# Step 3: Create AWS CodeDeploy Resources

This document explains all the detailed steps required to create AWS CodeDeploy resources to deploy a Flask application on EC2 instances.

## A. Create the CodeDeploy Service Role (IAM)

This IAM role allows CodeDeploy to act in your account.  
  
Steps:  
1. Go to IAM → Roles → Create role.  
2. Trusted entity: AWS service → Use case: CodeDeploy.  
3. Permissions: Attach the AWS managed policy AWSCodeDeployRole.  
4. Name the role (e.g., CodeDeployServiceRole) and create it.  
  
Behind the scenes, the trust policy includes codedeploy.amazonaws.com as the principal.

## B. Create the CodeDeploy Application

Steps:  
1. In the CodeDeploy console, go to Applications → Create application.  
2. Application name: FlaskDemoApp (or your choice).  
3. Compute platform: EC2/On-premises.  
4. Create application.

## C. Create the Deployment Group

Steps:  
1. Inside your application (FlaskDemoApp), choose Create deployment group.  
2. Deployment group name: FlaskDG.  
3. Service role: Select the CodeDeployServiceRole created earlier.  
4. Deployment type: In-place.  
5. Environment configuration: Amazon EC2 instances.  
 - Use EC2 tag filters.  
 - Key: CodeDeploy, Value: FlaskDemo (or your chosen tags).  
6. Deployment settings:  
 - Deployment configuration: CodeDeployDefault.OneAtATime.  
7. Load balancer: Leave unchecked (optional, add later if using ALB).  
8. Alarms and Rollback: Optional, leave disabled for simple setup.  
9. Create deployment group.

## D. Run a Deployment

Steps:  
1. Upload your deployment ZIP file (flask-codedeploy-demo.zip) to an S3 bucket.  
2. In the CodeDeploy console, go to Deployments → Create deployment.  
3. Select Application: FlaskDemoApp.  
4. Select Deployment group: FlaskDG.  
5. Revision type: My application is stored in Amazon S3.  
6. Bucket: YOUR-BUCKET, Key: flask-codedeploy-demo.zip.  
7. Use appspec.yml from the bundle.  
8. Create deployment.  
  
The deployment lifecycle will run: BeforeInstall → AfterInstall → ApplicationStart → ValidateService.

## E. CLI Equivalent (Optional)

You can create the application, deployment group, and deployment via AWS CLI:  
  
aws deploy create-application \  
 --application-name FlaskDemoApp \  
 --compute-platform Server  
  
aws deploy create-deployment-group \  
 --application-name FlaskDemoApp \  
 --deployment-group-name FlaskDG \  
 --deployment-config-name CodeDeployDefault.OneAtATime \  
 --ec2-tag-filters Key=CodeDeploy,Value=FlaskDemo,Type=KEY\_AND\_VALUE \  
 --service-role-arn arn:aws:iam::<ACCOUNT\_ID>:role/CodeDeployServiceRole  
  
aws deploy create-deployment \  
 --application-name FlaskDemoApp \  
 --deployment-group-name FlaskDG \  
 --s3-location bucket=YOUR-BUCKET,key=flask-codedeploy-demo.zip,bundleType=zip

## F. Common Checks & Gotchas

- Ensure the CodeDeploy agent is running on EC2 (systemctl status codedeploy-agent).  
- Verify IAM instance profile on EC2 includes AmazonS3ReadOnlyAccess.  
- Make sure security groups allow the correct ports (8000 or 80).  
- Ensure appspec.yml is at the root of your deployment ZIP.  
- Verify the user in systemd unit matches your instance's default user (ec2-user).