Step-by-Step Guide: Setting up AWS DevOps Environment

Step 1: Setting up CodeCommit

a. Create IAM User for CodeCommit Access:

- Navigate to the IAM dashboard in the AWS Management Console.
- Click on "Users" and then "Add user."
- Enter a username and select “Attach Policies Directly” in “Permission Option”
- Attach the policies "AwsCodeCommitFullAccess" and "AWSCodeCommitPowerUser" to the user.
- Click "Next" and follow the prompts to create the user.

b. Generate HTTPS Git Credentials:

- After creating the user, navigate to the IAM user's security credentials tab.
- Scroll down to "HTTPS Git credentials for AWS CodeCommit" and click on "Generate."
- Download the CSV file containing the credentials.

c. Create CodeCommit Repository:

- Go to the AWS CodeCommit console.
- Click on "Create repository."
- Enter a repository name and description.
- Choose the repository settings and create the repository.

Step 2: Push Code to CodeCommit using your machine CLI

a. Clone CodeCommit Repository:

- Copy the "code commit URL" from the CodeCommit console.
- Open a terminal and run the command

```
git clone <copied-url>
```

- It will ask you for username and password, give your IAM user credentials that you generated

b. Copy Project Files:

- Navigate into the cloned repository directory
- Run the command

```
< git clone https://github.com/Aakibgithuber/Aws-DevOps-Project >
< mv Aws-DevOps-Project/* . >
< rm -rf Aws-DevOps-Project >
```

c. Commit and Push Changes:

- Add the copied files to the Git repository

```
< git add . >
```

- Commit the changes

```
< git commit -m "New commit" >
```

- Push the changes to CodeCommit:bash

  < git push >
  (Enter your IAM credential)

Step 3: Setup S3 Bucket for Code Storage

- Go to the Amazon S3 console.
- Click on "Create bucket."
- Enter a unique bucket name.
- Configure bucket settings and create the bucket.

Step 4: Setup AWS CodeBuild for Building Code

- Go to the AWS CodeBuild console.
- Click on "Create build project."
- Enter a project name.
- On Repo option enter your repository name and branch to master
- Choose OS to Ubuntu, runtime to standard, and image version to latest.
- Click on “Use a Buildspec file” in the repository
- (Buildspec file= it is a file which tells the code build what steps is needed when you building the code.)
- Click on Build code and leave everything as it is
- After the code is built then go to the Codebuild again and click on Edit --> Artifact
- Change Artifact destination to the S3 bucket created in Step 3.
- Review and create the build project again.
- After Building you will see that your code is stored in S3 bucket.

Step 5: Setup CodeDeploy for Deployment

- Go to the AWS CodeDeploy console.
- Click on "Create application" and specify the application name.
- Choose the compute platform as EC2/on-premise.
- Create a deployment group with the desired configurations

Step 6: Setup EC2 Instance for Application Deployment

a. Launch Ubuntu EC2 Instance:

- Launch an Ubuntu EC2 instance with version 22.04 LTS
- Follow the instance creation wizard to configure instance details, storage, tags,in security group allow port 80, and review.
- Launch the instance and select an existing key pair or create a new one to access the instance.

b. Create AWS EC2 Service Role:

- Go to the IAM dashboard in the AWS Management Console.
- Click on "Roles" and then "Create role."
- Choose the trusted entity type as "AWS service" and the use case as "EC2."
- Select policies to attach to the role:
  - AmazonEC2FullAccess
  - AmazonEC2RoleforAWSCodeDeploy
  - AmazonEC2RoleforAWSCodeDeployLimited
  - AWSCodeDeployFullAccess
  - AWSCodeDeployFullRole

    - AmazonS3FullAcccess
- Complete the role creation wizard and give it a name.

c. Attach Role to EC2 Instance:

- After launching the EC2 instance, select it from the instances list.
- Go to the "Actions" menu, navigate to "Instance Settings," and click on "Attach/Replace IAM Role."
- Select the role created in Step 6b and attach it to the instance.

d. Create Another
CodeDeploy Service Role:

- Go back to the IAM dashboard and click on "Roles" and then "Create role."
- Choose the trusted entity type as "AWS service" and the use case as "CodeDeploy."
- Click on "Set permissions boundary" and choose "Use a permissions boundary to control the maximum role permissions."
- Select "AmazonEC2FullAccess" as the permissions boundary.
- Complete the role creation wizard and give it a name.

Step 7: Setting Up AWS CodeDeploy Agent on Ubuntu EC2

- SSH into the Ubuntu EC2 instance.
- Switch to the root user.
- Create a shell script named "agent.sh" with the provided content.

```
#!/bin/bash
# This installs the CodeDeploy agent and its prerequisites on Ubuntu 22.04.
sudo apt-get update
sudo apt-get install ruby-full ruby-webrick wget -y
cd /tmp
wget https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/releases/codedeploy-agent_1.3.2-1902_all.deb
mkdir codedeploy-agent_1.3.2-1902_ubuntu22
dpkg-deb -R codedeploy-agent_1.3.2-1902_all.deb codedeploy-agent_1.3.2-1902_ubuntu22
sed 's/Depends:.*/Depends:ruby3.0/' -i ./codedeploy-agent_1.3.2-1902_ubuntu22/DEBIAN/control
dpkg-deb -b codedeploy-agent_1.3.2-1902_ubuntu22/
sudo dpkg -i codedeploy-agent_1.3.2-1902_ubuntu22.deb
```

Run the script using the command

<sh agent.sh>

Step 8: Create Deployment Group and Deploy Application
a. Create Deployment Group:

- Go to the AWS CodeDeploy console.
- Click on "Create deployment group."
- Provide the following details:
    - **Deployment group name: your desired name**
    - **Service role:** Go to the IAM service, open the role section, select the service role of code deploy that you have created, copy its ARN, and paste it here.
    - **Deployment type:** In place
    - **Environment configuration:** Check "Amazon EC2 Instances"

        - Key: name
        - Value: Choose your EC2 instance where the CodeDeploy agent is running
    - **Enable load balancing:** Uncheck this option
    - **Install AWS CodeDeploy Agent:** Set to "Never"
- Click on "Create deployment group."

b. Deploy Application:

- After creating the deployment group, click on "Create deployment."
- Provide the following details:
    - **Deployment group:** your desired name
    - **Compute platform:** EC2/on premise
    - **Revision type:** My application stored in Amazon S3
    - **Revision location:** Paste the path of your build code stored in S3. Go to the S3 console, choose the folder containing your build code, and copy the S3 URL.
- Click on "Create deployment."
- After a few seconds, the deployment will be initiated.

Step 9: Access Deployed Application

- Go to the EC2 instance where the application is deployed.
- Ensure that port 80 is allowed in the security groups of the EC2 instance.
- Copy the public IP of the instance and paste it into a web browser.
- If the deployment is successful, you should see your application running on the web page.

Step 10: Setup CodePipeline to Automate the Whole Process

a. Create a Pipeline:

- Go to the AWS CodePipeline console.
- Click on "Create pipeline."
- Provide a name for your pipeline and click on "Next."

b. Configure Pipeline Details:

- Select pipeline type: Choose V2.
- Click on "Create role" to create a new service role if you don't have one already, and then click "Next."

c. Source Stage:

- Choose source provider: Select "AWS CodeCommit."
- Repository name: Choose your CodeCommit repository
- Branch: Select "master."
- Click on "Next."

d. Build Stage:

- Choose build provider: Select "AWS CodeBuild."
- Project name: Choose the CodeBuild project you created earlier
- Click on "Next."

e. Deploy Stage:

- Choose deploy provider: Select "AWS CodeDeploy."
- Application name: Choose the CodeDeploy application you created earlier
- Deployment group: Select the deployment group you created earlier
- Click on "Next."

f. Review and Create:

- Review the pipeline configuration.

- Click on "Create pipeline."

g. Start Pipeline:

- After creating the pipeline, it will be in the "Stopped" state.
- Click on "Release change" to start the pipeline.
- Now, whenever there is a new commit in the CodeCommit repository, the pipeline will automatically trigger the build and deployment process.

This setup ensures that your application deployment process is fully automated using AWS CodePipeline, integrating CodeCommit for source control, CodeBuild for building, and CodeDeploy for deployment.