy-us-east-1.s3.amazonaws.com (aws-codedeploy-us-east-1.s3.amazonaws.com)... 52.217.42.196, 52.217.82.236, 52.217
.124.73, ...
Connecting to aws-codedeploy-us-east-1.s3.amazonaws.com (aws-codedeploy-us-east-1.s3.amazonaws.com)|52.217.42.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19045 (19K) []
Saving to: 'install'

install                     100%[=================================================>]  18.60K  --.-KB/s    in 0.001s

2024-08-21 06:30:25 (23.4 MB/s) - 'install' saved [19045/19045]

[root@ip-172-31-22-238 ~]# ls
install
```

Step 3: Grant executable permissions to the installed agent using the command ‘chmod +x install’.

```
2024-08-21 14:48:17 (125 MB/s) - 'install' saved [19045/19045]

[root@ip-172-31-45-167 ~]# ls
install
[root@ip-172-31-45-167 ~]# chmod +x install
[root@ip-172-31-45-167 ~]# ls
install
[root@ip-172-31-45-167 ~]#
```

Step 4: Execute the file (‘ ./install auto’)

```
[root@ip-172-31-22-238 ~]# ./install auto
I, [2024-08-21T06:38:06.875650 #26119]  INFO -- : Starting Ruby version check.
W, [2024-08-21T06:38:06.875922 #26119]  WARN -- : The Ruby version in /usr/bin/ruby3.2 is 3.2.2, . Attempting to install anyway.
I, [2024-08-21T06:38:06.876069 #26119]  INFO -- : Starting update check.
I, [2024-08-21T06:38:06.876199 #26119]  INFO -- : Attempting to automatically detect supported package manager type for system...
I, [2024-08-21T06:38:06.894405 #26119]  INFO -- : Checking AWS_REGION environment variable for region information...
I, [2024-08-21T06:38:06.894572 #26119]  INFO -- : Checking EC2 metadata service for region information...
I, [2024-08-21T06:38:06.906646 #26119]  INFO -- : Checking AWS_DOMAIN environment variable for domain information...
I, [2024-08-21T06:38:06.906827 #26119]  INFO -- : Checking EC2 metadata service for domain information...
I, [2024-08-21T06:38:06.911486 #26119]  INFO -- : Downloading version file from bucket aws-codedeploy-us-east-1 and key latest/LATEST_V
ERSION...
I, [2024-08-21T06:38:06.911743 #26119]  INFO -- : Endpoint: https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/LATEST_V
ERSION
I, [2024-08-21T06:38:07.000479 #26119]  INFO -- : Downloading package from bucket aws-codedeploy-us-east-1 and key releases/codedeploy-
agent-1.7.0-92.noarch.rpm...
I, [2024-08-21T06:38:07.000715 #26119]  INFO -- : Endpoint: https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/releases/codede
ploy-agent-1.7.0-92.noarch.rpm
I, [2024-08-21T06:38:07.127051 #26119]  INFO -- : Executing `/usr/bin/yum -y localinstall /tmp/codedeploy-agent-1.7.0-92.noarch.tmp-202
40821-26119-wzlzmy.rpm`...
Last metadata expiration check: 0:15:56 ago on Wed Aug 21 06:22:11 2024.
Dependencies resolved.
```

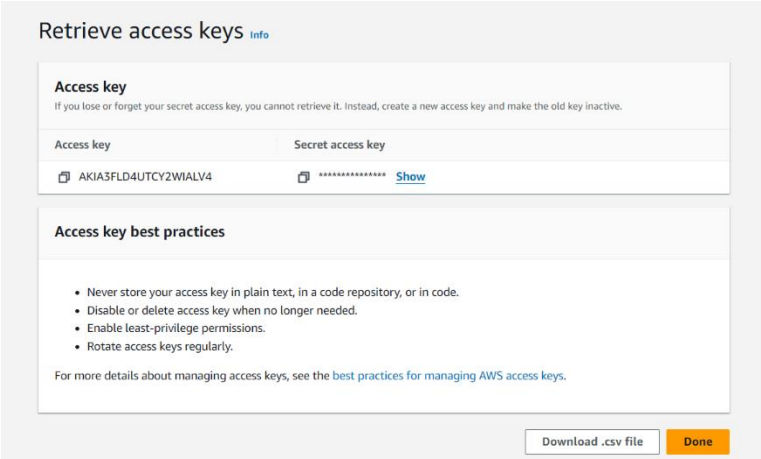
Step 5: Check the CodeDeploy agent status (‘`service codedeploy-agent status`’).

```
Installed:
  codedeploy-agent-1.7.0-92.noarch

Complete!
I, [2024-08-21T14:51:01.114132 #26382]  INFO -- : Update check complete.
I, [2024-08-21T14:51:01.114342 #26382]  INFO -- : Stopping updater.
[root@ip-172-31-45-167 ~]# service codedeploy-agent status
The AWS CodeDeploy agent is running as PID 26529
[root@ip-172-31-45-167 ~]#
```

5. **Developer machine:**

Step 1: Create an access key and a secret access key for the IAM user.



Step 2: Connect the IAM user to the developer machine using the access key.

```
root@ip-172-31-39-29:~
login as: ec2-user
Authenticating with public key "ubuntu.ppk"
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-39-29 ~]$ sudo -i
[root@ip-172-31-39-29 ~]# aws configure
AWS Access Key ID [None]: AKIA3FLD4UTCY2WIALV4
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
[root@ip-172-31-39-29 ~]#
```

Step 3: Create two directories

```
[root@ip-172-31-28-24 ~]# mkdir deploy-dir
[root@ip-172-31-28-24 ~]# cd deploy-dir
[root@ip-172-31-28-24 deploy-dir]# mkdir sampleapp
[root@ip-172-31-28-24 deploy-dir]# cd sampleapp
[root@ip-172-31-28-24 sampleapp]#
```

Step 4: Create index.html

Also, create the ‘`appspec.yaml`’ file (‘`vi appspec.yaml`’). This file helps deploy the source code to the web server automatically.

```
[root@ip-172-31-28-24 ~]# mkdir deploy-dir
[root@ip-172-31-28-24 ~]# cd deploy-dir
[root@ip-172-31-28-24 deploy-dir]# mkdir sampleapp
[root@ip-172-31-28-24 deploy-dir]# cd sampleapp
[root@ip-172-31-28-24 sampleapp]# vi index.html
[root@ip-172-31-28-24 sampleapp]# vi appspec.yaml
[root@ip-172-31-28-24 sampleapp]#
```

Step 5: Inside the YAML script,

```
root@ip-172-31-39-29:~/deploy-dir/sampleapp
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
```

Step 6: Create a directory named `scripts`, and inside the '`scripts`' directory, create the following files:

a) '`vi httpd_install.sh`'

b) '`vi httpd_start.sh`'

c) '`vi httpd_stop.sh`'

```
[root@ip-172-31-39-29 sampleapp]# cd scripts/
[root@ip-172-31-39-29 scripts]# vi httpd_install.sh
[root@ip-172-31-39-29 scripts]# vi httpd_start.sh
[root@ip-172-31-39-29 scripts]# vi httpd_stop.sh
[root@ip-172-31-39-29 scripts]#
```

Inside the files

root@ip-172-31-39-29:~/deploy-dir/sampleapp/scripts

```
#!/bin/bash
yum install -y httpd
```

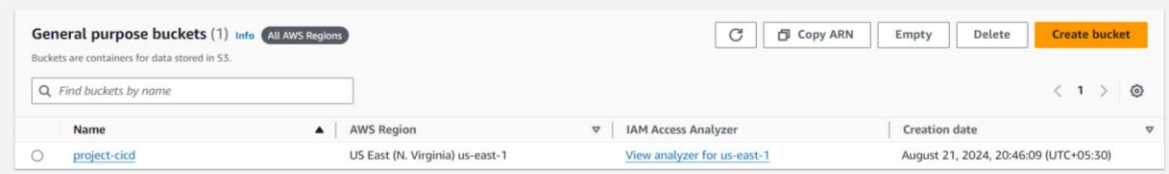
root@ip-172-31-39-29:~/deploy-dir/sampleapp/scripts

```
#!/bin/bash
systemctl start httpd
systemctl enable httpd
```

root@ip-172-31-39-29:~/deploy-dir/sampleapp/scripts

```
#!/bin/bash
systemctl stop httpd
```

6. Create S3 bucket:

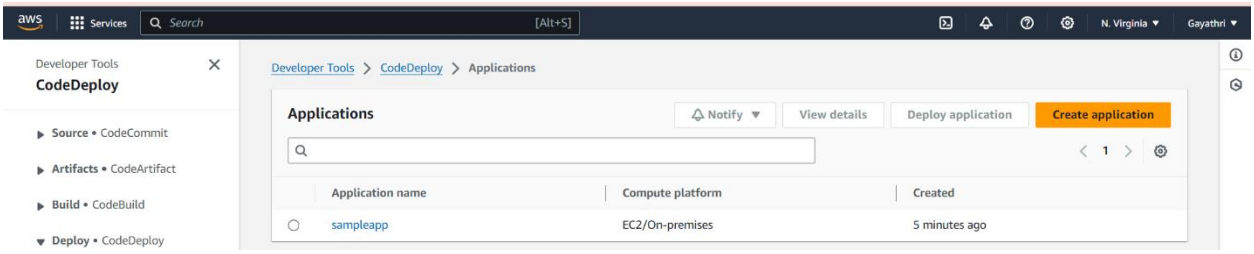


7. Create a CodeDeploy application using the CLI and push the code to an S3 bucket from the developer machine ('`aws deploy create-application --application-name`

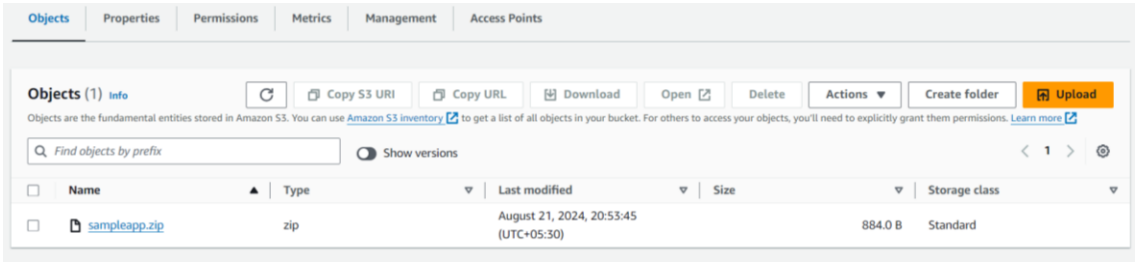
sampleapp’).

```
[root@ip-172-31-39-29 scripts]# cd ..
[root@ip-172-31-39-29 sampleapp]# aws deploy create-application --application-name sampleapp
{
    "applicationId": "c8e646d9-f941-44a6-9557-3e881e383ef1"
}
[root@ip-172-31-39-29 sampleapp]#
```

Check and confirm whether the sample application has been successfully created in the aws console.

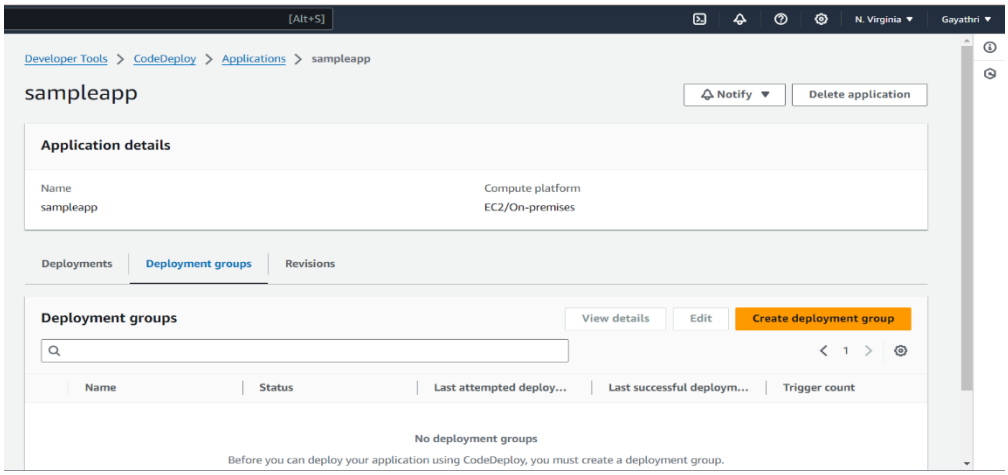


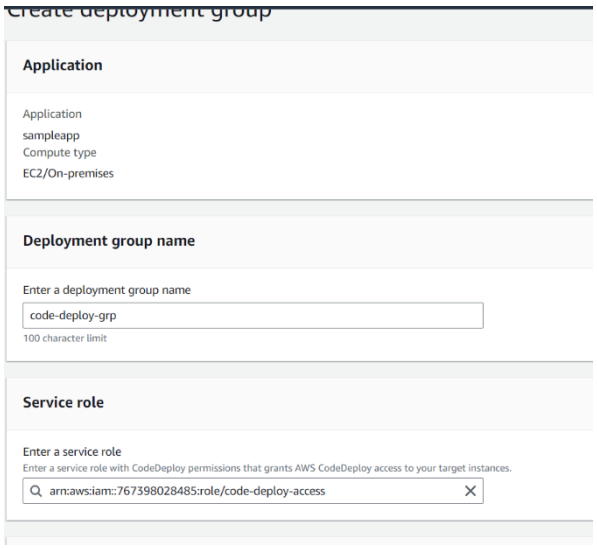
8. Upload the code to an S3 bucket using the CLI command (‘`aws deploy push --application-name sampleapp --s3-location s3://project-cicd/sampleapp.zip`’).

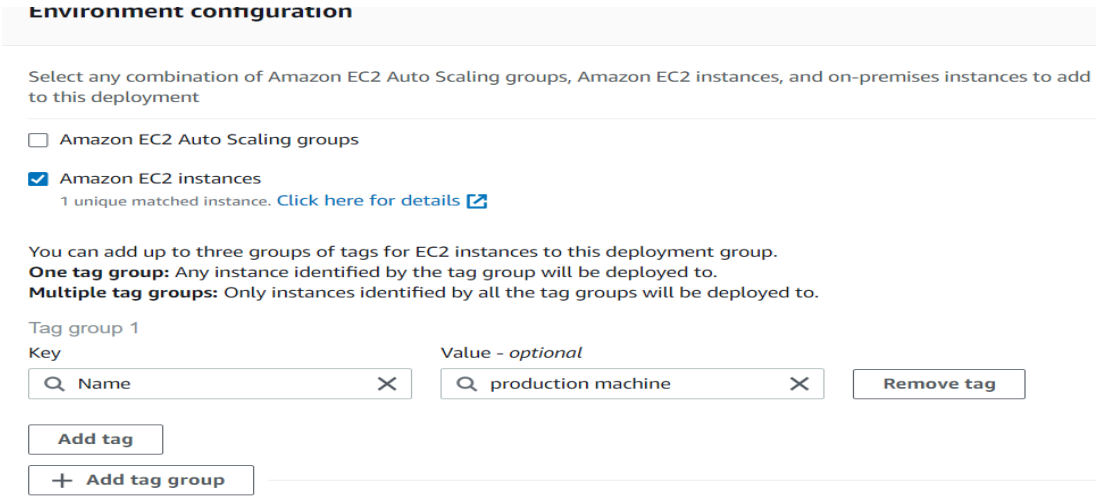


9. Create deployment group:

Select ‘`sampleapp`’ and click "Create Deployment Group" from the Deployment Groups tab.



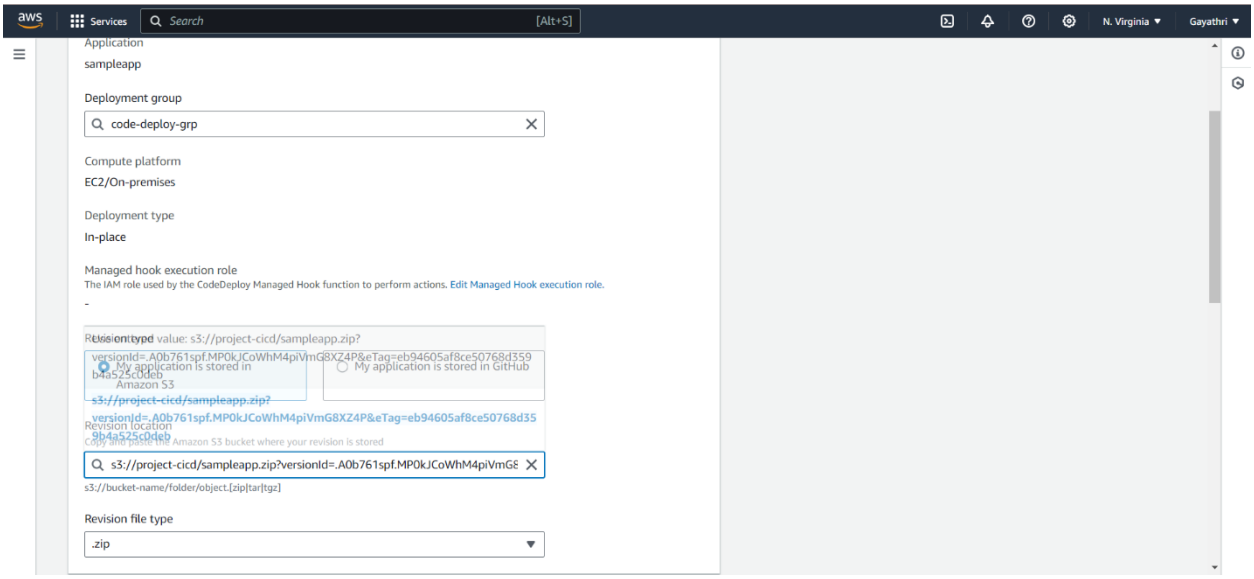




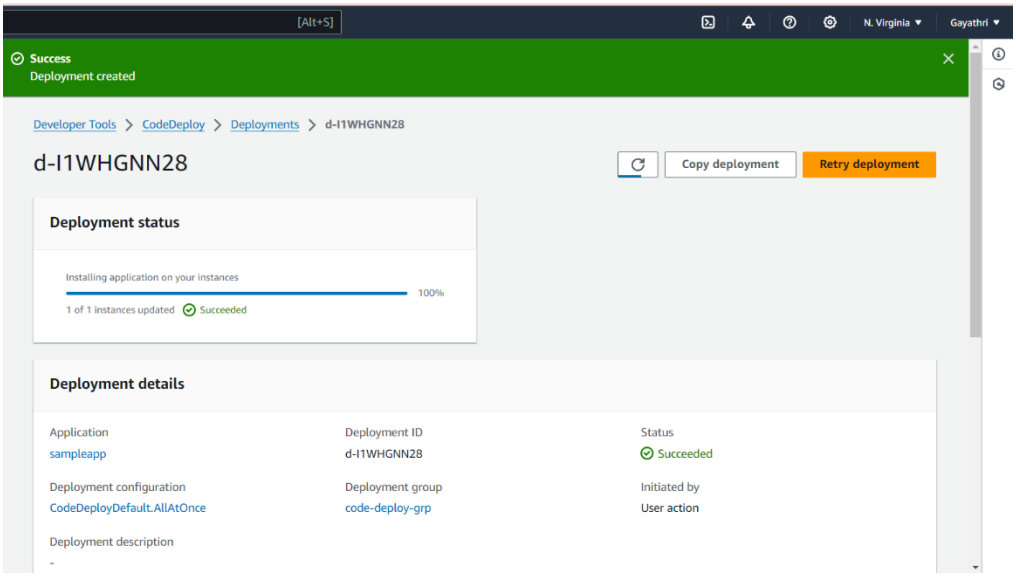
Fill in the required fields.

10. After creating the deployment group, proceed with creating the CodeDeploy deployment.

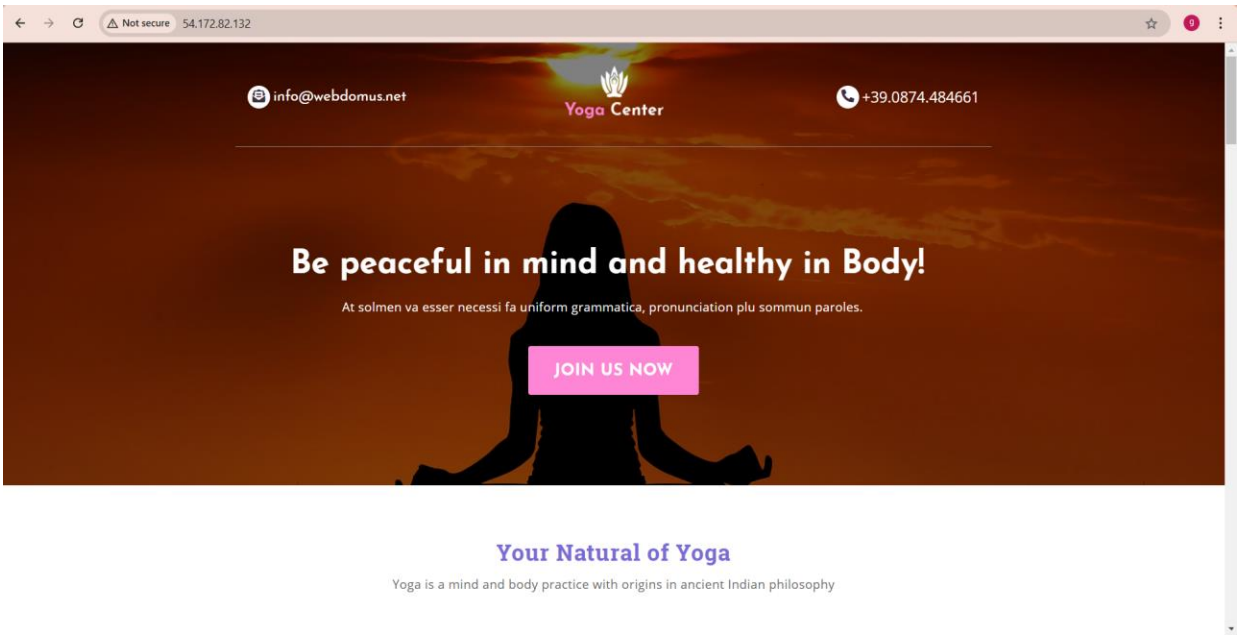
Attach the S3 bucket link in CodeDeploy and fill in the required fields.



11. The deployment status has been checked, and the deployment has been completed successfully.



12. The final output of this code deploy project is given below:

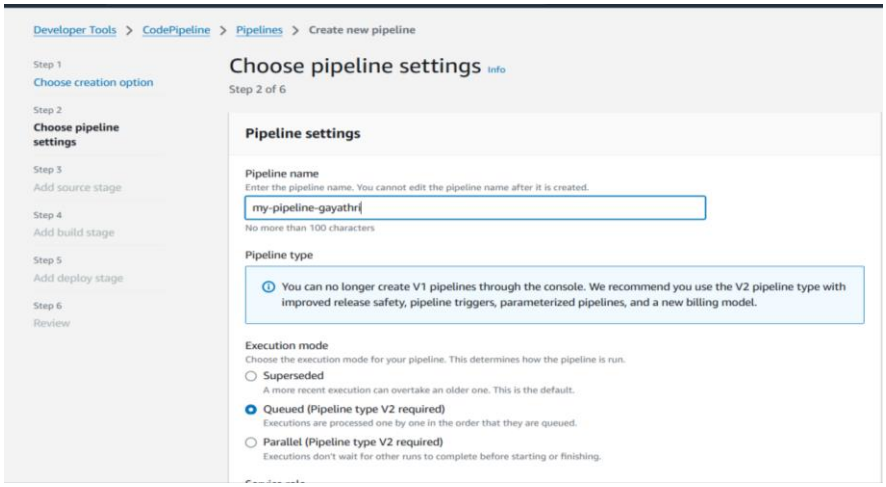


Code deploy managing the deployment process.

CODE PIPELINE:

AWS CodePipeline is a continuous integration and continuous delivery service that automates the steps required to release your software.

1. Go to the codepipeline service and follow the steps



2. Adding the source stage and also enabling the cloudwatch events for monitoring.

Source

Source provider

This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

Amazon S3

Bucket

mywebsiteg3

S3 object key

mywebsiteg3.zip

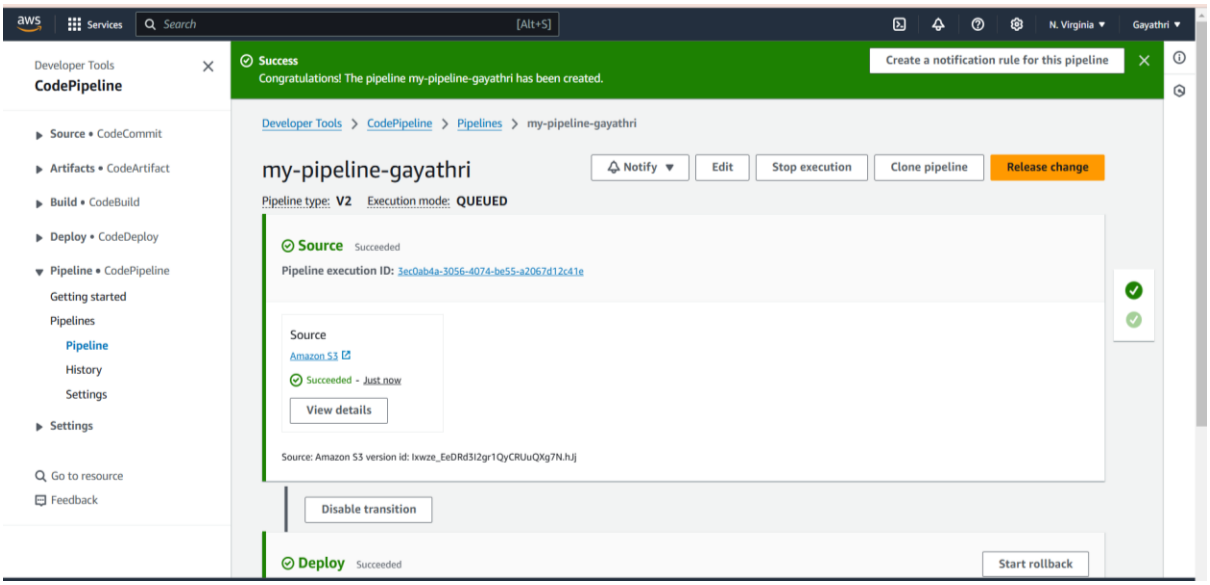
Enter the object key. You can include a file path without the delimiter character (/) at the beginning. Include the file extension. Example: SampleApp.zip

Change detection options

Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

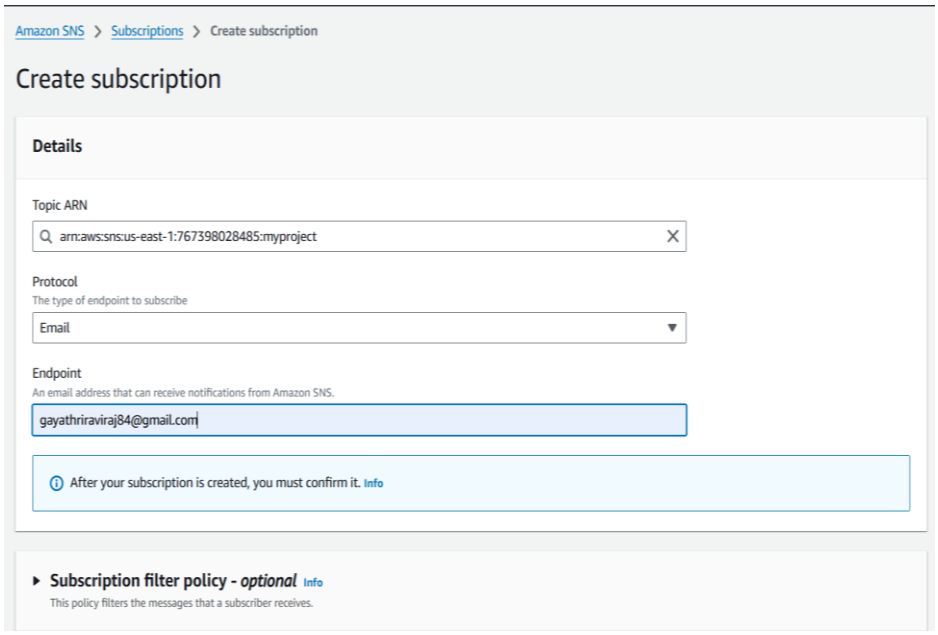
- Amazon CloudWatch Events (recommended)
  Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs
- AWS CodePipeline
  Use AWS CodePipeline to check periodically for changes

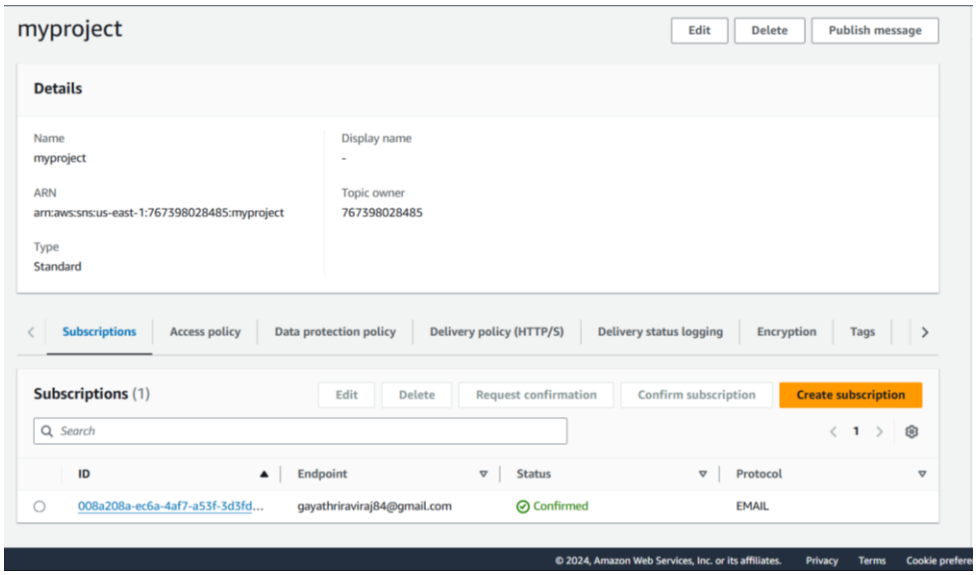
3. The success page of codepipeline CI/CD is



AWS CodePipeline enabled to automate the entire software release process.

4. Enable SNS for receiving deployment notification via email in CI/CD project.


Amazon SNS > Subscriptions > Create subscription
Create subscription
Details
Topic ARN
arn:aws:sns:us-east-1:767398028485:myproject
Protocol
The type of endpoint to subscribe
Email
Endpoint
An email address that can receive notifications from Amazon SNS.
gayathriraviraj84@gmail.com
After your subscription is created, you must confirm it. Info
Subscription filter policy - optional Info
This policy filters the messages that a subscriber receives.


myproject
Edit
Delete
Publish message
Details
Name
myproject
Display name
-
ARN
arn:aws:sns:us-east-1:767398028485:myproject
Topic owner
767398028485
Type
Standard
Subscriptions
Access policy
Data protection policy
Delivery policy (HTTP/S)
Delivery status logging
Encryption
Tags
Subscriptions (1)
Edit
Delete
Request confirmation
Confirm subscription
Create subscription
Search
ID
Endpoint
Status
Protocol
008a208a-ec6a-4af7-a53f-3d3fd...
gayathriraviraj84@gmail.com
Confirmed
EMAIL
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie prefere

CONCLUSION:

In this project, the implementation of a CI/CD pipeline using AWS CodePipeline and Code Deploy has demonstrated a robust, scalable, and automated solution for deploying web applications. The pipeline ensures seamless integration of code changes, thorough testing, and efficient deployment, significantly reducing manual intervention and the risk of errors.

AWS CodePipeline enabled us to automate the entire software release process, from source code retrieval to deployment. This continuous delivery process accelerates development cycles and ensures that new features, bug fixes, and updates are delivered to production swiftly and reliably.

AWS CodeDeploy played a crucial role in managing the deployment process, allowing for flexible deployment strategies like rolling updates, which minimized downtime and maintained application availability. The integration with other AWS services, such as S3 and CloudWatch, provided additional benefits, including versioning control, monitoring, and logging.

Overall, the project showcases the power of AWS CI/CD tools in enhancing the DevOps workflow, improving collaboration between development and operations teams, and ensuring consistent, high-quality software delivery. This approach not only optimizes resource utilization but also aligns with best practices for cloud-native application development and deployment.