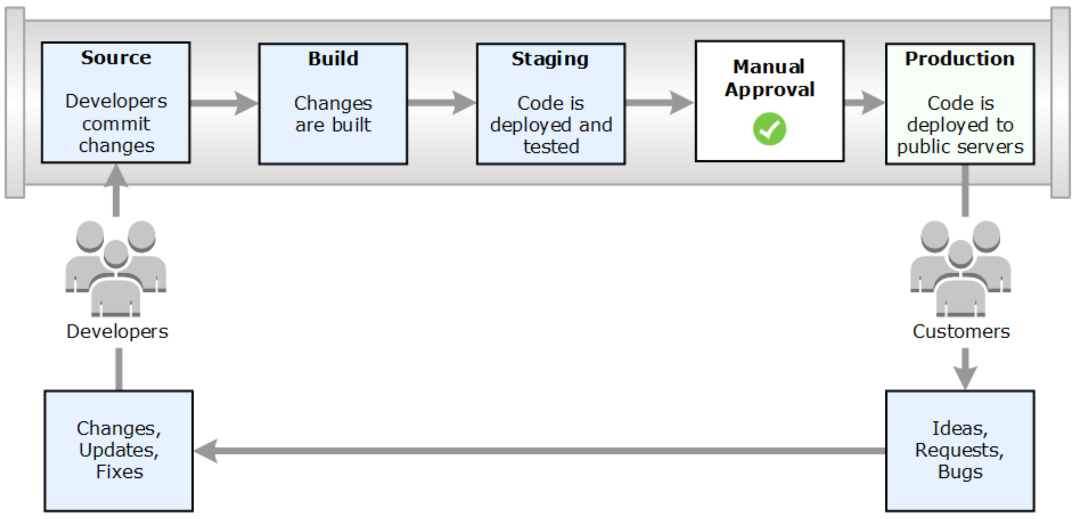
CI/CD Pipeline with AWS CodePipeline & CodeDeploy

Goal: Automatically deploy a simple web application to EC2 with versioned deployments and rollback support.



Architecture Overview

Developer
↓
Source (S3)
↓
CodePipeline
↓
CodeDeploy
↓
EC2 Instances

- **S3** → Stores application versions
- **CodePipeline** → Orchestrates CI/CD
- **CodeDeploy** → Deploys & rolls back
- **EC2** → Hosts web app (Apache)

Prerequisites

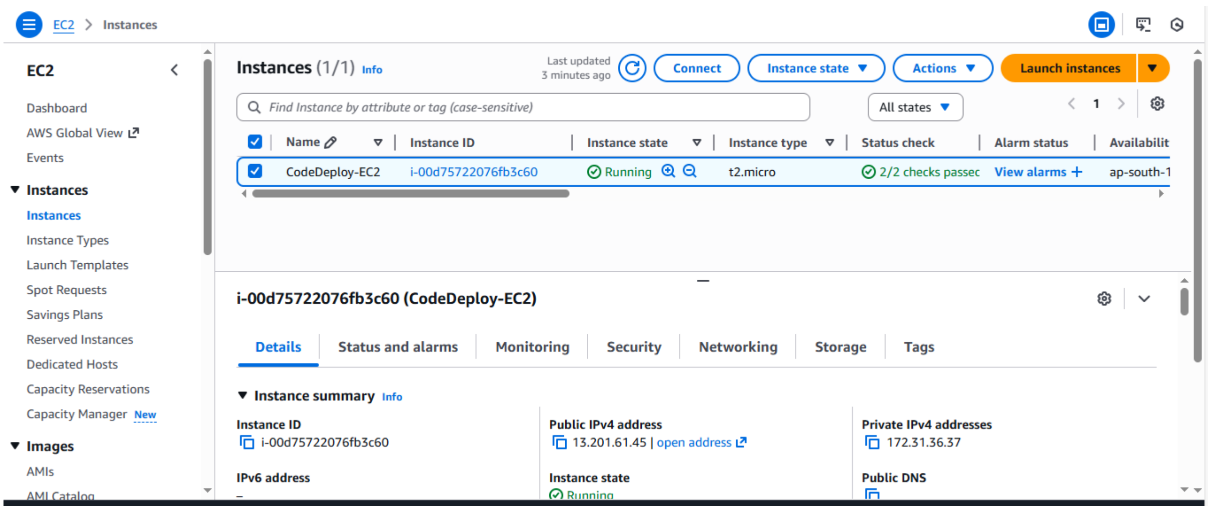
Before starting, ensure:

- AWS account (Free Tier works)
- IAM permissions (Admin or DevOps)
- EC2 key pair created
- Region selected (example: ap-south-1)

STEP 1: Create an EC2 Instance

1.1 Launch EC2

1. Go to **AWS Console → EC2**
2. Click **Launch instance**
3. Name: CodeDeploy-EC2
4. AMI: **Amazon Linux 2**
5. Instance type: t2.micro
6. Key pair: select existing
7. Network settings:
   - Allow **HTTP (80)**
   - Allow **SSH (22)**
8. Click **Launch instance**



Instances (1/1) Info — Last updated 3 minutes ago

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availabilit
CodeDeploy-EC2	i-00d75722076fb3c60	Running	t2.micro	2/2 checks passec	View alarms +	ap-south-1

i-00d75722076fb3c60 (CodeDeploy-EC2)

Instance summary Info

Instance ID: i-00d75722076fb3c60
Public IPv4 address: 13.201.61.45 | open address
Private IPv4 addresses: 172.31.36.37
IPv6 address: –
Instance state: Running
Public DNS

1.2 Install Apache & CodeDeploy Agent

Connect to EC2 → **EC2 Instance Connect**

Run:

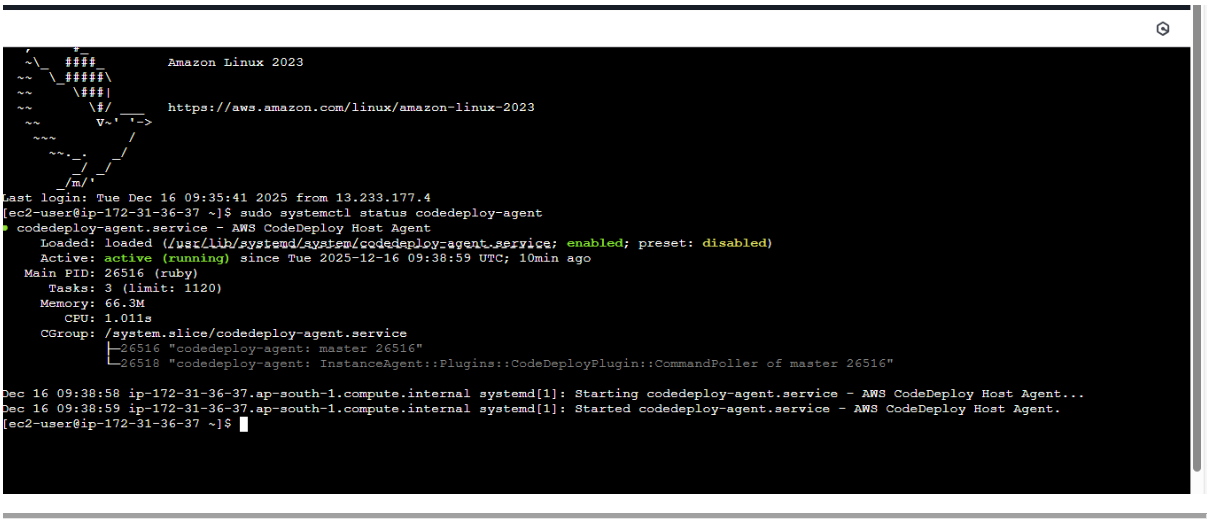
```
sudo yum update -y
sudo yum install -y ruby wget
sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
```

Install CodeDeploy Agent:

```
cd /home/ec2-user
wget https://aws-codedeploy-ap-south-1.s3.ap-south-1.amazonaws.com/latest/install
chmod +x install
sudo ./install auto
sudo systemctl start codedeploy-agent
```

Verify:

```
sudo systemctl status codedeploy-agent
```



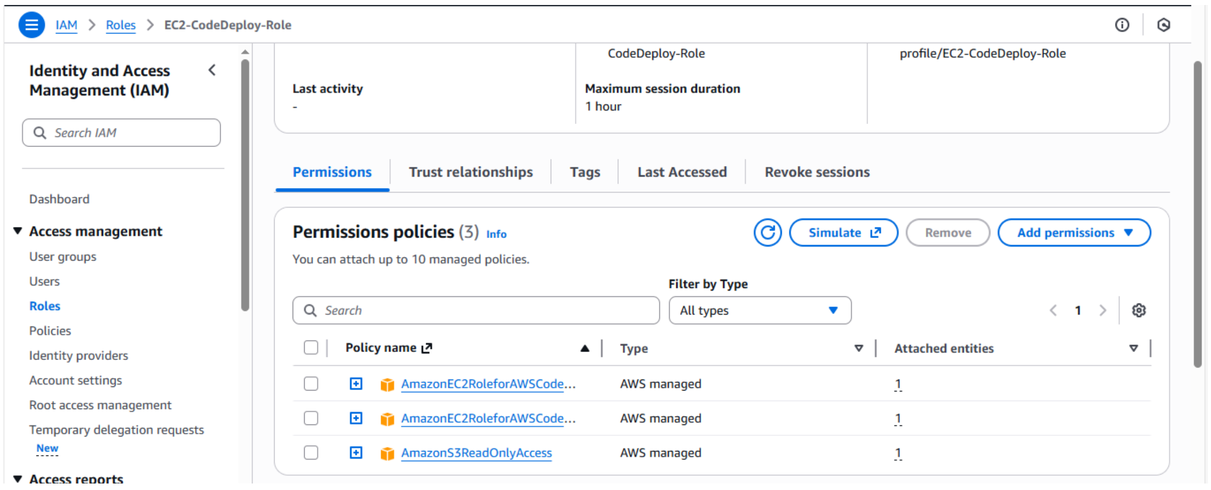
```
  ,     #_
  ~\_  ####_        Amazon Linux 2023
 ~~  \_#####\
 ~~     \###|
 ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
  ~~       V~' '->
   ~~~         /
     ~~._.   _/
        _/ _/
      _/m/'
Last login: Tue Dec 16 09:35:41 2025 from 13.233.177.4
[ec2-user@ip-172-31-36-37 ~]$ sudo systemctl status codedeploy-agent
● codedeploy-agent.service - AWS CodeDeploy Host Agent
     Loaded: loaded (/usr/lib/systemd/system/codedeploy-agent.service; enabled; preset: disabled)
     Active: active (running) since Tue 2025-12-16 09:38:59 UTC; 10min ago
   Main PID: 26516 (ruby)
      Tasks: 3 (limit: 1120)
     Memory: 66.3M
        CPU: 1.011s
     CGroup: /system.slice/codedeploy-agent.service
             ├─26516 "codedeploy-agent: master 26516"
             └─26518 "codedeploy-agent: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller of master 26516"

Dec 16 09:38:58 ip-172-31-36-37.ap-south-1.compute.internal systemd[1]: Starting codedeploy-agent.service - AWS CodeDeploy Host Agent...
Dec 16 09:38:59 ip-172-31-36-37.ap-south-1.compute.internal systemd[1]: Started codedeploy-agent.service - AWS CodeDeploy Host Agent.
[ec2-user@ip-172-31-36-37 ~]$
```

STEP 2: Create IAM Roles

2.1 EC2 Role (CodeDeploy Access)

1. Go to **IAM → Roles → Create role**
2. Trusted entity: **AWS service**
3. Service: **EC2**
4. Permissions:
   - AmazonS3ReadOnlyAccess
   - AWSCodeDeployFullAccess
5. Role name: EC2-CodeDeploy-Role
6. Create role

➡ Attach this role to your EC2 instance
EC2 → Instance → Actions → Security → Modify IAM Role



IAM > Roles > EC2-CodeDeploy-Role

CodeDeploy-Role | profile/EC2-CodeDeploy-Role

Last activity: -
Maximum session duration: 1 hour

Permissions policies (3) Info

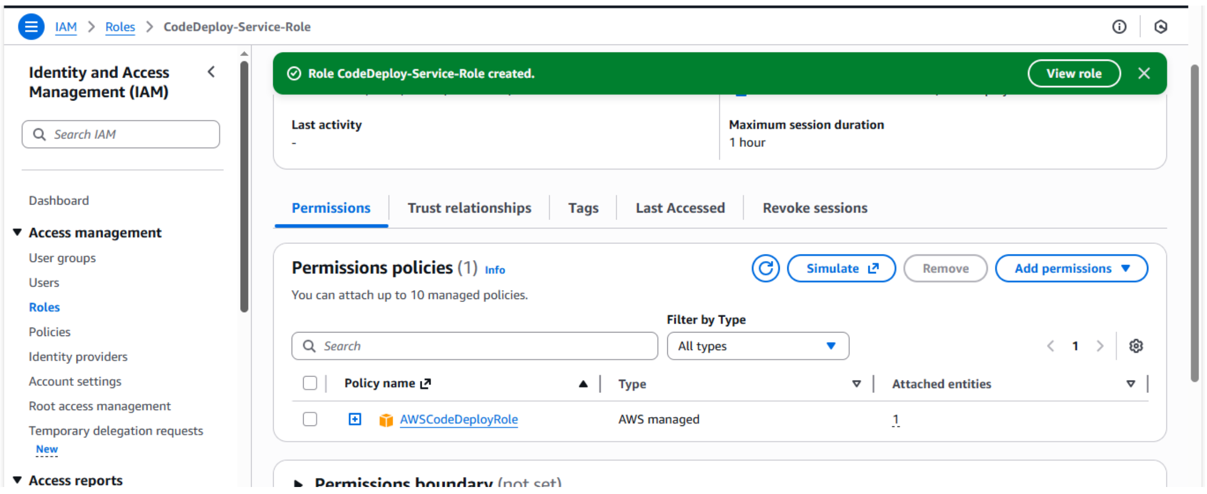
You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonEC2RoleforAWSCode...	AWS managed	1
AmazonEC2RoleforAWSCode...	AWS managed	1
AmazonS3ReadOnlyAccess	AWS managed	1

2.2 CodeDeploy Service Role

1. IAM → Roles → Create role
2. Service: **CodeDeploy**
3. Use case: **CodeDeploy**

4. Attach policy: AWSCodeDeployRole
5. Role name: CodeDeploy-Service-Role


IAM > Roles > CodeDeploy-Service-Role

Identity and Access Management (IAM)

Search IAM

Dashboard
- Access management
  - User groups
  - Users
  - Roles
  - Policies
  - Identity providers
  - Account settings
  - Root access management
  - Temporary delegation requests New
- Access reports

Role CodeDeploy-Service-Role created. View role

Last activity
-

Maximum session duration
1 hour

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

Permissions policies (1) Info

Simulate | Remove | Add permissions

You can attach up to 10 managed policies.

Filter by Type: All types

Search

Policy name	Type	Attached entities
AWSCodeDeployRole	AWS managed	1

Permissions boundary (not set)

STEP 3: Prepare Application Files

3.1 Create App Files (Local System)

```
myapp/
├── index.html
├── appspec.yml
└── scripts/
    ├── install.sh
    └── start.sh
```

index.html

```
<h1>CI/CD Deployment Successful □</h1>
```

appspec.yml

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/
hooks:
  AfterInstall:
    - location: scripts/install.sh
      timeout: 300
  ApplicationStart:
    - location: scripts/start.sh
      timeout: 300
```

scripts/install.sh

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

scripts/start.sh

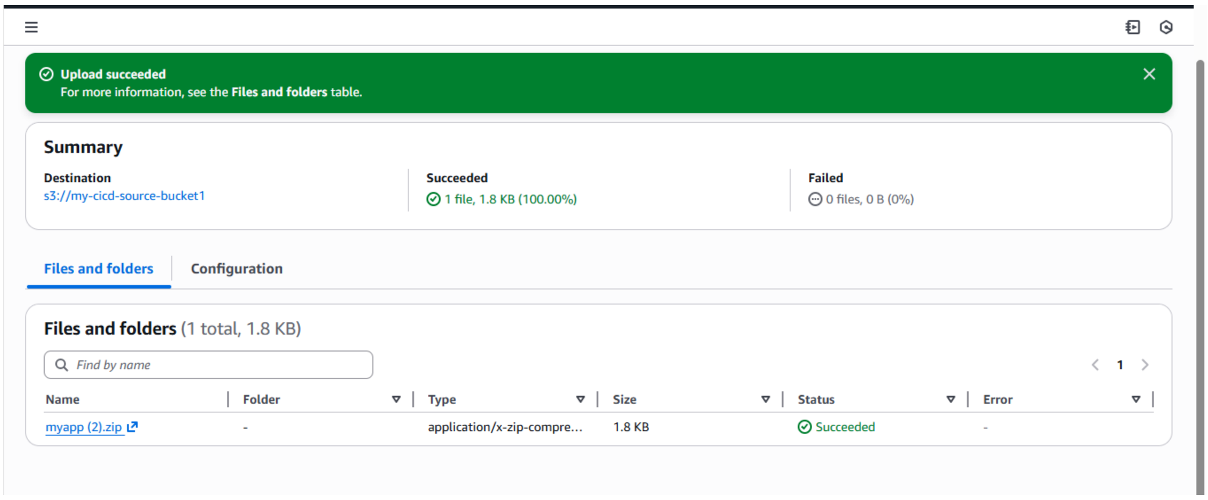
```
#!/bin/bash
sudo systemctl start httpd
```

Zip the folder:

myapp.zip

STEP 4: Create S3 Bucket (Source Stage)

1. Go to **S3** → **Create bucket**
2. Bucket name: my-cicd-source-bucket
3. Region: same as EC2
4. Keep defaults → Create bucket
5. Upload **myapp.zip**



Upload succeeded
For more information, see the **Files and folders** table.

Summary

Destination	Succeeded	Failed
s3://my-cicd-source-bucket1	1 file, 1.8 KB (100.00%)	0 files, 0 B (0%)

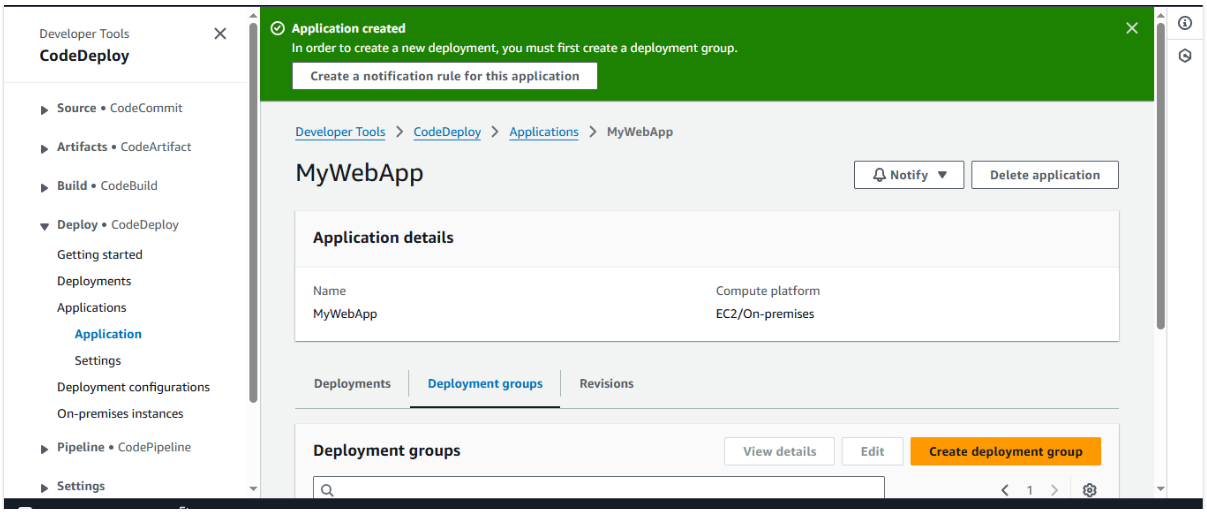
Files and folders (1 total, 1.8 KB)

Name	Folder	Type	Size	Status	Error
myapp (2).zip	-	application/x-zip-compre...	1.8 KB	Succeeded	-

---

STEP 5: Create CodeDeploy Application

1. Go to **CodeDeploy**
2. Applications → **Create application**
3. Name: MyWebApp
4. Compute platform: **EC2/On-Premises**
5. Create application



Application created

In order to create a new deployment, you must first create a deployment group.

Create a notification rule for this application

Developer Tools > CodeDeploy > Applications > MyWebApp

MyWebApp

Notify ▼ | Delete application

Application details

Name	Compute platform
MyWebApp	EC2/On-premises

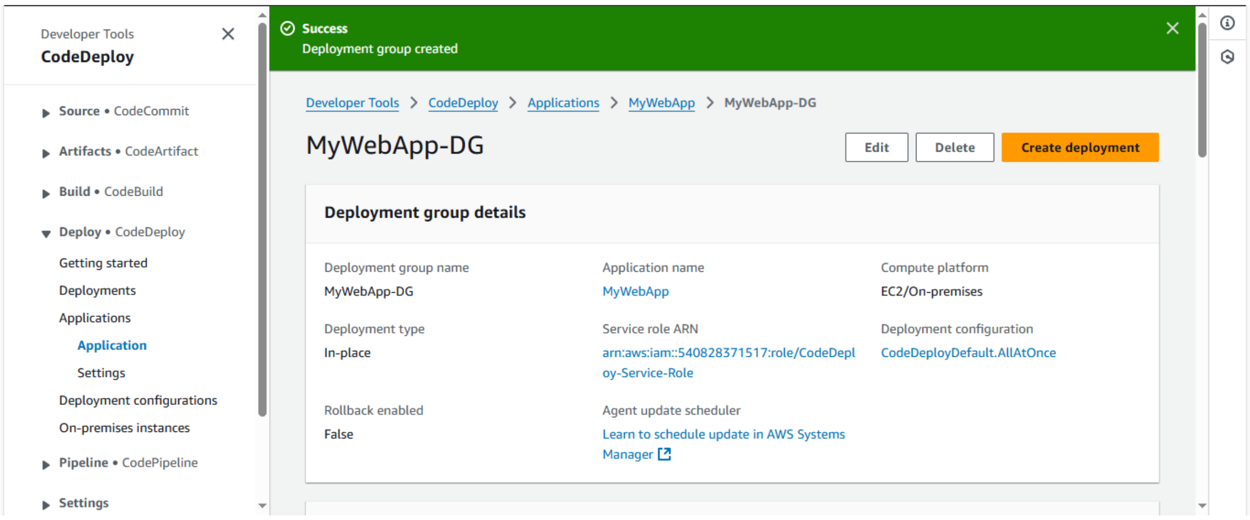
Deployments | Deployment groups | Revisions

Deployment groups

View details | Edit | Create deployment group

5.1 Create Deployment Group

1. Inside app → **Create deployment group**
2. Name: MyWebApp-DG
3. Service role: CodeDeploy-Service-Role
4. Deployment type:
   - In-place
   - With rollback enabled ✅
5. Environment:
   - Amazon EC2 instances
6. Tag EC2:
   - Key: Name
   - Value: CodeDeploy-EC2
7. Deployment settings:
   - CodeDeployDefault.AllAtOnce
8. Disable Load Balancer
9. Create deployment group



Success

Deployment group created

Developer Tools > CodeDeploy > Applications > MyWebApp > MyWebApp-DG

MyWebApp-DG

Edit | Delete | Create deployment

Deployment group details

Deployment group name	Application name	Compute platform
MyWebApp-DG	MyWebApp	EC2/On-premises
Deployment type	**Service role ARN**	**Deployment configuration**
In-place	arn:aws:iam::540828371517:role/CodeDeploy-Service-Role	CodeDeployDefault.AllAtOnce
Rollback enabled	**Agent update scheduler**	
False	Learn to schedule update in AWS Systems Manager	

STEP 6: Create CodePipeline

6.1 Start Pipeline Creation

1. Go to **CodePipeline**
2. Click **Create pipeline**
3. Pipeline name: MyWebApp-Pipeline
4. Service role: New role
5. Artifact store: Default (S3)
6. Next

6.2 Source Stage

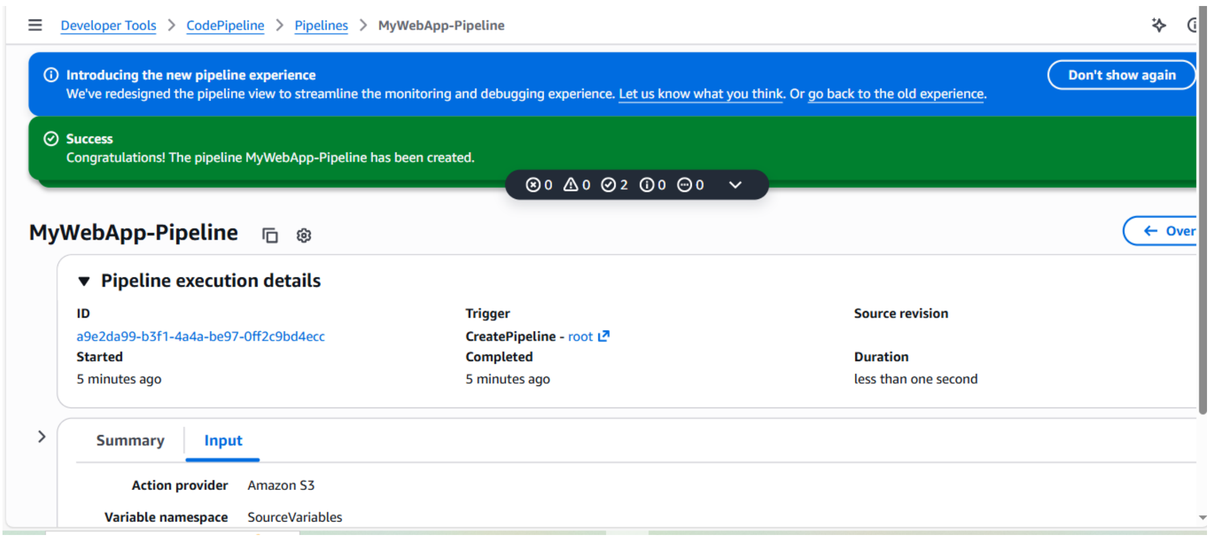
1. Source provider: **Amazon S3**
2. Bucket: my-cicd-source-bucket
3. Object key: myapp.zip
4. Change detection: Enabled
5. Next

6.3 Build Stage

➡ **Skip build stage**
(Static HTML app)

6.4 Deploy Stage

1. Deploy provider: **AWS CodeDeploy**
2. Application name: MyWebApp
3. Deployment group: MyWebApp-DG
4. Next
5. Create pipeline


Developer Tools > CodePipeline > Pipelines > MyWebApp-Pipeline
Don't show again
0 0 2 0 0
Over
Summary
Input

Introducing the new pipeline experience
We've redesigned the pipeline view to streamline the monitoring and debugging experience. Let us know what you think. Or go back to the old experience.

Success
Congratulations! The pipeline MyWebApp-Pipeline has been created.

MyWebApp-Pipeline

Pipeline execution details

ID	Trigger	Source revision
a9e2da99-b3f1-4a4a-be97-0ff2c9bd4ecc	CreatePipeline - root	
Started	Completed	Duration
5 minutes ago	5 minutes ago	less than one second

Action provider	Amazon S3
Variable namespace	SourceVariables

STEP 7: Test Deployment

1. Pipeline automatically starts
2. Wait for **Deploy → Success**
3. Copy EC2 **Public IPv4 address**
4. Open browser:

http://13.201.61.45
✅Output:
CI/CD Deployment Successful

STEP 8: Test Versioning & Rollback

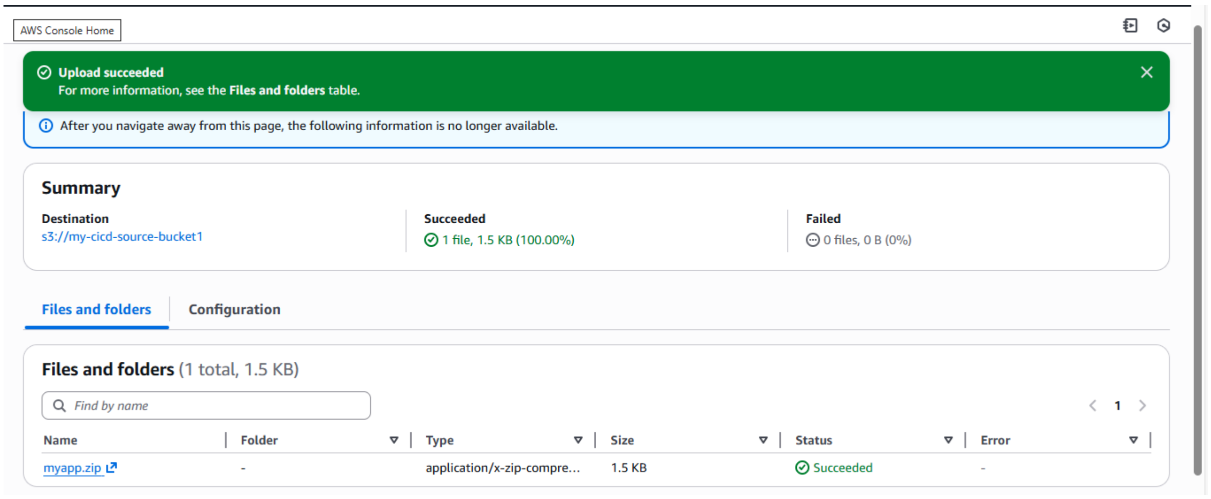
8.1 Update Code

Edit index.html:

<h1>Version 2 Deployed Successfully □</h1>

Zip again → upload to S3 (same object name)

➡ Pipeline auto-triggers



Upload succeeded
For more information, see the **Files and folders** table.

After you navigate away from this page, the following information is no longer available.

Summary

Destination	Succeeded	Failed
s3://my-cicd-source-bucket1	1 file, 1.5 KB (100.00%)	0 files, 0 B (0%)

Files and folders (1 total, 1.5 KB)

Name	Folder	Type	Size	Status	Error
myapp.zip	-	application/x-zip-compre...	1.5 KB	Succeeded	-

8.2 Rollback

1. Go to **CodeDeploy → Deployments**
2. Select failed or previous deployment
3. Click **Rollback deployment**

Project Summary

In this project, a **CI/CD pipeline** was successfully implemented using **AWS CodePipeline, CodeDeploy, EC2, and S3** to automate the deployment of a simple web application. The pipeline continuously monitors an **Amazon S3 bucket** for code changes and automatically deploys updated application versions to an **EC2 instance** using **AWS CodeDeploy**.

The EC2 instance was configured with the **CodeDeploy agent** and an **IAM role** to securely access AWS services. Application versions were packaged and stored in S3, while **CodePipeline** orchestrated the end-to-end deployment process. **CodeDeploy deployment groups** ensured controlled, in-place deployments with rollback capability in case of failures.

This setup demonstrates core **DevOps principles** such as automation, version control, continuous delivery, and operational reliability.

Conclusion

The CI/CD pipeline effectively eliminates manual deployment steps by enabling **automated, consistent, and repeatable deployments** on AWS. By integrating CodePipeline with CodeDeploy and EC2, the system ensures faster application updates, reduced human error, and improved deployment reliability.

The inclusion of **versioned deployments and rollback support** enhances application stability and minimizes downtime during failures. This project provides a strong foundation for real-world DevOps workflows and can be easily extended by integrating **GitHub as a source**, adding **build stages with CodeBuild**, or deploying to **Auto Scaling groups and Load Balancers**.

Overall, this project validates practical skills in **AWS DevOps, continuous integration, and continuous deployment**, making it highly relevant for cloud engineering roles, academic projects, and real-world production environments.