Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Cloud Deployment with Automation

Concepts Used:

- AWS CodePipeline
- EC2
- S3

Problem Statement:

Build a simple web application using AWS CodeBuild and deploy it to an S3 bucket. Then, automate the deployment process using AWS CodePipeline, ensuring the application is deployed on an EC2 instance. A sample index.html page will be used for demonstration.

Tasks:

1. **Set up AWS CodeBuild** for the web app.
2. **Create a pipeline** that deploys the web app to an S3 bucket.
3. **Use AWS CodeDeploy** to push updates to an EC2 instance.

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

1. Introduction

Case Study Overview

This case study focuses on the automation of cloud deployment using AWS services, specifically leveraging AWS CodePipeline, EC2, and S3. The primary objective is to build a simple yet effective web application while automating its deployment process to enhance efficiency and reliability. By integrating AWS CodeDeploy into this workflow, we aim to ensure seamless and efficient updates, allowing for quick rollouts of new features and bug fixes. The use of these AWS tools enables us to create a robust, scalable, and easily maintainable web application infrastructure that can adapt to the evolving needs of users and businesses. This approach not only streamlines development workflows but also promotes best practices in software deployment, fostering a culture of continuous improvement.

Key Feature and Application

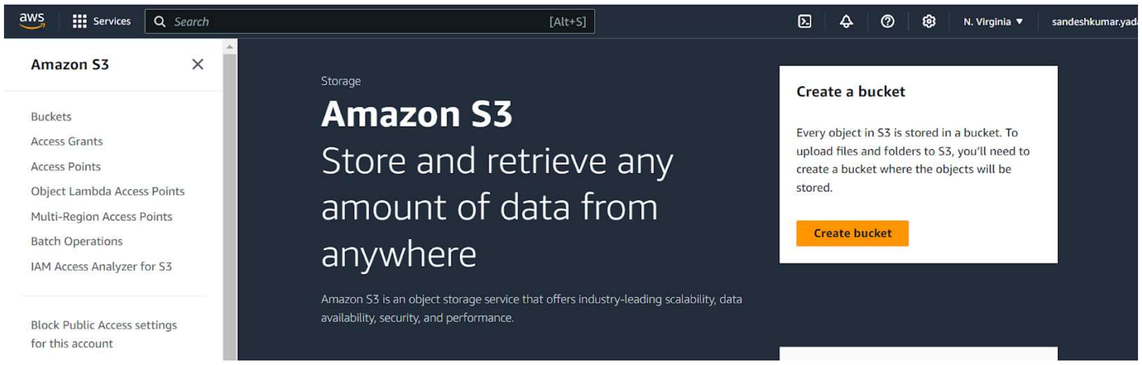
A standout feature of this case study is the automation of the deployment process through AWS CodePipeline. This powerful tool orchestrates the various stages of application delivery, ensuring that any modifications made to the web application are automatically built, tested, and deployed with minimal manual intervention. This automation significantly reduces the time and effort required for deployment, allowing development teams to focus on enhancing application features and addressing user feedback promptly. By utilising AWS services, organisations can effectively implement continuous integration and continuous deployment (CI/CD) practices. This leads to a more agile development environment, where updates can be delivered rapidly and reliably to end users, thus improving user satisfaction and engagement. Additionally, this automated deployment process minimises the risk of human error, ensuring that the application remains stable and performs optimally in a production setting.

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

2. Step-by-Step Explanation

Step 1: Create an S3 Bucket

1. Navigate to **S3** in the AWS Management Console.
2. Click **Create bucket** and provide a unique name (e.g., my-app-bucket).
3. Uncheck **Block all public access** for testing.
4. Enable versioning in the **Properties** tab by selecting "Enable" and saving changes.


aws
Services
Search
[Alt+S]
N. Virginia
sandeshkumar.yad
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Create bucket

Storage

Amazon S3

Store and retrieve any amount of data from anywhere

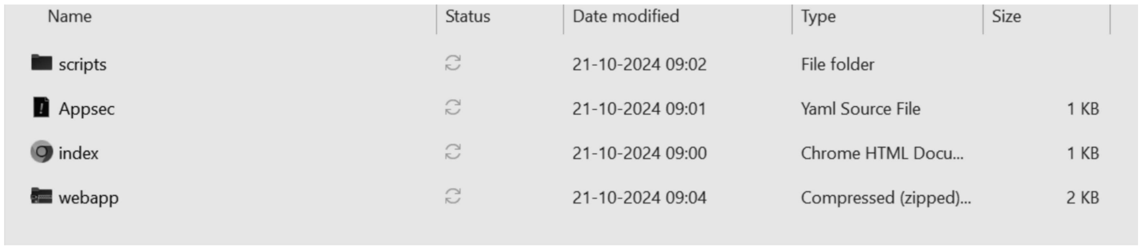
Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Create a bucket

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

Step 2: Create a Simple Web Application

1. Create a file named **index.html** on your local machine and add basic HTML code.
2. Create a configuration file named **Appsec.yml**.
3. Create a script **install_dependencies.sh** to update the system and install Nginx.
4. Create a script **start_nginx.sh**.
5. Zip all files into **webapp.zip** for easier upload.

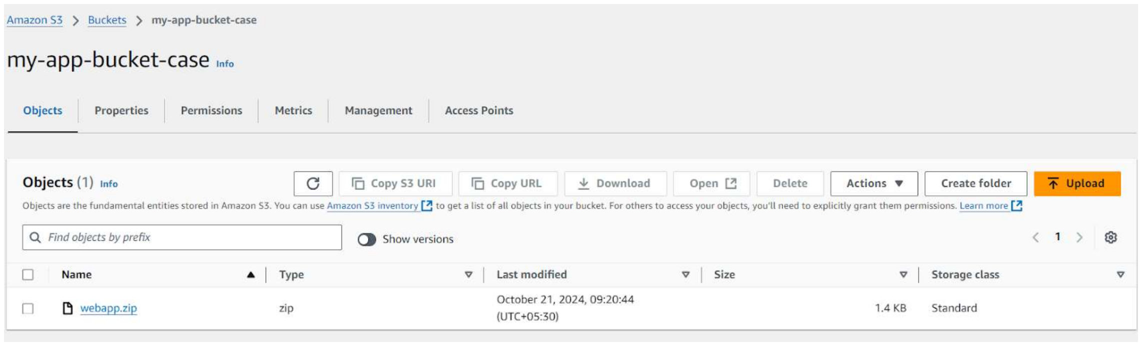


Name	Status	Date modified	Type	Size
scripts		21-10-2024 09:02	File folder	
Appsec		21-10-2024 09:01	Yaml Source File	1 KB
index		21-10-2024 09:00	Chrome HTML Docu...	1 KB
webapp		21-10-2024 09:04	Compressed (zipped)...	2 KB

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Step 3: Upload the Web App to S3

1. Go back to the S3 console and select your bucket.
2. Click **Upload**, add **webapp.zip**, and click **Upload**.



my-app-bucket-case

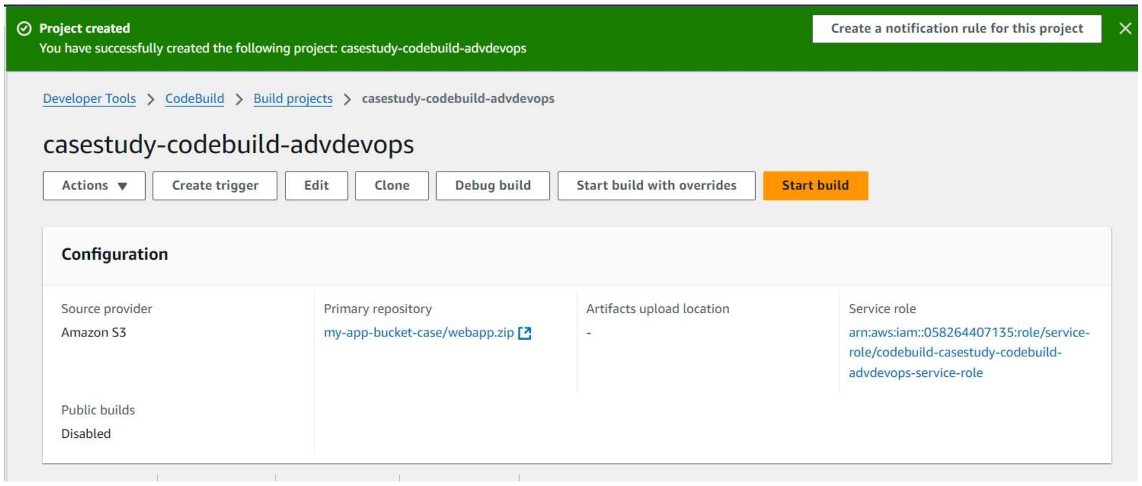
Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
webapp.zip	zip	October 21, 2024, 09:20:44 (UTC+05:30)	1.4 KB	Standard

Step 4: Set Up AWS CodeBuild

1. Click **Create build project**.
2. Configure the project with the following:
   - **Project Name**: Name your project.
   - **Source Provider**: Select Amazon S3.
   - **Bucket**: Select your bucket.
   - **Object Key**: Choose **webapp.zip**.
   - **Operating System**: Select Ubuntu.
   - **Buildspec**: Insert build commands (YAML script).
3. Uncheck uploading logs to CloudWatch.


Create a notification rule for this project
Developer Tools > CodeBuild > Build projects > casestudy-codebuild-advdevops
Actions
Create trigger
Edit
Clone
Debug build
Start build with overrides
Start build

Project created
You have successfully created the following project: casestudy-codebuild-advdevops

casestudy-codebuild-advdevops

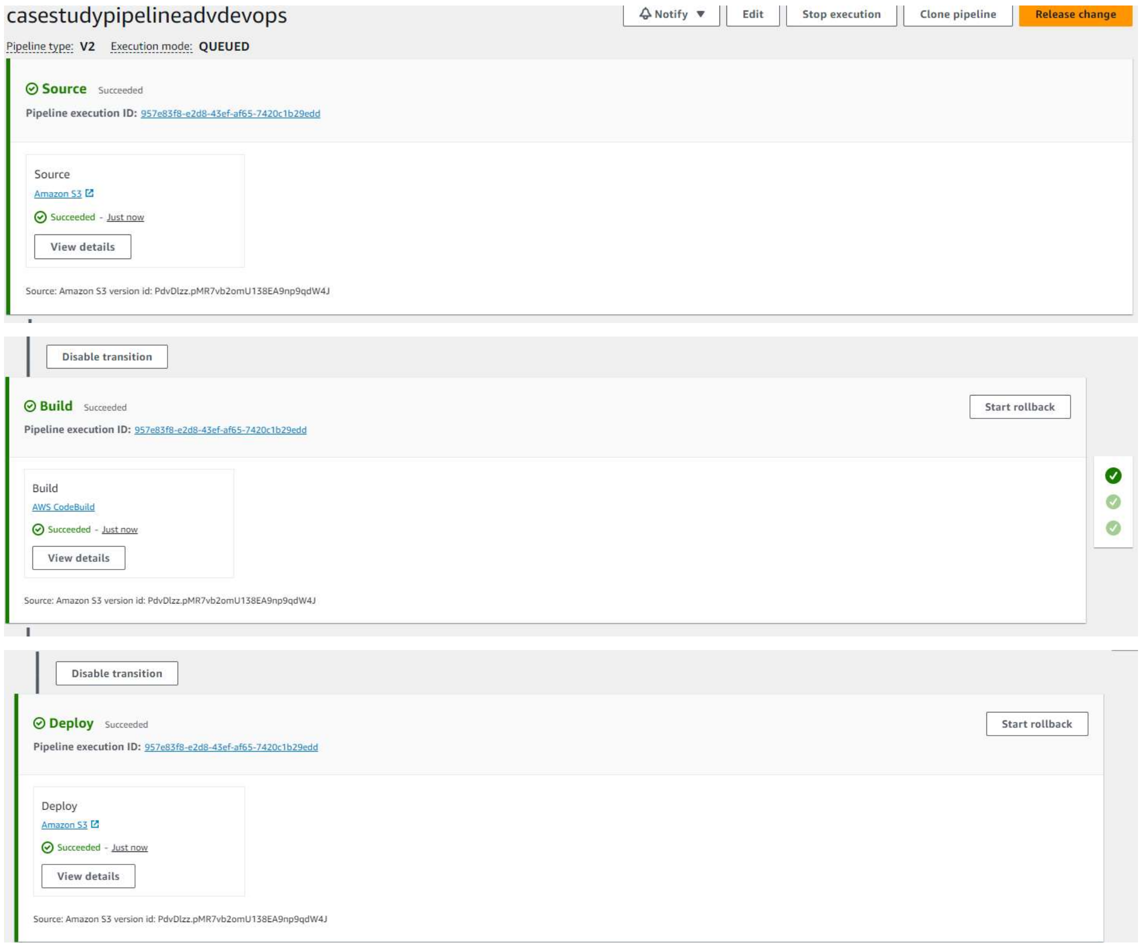
Configuration

Source provider: Amazon S3
Primary repository: my-app-bucket-case/webapp.zip
Artifacts upload location: -
Service role: arn:aws:iam::058264407135:role/service-role/codebuild-casestudy-codebuild-advdevops-service-role
Public builds: Disabled

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Step 5: Set Up AWS CodePipeline

1. In the AWS Management Console, search for **CodePipeline**.
2. Click **Create Pipeline** and configure:
   - **Pipeline Name**: Enter a name.
   - **Service Role**: Create a new role.
   - **Source Provider**: Select Amazon S3.
   - **Bucket**: Choose the same S3 bucket.
   - **S3 object key**: Enter **webapp.zip**.
   - **Build Provider**: Select AWS CodeBuild.
   - **Deploy Provider**: Select Amazon S3, enabling extraction before deployment.



casestudypipelineadvdevops

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded
Pipeline execution ID: 957e83f8-e2d8-43ef-af65-7420c1b29edd

Source
Amazon S3
Succeeded - Just now

Source: Amazon S3 version id: PdvDlzz.pMR7vb2omU138EA9np9qdW4J

Build Succeeded
Pipeline execution ID: 957e83f8-e2d8-43ef-af65-7420c1b29edd

Build
AWS CodeBuild
Succeeded - Just now

Source: Amazon S3 version id: PdvDlzz.pMR7vb2omU138EA9np9qdW4J

Deploy Succeeded
Pipeline execution ID: 957e83f8-e2d8-43ef-af65-7420c1b29edd

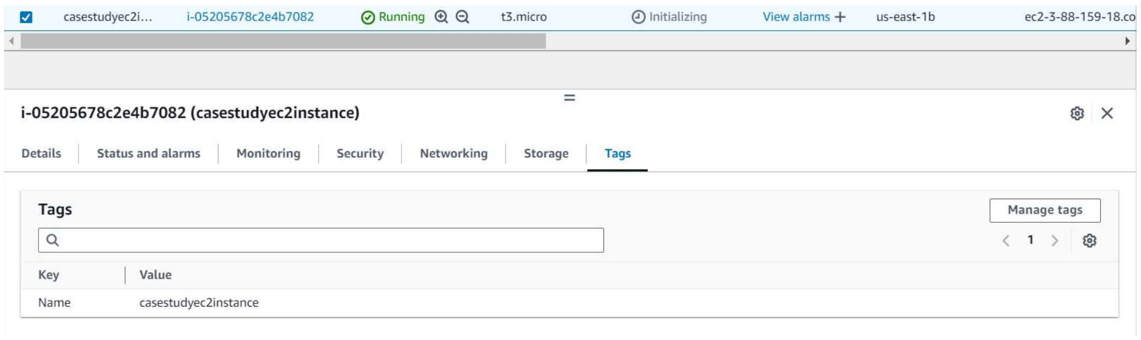
Deploy
Amazon S3
Succeeded - Just now

Source: Amazon S3 version id: PdvDlzz.pMR7vb2omU138EA9np9qdW4J

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Step 6: Create an EC2 Instance

1. Click **Launch Instance** and choose an instance type (e.g., t3.micro).
2. Create a new RSA key pair for SSH access.
3. Allow HTTP/HTTPS traffic from anywhere.



casestudyec2i... i-05205678c2e4b7082 Running t3.micro Initializing View alarms us-east-1b ec2-3-88-159-18.co

i-05205678c2e4b7082 (casestudyec2instance)

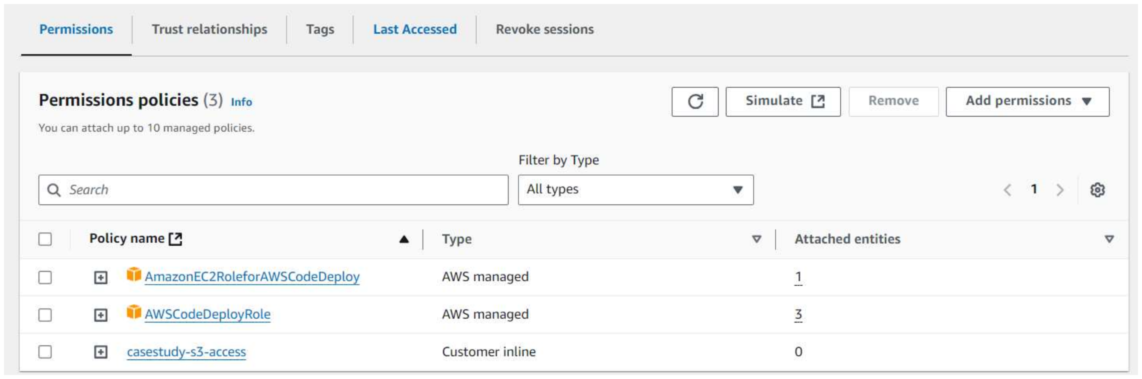
Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Tags

Key	Value
Name	casestudyec2instance

Step 7: Create an IAM Role for EC2 with CodeDeploy

1. Go to **IAM**, click **Roles**, and create a new role for EC2.
2. Attach **AWSCodeDeployRole** and **AmazonEC2RoleforAWSCodeDeploy** policies.
3. Edit the trust policy to allow CodeDeploy to assume the role.
4. Add an inline policy allowing access to S3 objects.



Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (3) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonEC2RoleforAWSCodeDeploy	AWS managed	1
AWSCodeDeployRole	AWS managed	3
casestudy-s3-access	Customer inline	0

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Step 8: Install CodeDeploy Agent and Nginx on EC2

1. SSH into your EC2 instance and run the following commands

   sudo yum install -y ruby

   cd /tmp

   wget https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/latest/install

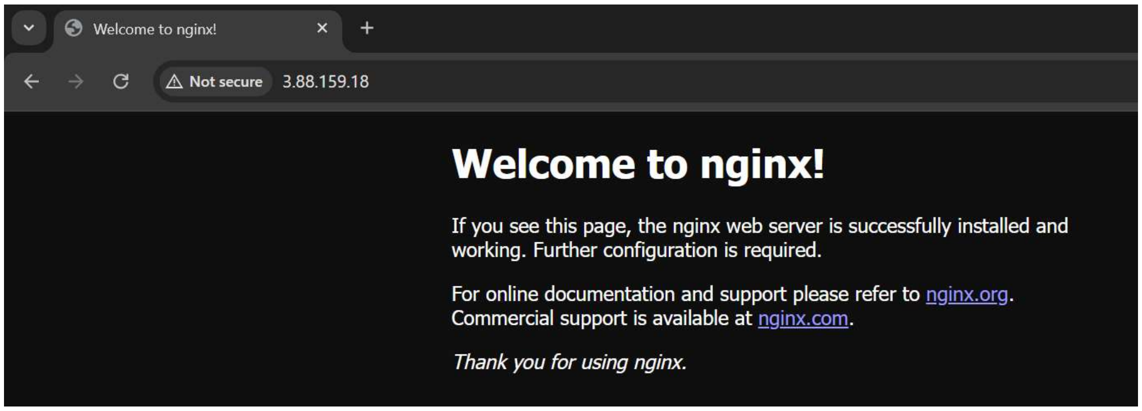
   chmod +x ./install

   sudo ./install auto

   sudo service codedeploy-agent start

   sudo yum install -y nginx

   sudo service nginx start



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

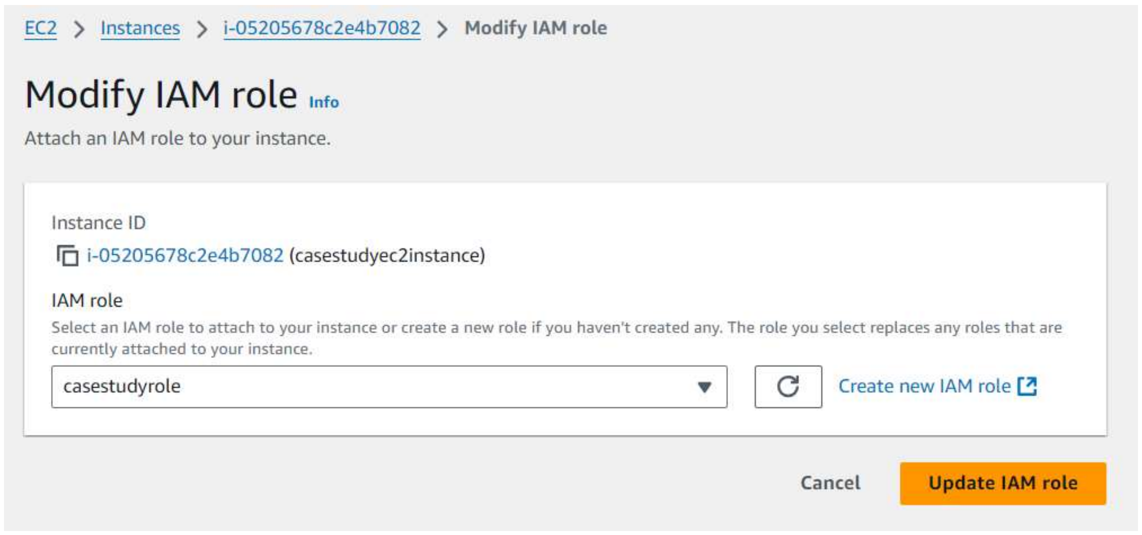
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Step 9: Attach the IAM Role to the EC2 Instance

1. In the EC2 Dashboard, select your instance.
2. Click **Actions > Security > Modify IAM Role** and attach the created IAM role.


EC2 > Instances > i-05205678c2e4b7082 > Modify IAM role

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

i-05205678c2e4b7082 (casestudyec2instance)

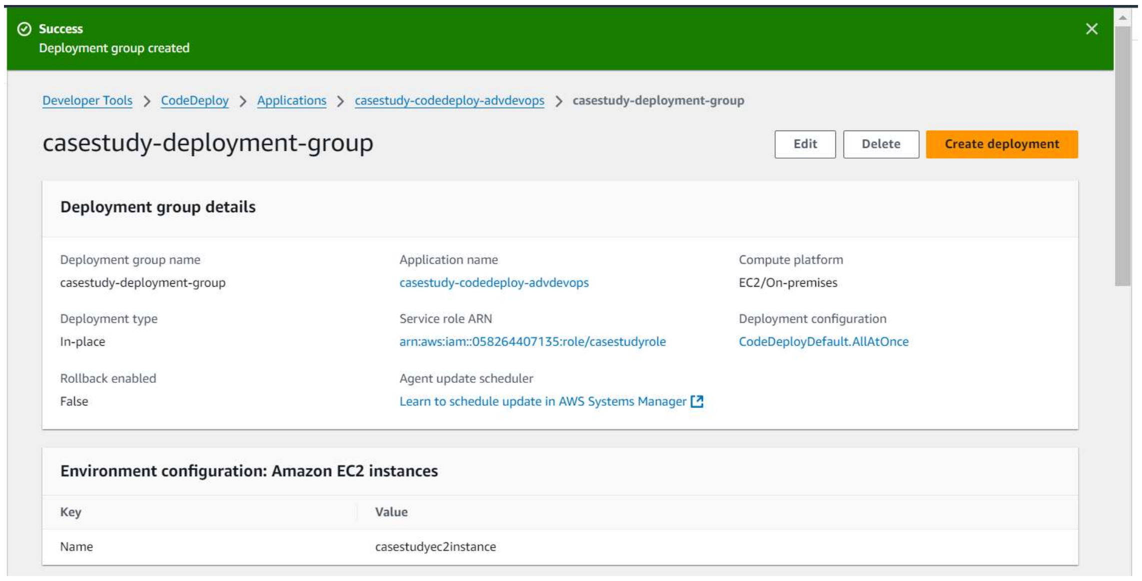
IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

casestudyrole

Step 10: Set Up AWS CodeDeploy

1. Open **CodeDeploy Console** and create a new application.
2. Create a deployment group, entering the ARN of the service role created for CodeDeploy.
3. For environment configuration, select EC2 instances and enter the tag key-value pair.


Success
Deployment group created

Developer Tools > CodeDeploy > Applications > casestudy-codedeploy-advdevops > casestudy-deployment-group

casestudy-deployment-group

Deployment group details

Deployment group name	Application name	Compute platform
casestudy-deployment-group	casestudy-codedeploy-advdevops	EC2/On-premises
Deployment type	Service role ARN	Deployment configuration
In-place	arn:aws:iam::058264407135:role/casestudyrole	CodeDeployDefault.AllAtOnce
Rollback enabled	Agent update scheduler	
False	Learn to schedule update in AWS Systems Manager	

Environment configuration: Amazon EC2 instances

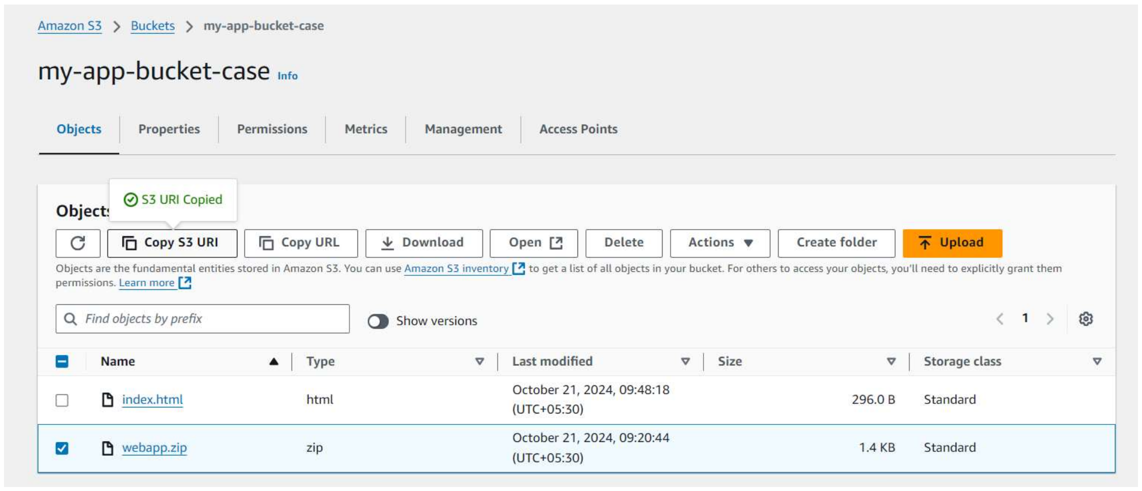
Key	Value
Name	casestudyec2instance

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Step 11: Give S3 Access to CodeDeploy

1. Navigate to your S3 bucket and go to the **Permissions** tab.
2. Ensure the bucket policy allows CodeDeploy access

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
    "Effect": "Allow",
    "Principal": {
      "Service": "codedeploy.amazonaws.com"
    },
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::my-app-bucket/*"
   }
 ]
}
```



Amazon S3 > Buckets > my-app-bucket-case

my-app-bucket-case Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects — S3 URI Copied

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

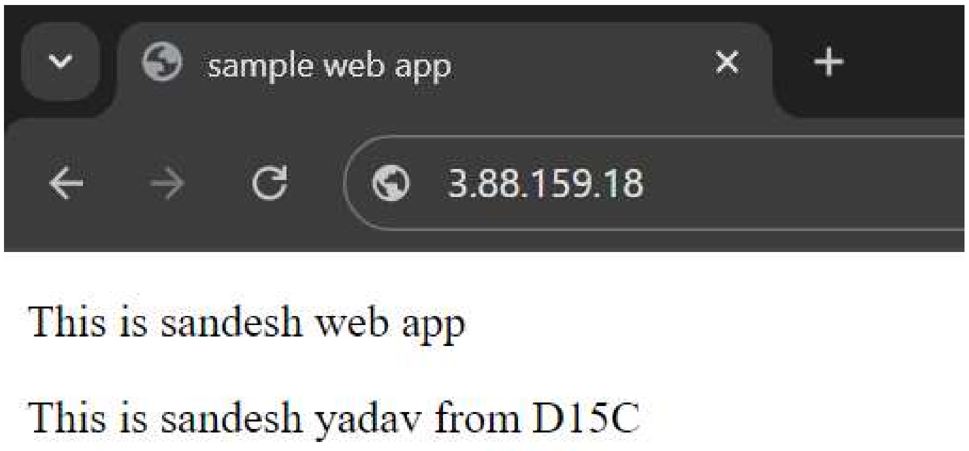
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Find objects by prefix — Show versions — 1

	Name	Type	Last modified	Size	Storage class
☐	index.html	html	October 21, 2024, 09:48:18 (UTC+05:30)	296.0 B	Standard
☑	webapp.zip	zip	October 21, 2024, 09:20:44 (UTC+05:30)	1.4 KB	Standard

Step 12: Create a Deployment for Your Application

1. In the **Deployments** tab, click **Create deployment**.
2. Check if your application is accessible using the public IPv4 address of your EC2 instance.



This is sandesh web app

This is sandesh yadav from D15C

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Guidelines and Best Practices

1. **Ensure Your S3 Bucket Names Are Unique**:
   - **Global Uniqueness**: S3 bucket names must be unique across all AWS accounts. Always check for existing names when creating a bucket.
   - **Descriptive Names**: Use clear naming conventions that indicate the bucket's purpose (e.g.,my-casestudy-app-bucket).
   - **Versioning**: Consider including a timestamp or version number in the bucket name to track changes.
2. **Regularly Monitor Your Deployments and Logs in AWS**:
   - **Utilise CloudWatch**: Monitor resource utilisation and application performance in real-time. Set up dashboards and alerts for critical thresholds.
   - **Review Logs**: Regularly check AWS CodeDeploy and CodePipeline logs to identify issues and improve deployment processes.
   - **Automated Alerts**: Configure notifications for critical issues to maintain application reliability.
3. **Implement Appropriate Security Measures**:
   - **Limit Public Access**: Use S3 bucket policies to restrict public access and allow access only to specific users or IPs.
   - **Least Privilege IAM Roles**: Grant permissions only necessary for tasks, ensuring EC2 instances have limited access to resources.
   - **Resource-Based Policies**: Utilise these policies to control access to S3 buckets based on specific conditions.
   - **Regular IAM Audits**: Periodically review and remove unused IAM roles and policies.
   - **Data Encryption**: Enable encryption for data stored in S3 and use AWS Key Management Service (KMS) for managing keys.

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

3. Demonstration Preparation

1. **Key Points**:
    - **Overview of AWS Services**: Briefly explain AWS CodePipeline, CodeBuild, CodeDeploy, S3, and EC2, and how they integrate to automate deployment.
    - **Step-by-Step Process**: Highlight each step of the deployment process:
        - Creating an S3 bucket and enabling versioning.
        - Building the web application and preparing the deployment package.
        - Setting up AWS CodeBuild and CodePipeline.
        - Launching an EC2 instance and configuring it for deployment.
        - Establishing IAM roles and permissions.
        - Ensuring successful deployment through AWS CodeDeploy.
    - **Security Practices**: Emphasise the importance of security measures, such as S3 bucket policies and IAM role restrictions.
    - **Monitoring and Maintenance**: Discuss the need for regular monitoring using CloudWatch and reviewing logs for issues.
2. **Practice**:
    - **Rehearse the Demonstration**: Practise presenting the material to ensure clarity and confidence. Familiarise yourself with each step to speak naturally and fluently.
    - **Time Management**: Aim to stay within the allocated time for the demonstration by timing each section during practice.
    - **Technical Familiarity**: Ensure you are comfortable with navigating the AWS console and executing commands to handle any live demonstrations or troubleshooting during the presentation.
3. **Questions**:
    - **Anticipate Common Questions**:

What are the benefits of using AWS for deployment?

- AWS offers scalability, reliability, and seamless integration of various services, making deployment efficient and flexible.

How do you handle errors during deployment?

- Monitoring logs and utilising automated alerts are crucial for identifying and resolving errors promptly.

What security measures are critical for this deployment?

- Key security measures include implementing IAM role restrictions, defining bucket policies, and employing encryption practices.

Can you explain the difference between CodeDeploy and CodePipeline?

- CodeDeploy focuses on deploying applications, while CodePipeline manages the overall deployment workflow from source to production.

Name : Sandeshkumar.M.Yadav **Div : D15C** **Roll : 61**

Conclusion

In this case study, we examined the automation of cloud deployment using AWS services, particularly AWS CodePipeline, EC2, and S3, by building and deploying a simple web application. We demonstrated how these tools work together to create a scalable and efficient deployment process, enabling continuous integration and deployment through AWS CodeBuild and CodeDeploy. Emphasising security, we highlighted the importance of unique S3 bucket naming, IAM role restrictions, and effective monitoring practices to maintain application integrity and performance. Ultimately, leveraging AWS's ecosystem streamlines development processes, enhances productivity, and allows organisations to deliver high-quality applications swiftly, ensuring competitiveness in the evolving digital landscape.