

**Name:** Pranav Chaudhari

**Task:** Create a complete sample app CI/CD Pipeline with EC2, CodeBuild, CodeDeploy, Deployment Group, and CodePipeline using CloudFormation script.

**Github Repo:** <https://github.com/techdecipher/using-aws-dev-tools>

**Briefings:** AWS has its native IAC solution that is CloudFormation by which we can provision and manage infrastructure framework using code / script.

**Step 1) Unified IAM Role with Necessary Permissions:** Write an IAM role with the necessary permissions for EC2 instance to allow AWS services to interact with each other. Added assume roles and its policy for codebuild, ec2, codedeploy and pipeline to consider this role to do their task.

```
UnifiedCICDRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: 'Allow'
          Principal:
            Service:
              - 'ec2.amazonaws.com'
              - 'codebuild.amazonaws.com'
              - 'codedeploy.amazonaws.com'
              - 'codepipeline.amazonaws.com'
          Action: 'sts:AssumeRole'
    Policies:
      - PolicyName: 'UnifiedCICDPolicy'
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: 'Allow'
              Action:
                - 'ec2:Describe*'
                - 'ec2:AuthorizeSecurityGroupIngress'
                - 'ec2:AuthorizeSecurityGroupEgress'
                - 'ec2:RevokeSecurityGroupIngress'
                - 'ec2:RevokeSecurityGroupEgress'
                - 'ec2:CreateSecurityGroup'
                - 'ec2>DeleteSecurityGroup'
                - 'ec2:CreateTags'
                - 'ec2>DeleteTags'
                - 'codebuild:*'
                - 'codedeploy:*'
                - 'codepipeline:*'
                - 's3:GetObject'
                - 's3:PutObject'
                - 's3:ListBucket'
                - 'logs:*'
                - 'iam:PassRole'
              Resource: '*'
```

**Step 2) Security Group for EC2:** Write Security Group for EC2.

```
InstanceSecurityGroup:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupDescription: 'Enable SSH and HTTP access'
    VpcId: 'vpc-08feca1900f25be53'
    SecurityGroupIngress:
      - IpProtocol: 'tcp'
        FromPort: '22'
        ToPort: '22'
        CidrIp: '0.0.0.0/0'
      - IpProtocol: 'tcp'
        FromPort: '80'
        ToPort: '80'
        CidrIp: '0.0.0.0/0'
    SecurityGroupEgress:
      - IpProtocol: '-1'
        FromPort: '-1'
        ToPort: '-1'
        CidrIp: '0.0.0.0/0'
```

**Step 3) Instance Profile for EC2:** Instance Profile is created to associate the UnifiedCICDRole with the respective EC2 instance.

```
UnifiedInstanceProfile:
  Type: 'AWS::IAM::InstanceProfile'
  Properties:
    Roles:
      - !Ref UnifiedCICDRole
```

**Step 4) EC2 Instance:** Write script for EC2 instance with httpd preinstalled as soon as it is launched. And CodeDeploy agent too, just so it can facilitate the deployment of application. The dummy index.html is there so it can pass the health check because it was giving me an error for it, so I just added it there which will be replace later once we are good to deploy it in ec2.

```
MyEC2Instance:
  Type: 'AWS::EC2::Instance'
  Properties:
    InstanceType: 't2.micro'
    KeyName: 'sample-app-using-ec2'
    ImageId: 'ami-074be47313f84fa38'
    SecurityGroupIds:
      - !Ref InstanceSecurityGroup
    IamInstanceProfile: !Ref UnifiedInstanceProfile
    UserData: !Base64 |
      #!/bin/bash
      yum update -y
      yum install -y httpd
      systemctl start httpd
      systemctl enable httpd
      # Create a dummy index.html file to pass health check
      echo "<html><h1>Dummy Index</h1></html>" > /var/www/html/index.html
      # Install CodeDeploy Agent
      yum install -y ruby
      cd /home/ec2-user
      wget https://aws-codedeploy-us-west-2.s3.us-west-2.amazonaws.com/latest/install
      chmod +x ./install
      ./install auto
      service codedeploy-agent start
    AvailabilityZone: us-west-2b
    Tags:
      - Key: Name
        Value: SampleAppEC2
      - Key: Intent
        Value: SampleAppEC2
```

**Step 5) IAM Role for CodeBuild:** Write roles for CodeBuild which I am going to use next.

```
CodeBuildServiceRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: 'Allow'
          Principal:
            Service: 'codebuild.amazonaws.com'
          Action: 'sts:AssumeRole'
    Policies:
      - PolicyName: 'CodeBuildPolicy'
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: 'Allow'
              Action:
                - 'logs:CreateLogGroup'
                - 'logs:CreateLogStream'
                - 'logs:PutLogEvents'
                - 's3:GetObject'
                - 's3:PutObject'
                - 's3:ListBucket'
                - 'ec2:Describe*'
                - 'codedeploy:*'
                - 'codepipeline:*'
              Resource: '*'
```

### Step 6) CodeBuild Project: write the script for codeBuild

```
MyCodeBuildProject:
  Type: 'AWS::CodeBuild::Project'
  Properties:
    Name: 'SampleCB'
    ServiceRole: !GetAtt CodeBuildServiceRole.Arn
    Artifacts:
      Type: 'CODEPIPELINE'
    Environment:
      ComputeType: 'BUILD_GENERAL1_SMALL'
      Image: 'aws/codebuild/standard:4.0'
      Type: 'LINUX_CONTAINER'
    Source:
      Type: 'CODEPIPELINE'
    TimeoutInMinutes: 10
```

### Step 7) CodeDeploy Application & Deployment Group: Write script for CodeDeploy application that serves as a container for the deployment configurations and resources and Deployment group which specifies the Ec2 instance where it will deploy.

```
CodeDeployApplication:
  Type: 'AWS::CodeDeploy::Application'
  Properties:
    ApplicationName: 'SampleCDApp'

CodeDeployDeploymentGroup:
  Type: 'AWS::CodeDeploy::DeploymentGroup'
  Properties:
    ApplicationName: !Ref CodeDeployApplication
    DeploymentGroupName: 'SampleDG'
    ServiceRoleArn: 'arn:aws:iam::363010889649:role/CodeDeploy-custom-roles'
    DeploymentConfigName: CodeDeployDefault.OneAtATime
    Ec2TagFilters:
      - Key: Name
        Value: SampleAppEC2
      Type: KEY_AND_VALUE
```

### Step 8) S3 Bucket: This s3 bucket is for CodePipeline to store and reference the Artifacts from.

```
ArtifactBucket:
  Type: 'AWS::S3::Bucket'
  Properties:
    BucketName: 'sample-cicd-artifacts-bucket-62'
```

### Step 9) IAM Role for CodePipeline: write IAM role this role is used by CodePipeline to manage all stages.

```
CodePipelineServiceRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: 'Allow'
          Principal:
            Service: 'codepipeline.amazonaws.com'
          Action: 'sts:AssumeRole'
    Policies:
      - PolicyName: 'CodePipelinePolicy'
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: 'Allow'
              Action:
                - 's3:GetObject'
                - 's3:PutObject'
                - 's3:ListBucket'
                - 'codedeploy:CreateDeployment'
                - 'codedeploy:GetApplication'
                - 'codedeploy:GetApplicationRevision'
                - 'codedeploy:GetDeployment'
                - 'codedeploy:GetDeploymentConfig'
                - 'codedeploy:RegisterApplicationRevision'
                - 'codebuild:BatchGetBuilds'
                - 'codebuild:StartBuild'
                - 'codebuild:BatchGetProjects'
```

**Step 11) CodePipeline:** write codepipeline with source, built and deploy stages.


```
MyPipeline:
  Type: 'AWS::CodePipeline::Pipeline'
  Properties:
    RoleArn: !GetAtt CodePipelineServiceRole.Arn
    ArtifactStore:
      Type: 'S3'
      Location: !Ref ArtifactBucket
    Stages:
      - Name: Source
        Actions:
          - Name: SourceAction
            ActionType:
              Category: Source
              Owner: ThirdParty
              Provider: GitHub
              Version: 1
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              Owner: 'techdecipher'
              Repo: 'using-aws-dev-tools'
              Branch: 'main'
              OAuthToken: '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'
      - Name: Build
        Actions:
          - Name: BuildAction
            ActionType:
              Category: Build
              Owner: AWS
              Provider: CodeBuild
              Version: 1
            InputArtifacts:
              - Name: SourceOutput
            OutputArtifacts:
              - Name: BuildOutput
            Configuration:
              ProjectName: !Ref MyCodeBuildProject
      - Name: Deploy
        Actions:
          - Name: DeployAction
            ActionType:
              Category: Deploy
              Owner: AWS
              Provider: CodeDeploy
              Version: 1
            InputArtifacts:
              - Name: BuildOutput
            Configuration:
              ApplicationName: !Ref CodeDeployApplication
              DeploymentGroupName: !Ref CodeDeployDeploymentGroup
```

## Step 12) Final script

<https://github.com/techdecipher/cloudformation-templates/blob/main/template.yml>

**Step 13) Upload on Cloudformation:** Goto Cloudformation > Create Stack > Upload a template file >

Upload a template file

 Choose file

template.yml



JSON or YAML formatted file

S3 URL: <https://s3.us-west-2.amazonaws.com/cf-templates-il8y2caumvmx-us-west-2/2024-07-26T111135.578Z20p-template.yml>

[View in Application Composer](#)

## Step 14) Stack and its Stages:

CloudFormation > Stacks

Stacks (1)

Filter by stack name

Filter status: Active View nested

Stack name	Status	Created time	Description
<a href="#">sample-CICD-new</a>	CREATE_COMPLETE	2024-07-26 16:44:25 UTC+0530	Complete CI/CD Pipeline with EC2, CodeBuild, CodeDeploy, Deployment Group, and CodePipeline

sample-CICD-new

Stack info | **Events** | Resources | Outputs | Parameters | Template | Change sets | Git sync - new

Events (38)

Search events

Timestamp	Logical ID	Status	Detailed status	Status reason
2024-07-26 16:46:58 UTC+0530	sample-CICD-new	CREATE_COMPLETE	-	-
2024-07-26 16:46:57 UTC+0530	UnifiedInstanceProfile	CREATE_COMPLETE	-	-
2024-07-26 16:46:21 UTC+0530	MyEC2Instance	CREATE_COMPLETE	-	-
2024-07-26 16:46:01 UTC+0530	sample-CICD-new	CREATE_IN_PROGRESS	CONFIGURATION_COMPLETE	Eventual consistency check initiated
2024-07-26 16:46:01 UTC+0530	MyEC2Instance	CREATE_IN_PROGRESS	CONFIGURATION_COMPLETE	Eventual consistency check initiated

## Step 15) Pipeline Creation:

All the stages have been successful and also whenever we hit any change, we get to see the pipeline in action to follow its stages one after the other, finally deploying the artifact.

Developer Tools > CodePipeline > Pipelines

Introducing the new V2 pipeline type with improved release safety, pipeline triggers, parameterized pipelines, and a new billing model. [Learn more](#)

Pipelines Info

Notify View history Release change Delete pipeline **Create pipeline**

Search

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
<a href="#">sample-CICD-new-MyPipeline-biWTQxx5gALc</a> (Type: V1   Execution mode: SUPERSEDED)	✓ Succeeded	SourceAction – <a href="#">666d6e06</a> : Update index.html	4 minutes ago	✓ View details

## Step 16) Final output:

