**Presented by**

**B. MASHA**

**AWS CODEDEPLOY AND CODEPIPELINE WEB HOSTING**

**Introduction to AWS CodeDeploy and CodePipeline for Web Hosting**

In the dynamic world of web hosting, deploying updates swiftly and efficiently is essential for maintaining a seamless user experience. AWS CodeDeploy and CodePipeline offer robust solutions tailored to streamline the deployment process, ensuring reliability, scalability, and automation every step of the way.

**AWS CodeDeploy:**

AWS CodeDeploy simplifies the deployment of applications to a variety of compute services, including Amazon EC2 instances, Lambda functions, and on-premises servers. With CodeDeploy, you can automate your deployment process, allowing for rapid and consistent releases without downtime. Its intelligent deployment strategies enable you to roll out updates gradually, minimizing risks and maximizing availability.

**AWS CodePipeline:**

AWS CodePipeline orchestrates the continuous delivery process, automating the build, test, and deployment phases of your application. By creating customizable pipelines, you can define the workflow that best suits your development and release cycle. CodePipeline integrates seamlessly with other AWS services, third-party tools, and custom scripts, providing flexibility and extensibility to your deployment workflow.

**Web Hosting Project:**

For a web hosting project, AWS CodeDeploy and CodePipeline offer a powerful combination to manage deployments efficiently. Whether you're launching a new website, updating features, or scaling infrastructure, these services provide the tools to automate and streamline your deployment pipeline. From code changes to production release, you can ensure a seamless delivery process while maintaining control and visibility at every stage.

In this guide, we'll explore how to set up and configure AWS CodeDeploy and CodePipeline for your web hosting project, leveraging their features to achieve reliable, scalable, and automated deployments. From initial setup to advanced deployment strategies, you'll learn how to harness the full potential of these AWS services to optimize your web hosting workflow and deliver exceptional experiences to your users.

**OBJECTIVE**

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

**AWS SERVICES USED**

EC2, S3, CodeDeploy, CodePipeline, CloudWatch, IAM, SNS.

**PROJECT IMPLEMENTATION DETAILS**

Step 1: Create IAM roles for EC2-S3 full access and Code deploy access.

Step 2: Create developer user with full administrator access.

Step 3: Launch production instance attached with ec2-s3 IAM role and launch developer instance with normal configuration.

Step 4: Now connect the production server and install code deploy.

Step 5: Now connect the developer machine and login the IAM user (Pro\_developer) by entering security credentials.

Step 6: Create directory to run the code.

Step 7: Create yaml script for deplopment in web server.

Step 8: Create folder for scripts to INSTALL, START, STOP.

Step 9: Create S3 bucket to store code and scripts

Step 10: Open console and open code deploy

- create application for that in developer instance

aws deploy create-application --application-name sampleapp

- to push deploy codes to S3 bucket in zip file

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

Step 11: Now mention the deployment server in code deploy service in console

* enter into code deploy
* open application sampleapp
* open create deployment group
* create deployment

-----------------------------------------------------------------------------------------------

**CodePipeline for version update**

* Open code pipeline in console.
* Create pipeline.
* Then edit the code in index.html.
* zip -r ../sampleapp.zip .
* aws s3 cp sampleapp.zip s3://cdcpproject.

**CHALLENGES FACED**

During the implementation of the project, I have made an error in source code YAML script which leads to unsuccessful deployment and I discovered that error in deployment events and I troubleshooted it.

**LESSONS LEARNED**

Understood the deployment lifecycle. Each stage, from code commit to production deployment, should be meticulously planned and tested. This includes considering pre-deployment checks, deployment strategies, and post-deployment validation. CodePipeline allows automate the entire release process, from source control to production deployment. By this project session, I had a view of skills learned and implement those and troubleshooting the errors happened while deployment.

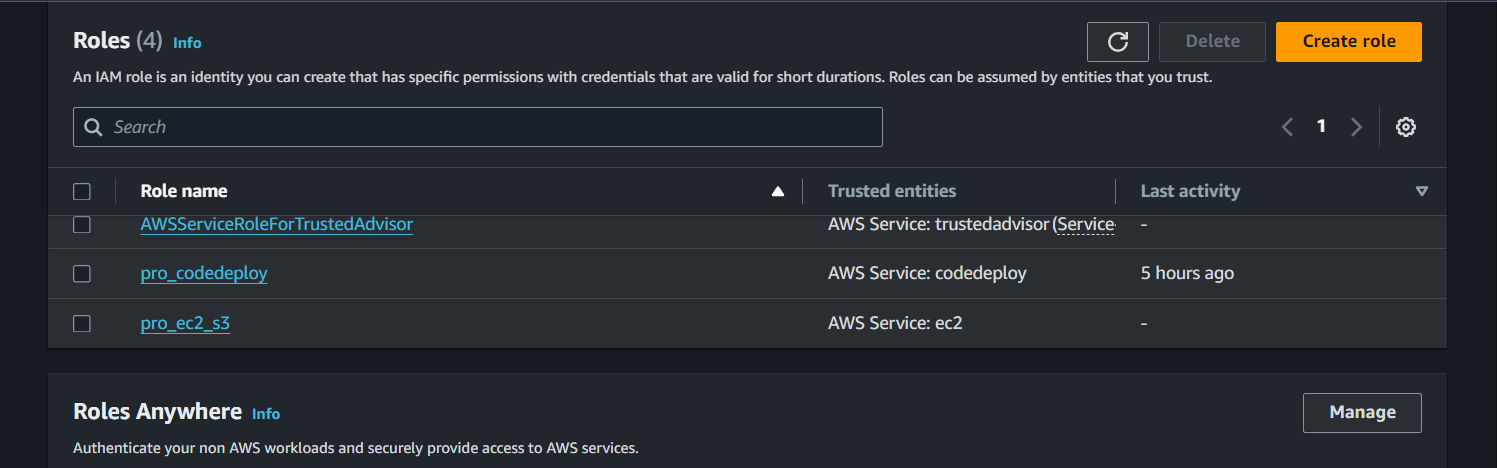
**CONCLUSION**

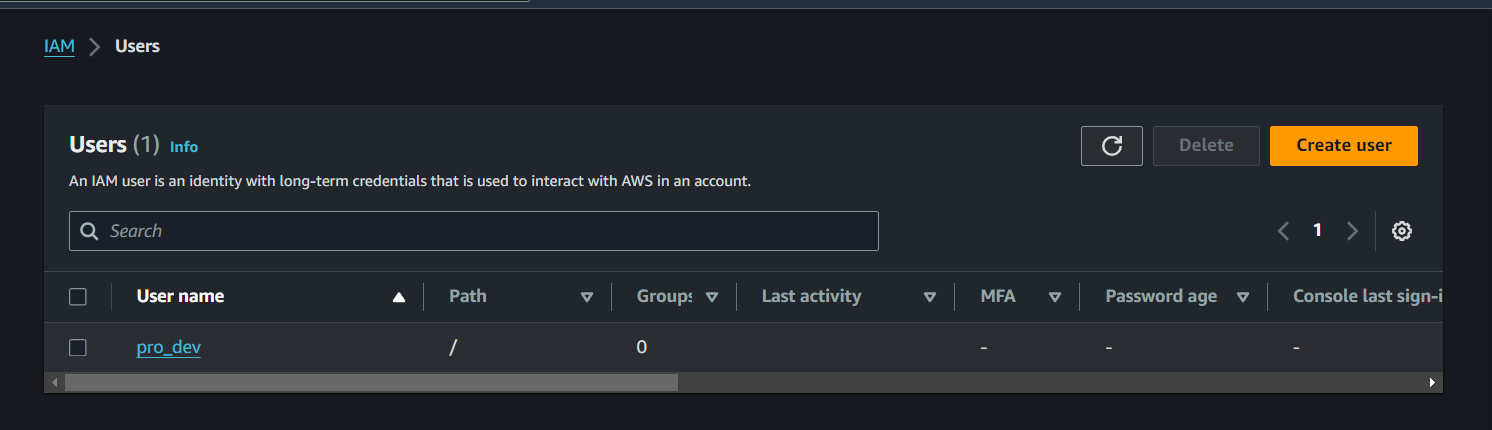
In conclusion, an AWS CodeDeploy and CodePipeline web hosting project offers a robust framework for automating and optimizing the deployment process. Throughout this project, several key takeaways have emerged, highlighting the importance of meticulous planning, automation, testing, monitoring, security, documentation, and continuous improvement.

By leveraging AWS CodeDeploy and CodePipeline, teams can effectively manage the deployment lifecycle, from code commit to production release, with efficiency and reliability. Automation plays a pivotal role in streamlining the release process, reducing manual intervention, and mitigating the risk of errors.

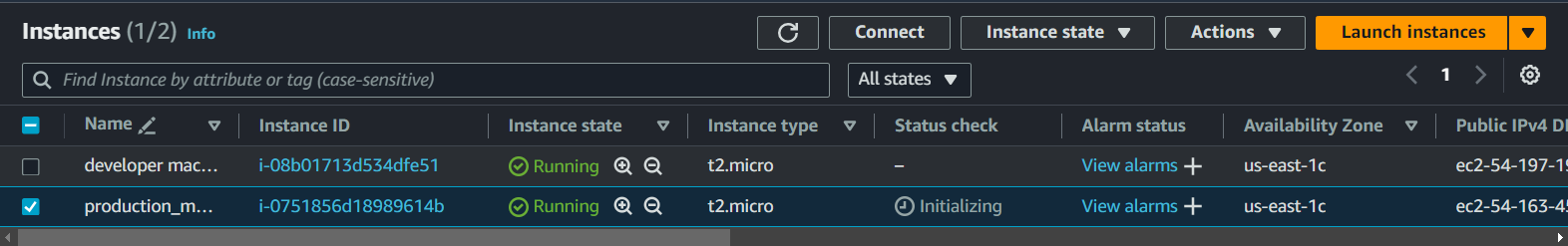
**SCREENSHOTS**

* Created IAM roles and Developer user.

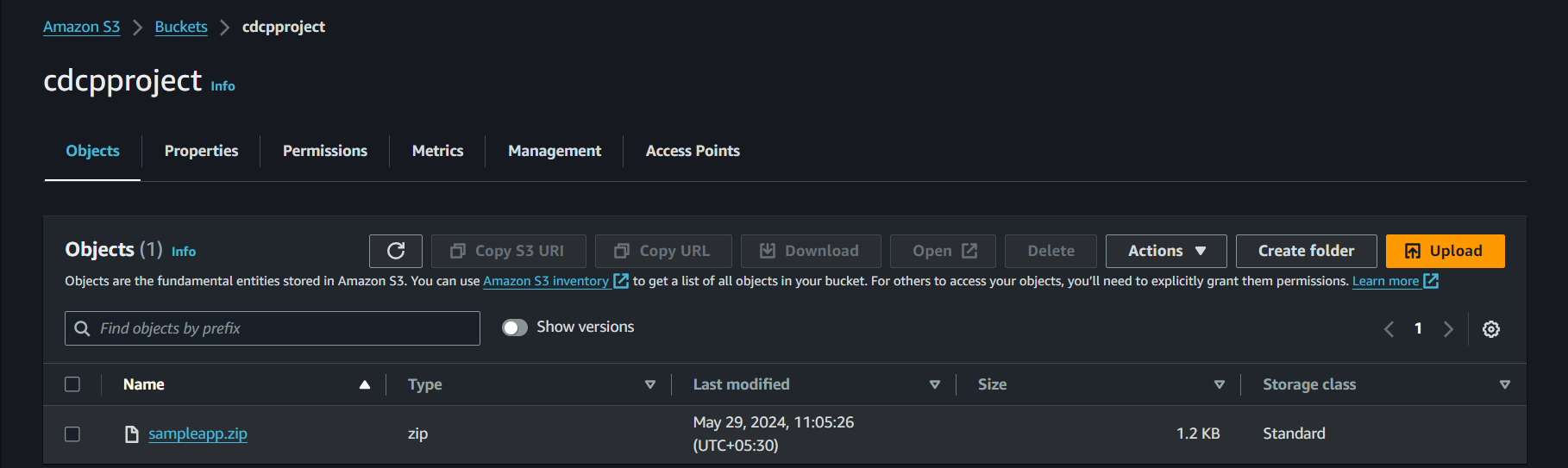
****

****

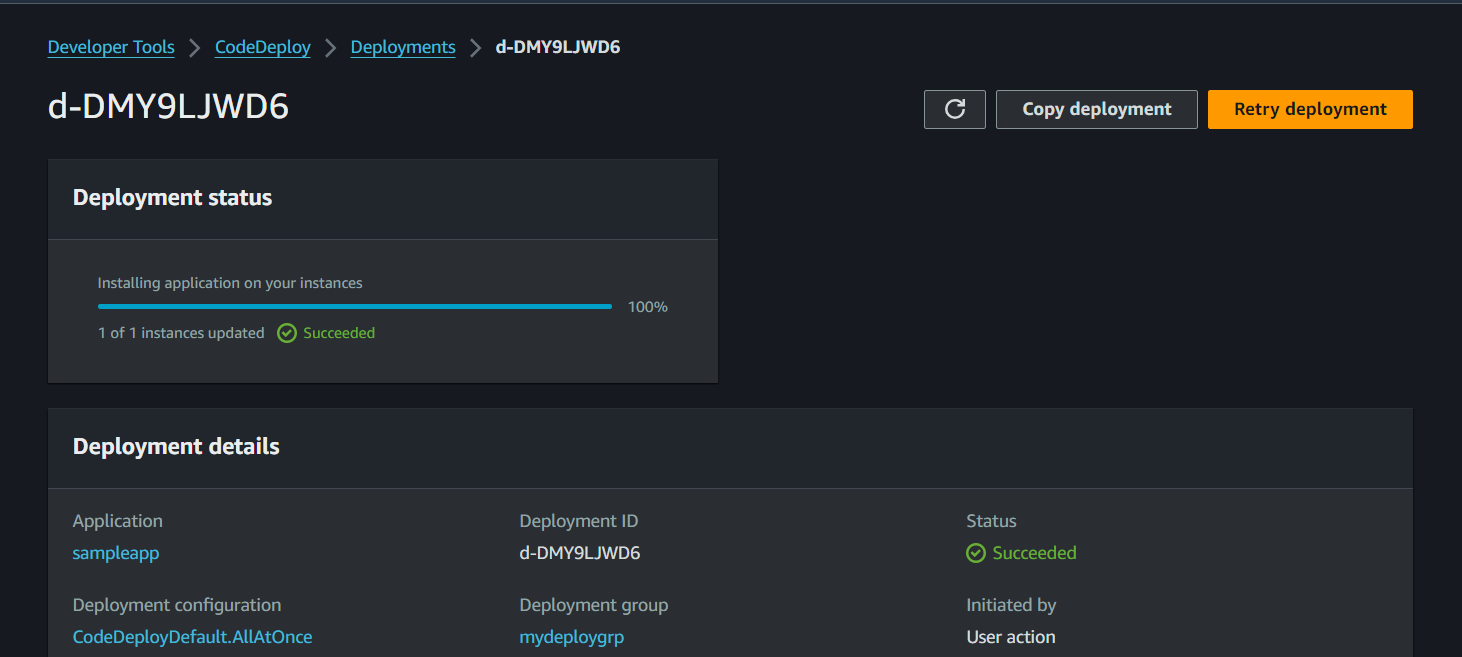
* Created EC2 developer instance and production instance.

****

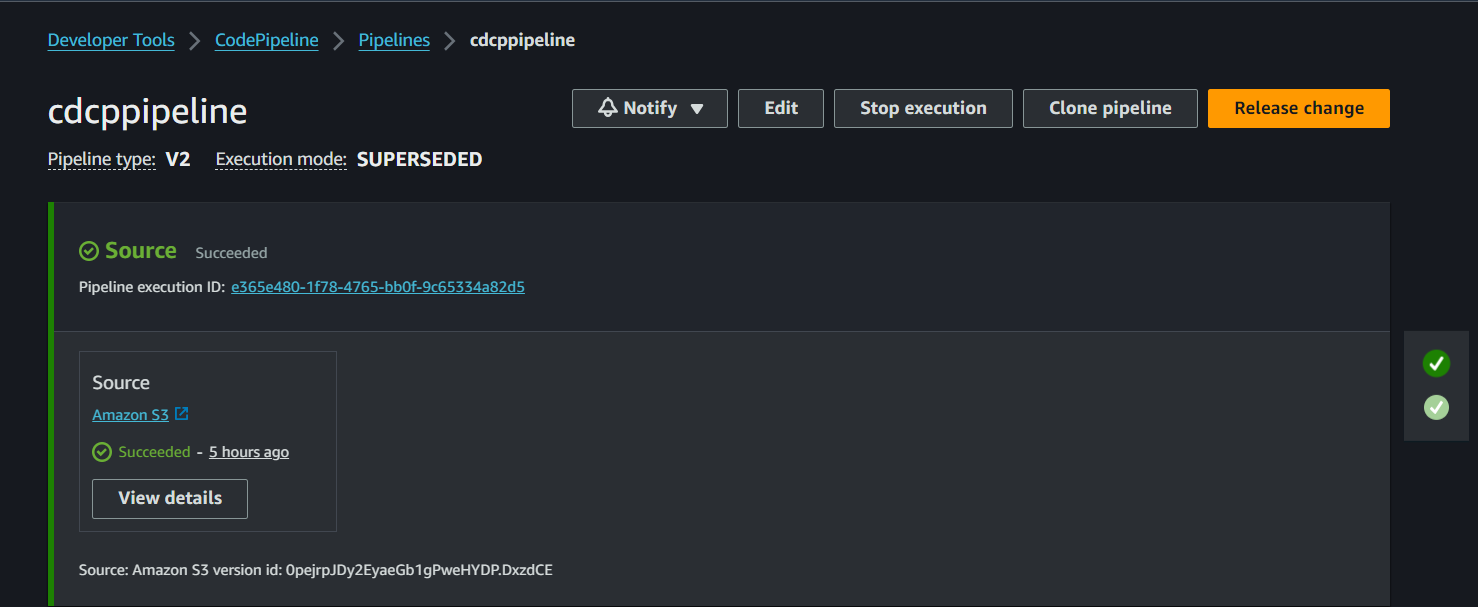
* Created S3 bucket and uploaded the objects for deployment (Source code, YAML scripts) as zip file.

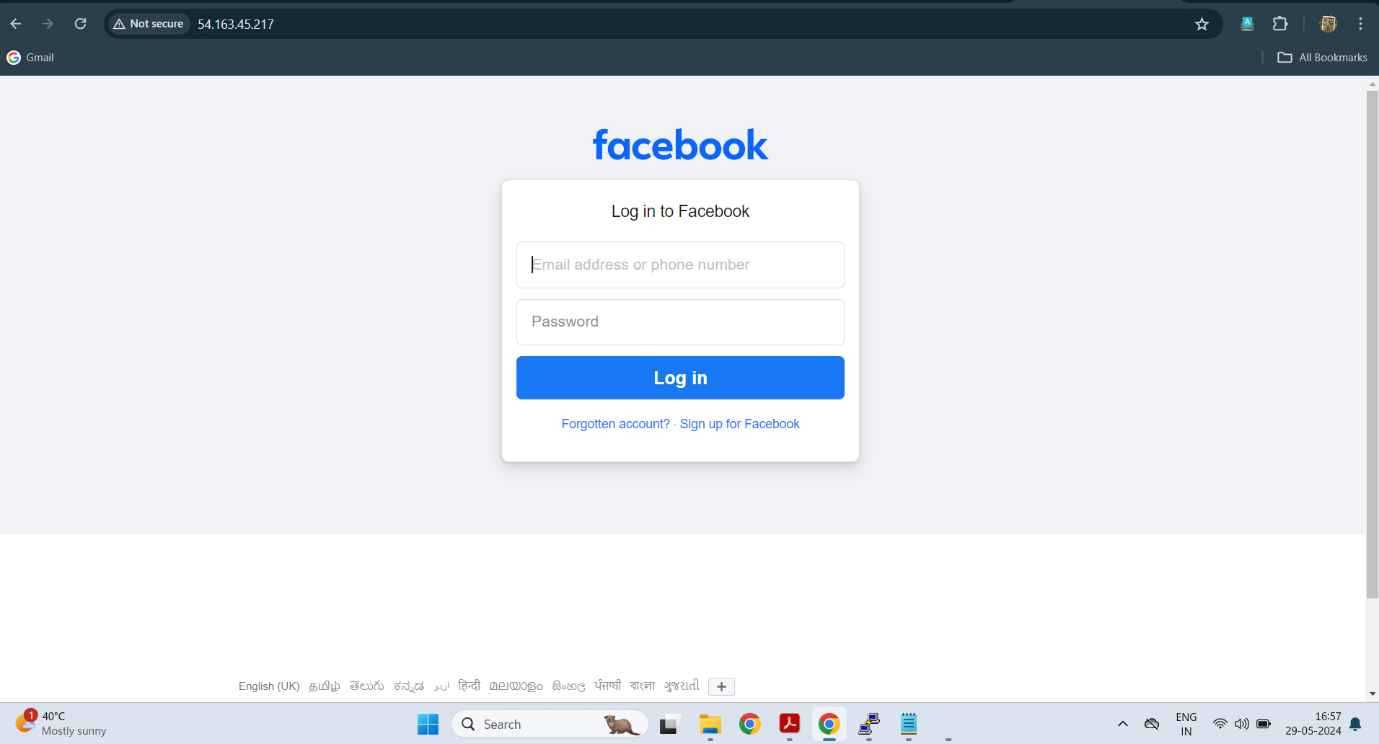
****

* Created Application and deployment in CodeDeploy and deployed the code.

****

* Created codePipeline and connected with S3 for versioning for the deployment further.

****

****