**Automated Deployment Pipeline and Notifications for Streamlit App on EC2 (GitHub, CodePipeline, CodeDeploy, SNS)**

This guide will walk you through setting up a CI/CD pipeline that automatically deploys changes from your GitHub repository to your EC2 instance every time you push code to your main branch, and then notifies you of the deployment status via email.

**Prerequisites:**

1. **Running EC2 Instance:** Your myphoto-app-ec2 instance should be running with your Streamlit app configured to run via systemd and Nginx acting as a reverse proxy, as per our previous setup.
2. **Git Repository:** Your Streamlit application code, including app.py, aws\_utils.py, config.py, and requirements.txt, is pushed to your GitHub repository.
3. **IAM Role for EC2 Instance:** Your EC2 instance should already have an IAM Instance Profile (role) attached that grants it permissions to access S3 and DynamoDB (e.g., AmazonS3FullAccess and AmazonDynamoDBFullAccess for testing, or more restrictive permissions for production). We will update this role to allow the CodeDeploy Agent to function.

**Part 1: Prepare Your EC2 Instance for CodeDeploy**

This involves attaching an IAM role that allows the CodeDeploy Agent to function and then installing the agent itself.

**1.1. Update EC2 Instance IAM Role (if needed):** Your EC2 instance needs permission to interact with the CodeDeploy service.

* Go to the AWS IAM Console.
* Click on **Roles** in the left navigation pane.
* Find and click on the IAM role currently attached to your myphoto-app-ec2 instance (e.g., myphoto-app-ec2-role if you created one, or the default ec2\_instance\_profile\_for\_codedeploy if you used that).
* Click **Add permissions** -> **Attach policies**.
* Search for AmazonEC2RoleforAWSCodeDeploy and AmazonEC2RoleforAWSCodeDeployLimited. Select **AmazonEC2RoleforAWSCodeDeploy**.
* Click **Add permissions**.

**1.2. Install CodeDeploy Agent on EC2 Instance:** This agent allows CodeDeploy to communicate with your instance and execute deployment scripts.

* **SSH into your myphoto-app-ec2 instance:**

Bash

ssh -i your-key.pem ec2-user@YOUR\_PUBLIC\_IP\_ADDRESS

* **Run the following commands to install the CodeDeploy agent:** (Remember to adjust the wget URL if your EC2 instance is not in eu-west-1 region.)

Bash

sudo yum update -y

sudo yum install -y ruby # CodeDeploy Agent needs Ruby

sudo yum install -y wget

# Navigate to home directory

cd /home/ec2-user

# Download the CodeDeploy agent installer (adjust region if needed)

# For eu-west-1 (Ireland):

wget https://aws-codedeploy-eu-west-1.s3.eu-west-1.amazonaws.com/latest/install

# For us-east-1 (N. Virginia):

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

# For other regions, replace 'eu-west-1' accordingly.

# Make the installer executable

chmod +x ./install

# Run the installer

sudo ./install auto

# Verify the CodeDeploy agent status

sudo service codedeploy-agent status

You should see codedeploy-agent is running in the output. If it's not running, you might need to troubleshoot the installation or check logs in /var/log/aws/codedeploy-agent/.

**Part 2: Prepare Your Git Repository**

You need to add specific files to your repository that instruct CodeDeploy on how to perform the deployment.

**2.1. Create scripts Directory:** In the root of your local Git repository (where your app.py is), create a new directory named scripts.

**2.2. Create Deployment Hook Scripts:** Inside the scripts directory, create the following three files with the exact content provided.

* **scripts/stop\_app.sh**: This script will stop your Streamlit application service before new code is copied.

Bash

#!/bin/bash

echo "Stopping Streamlit app..."

# Use '|| true' to prevent script from failing if the service isn't running (e.g., first deployment)

sudo systemctl stop streamlit\_app || true

echo "Streamlit app stopped."

* **scripts/install\_dependencies.sh**: This script will activate your virtual environment and install/update Python dependencies after the new code is deployed.

Bash

#!/bin/bash

# This script is designed to be run as 'ec2-user' by CodeDeploy

APP\_DIR="/home/ec2-user/my\_photo\_app"

echo "Navigating to app directory: $APP\_DIR"

cd "$APP\_DIR" || { echo "ERROR: Failed to change directory to $APP\_DIR. Exiting."; exit 1; }

echo "Activating virtual environment..."

source .venv/bin/activate || { echo "ERROR: Failed to activate venv. Exiting."; exit 1; }

echo "Installing/updating Python dependencies (pip install -r requirements.txt)..."

pip install -r requirements.txt || { echo "ERROR: Failed to install dependencies. Exiting."; exit 1; }

echo "Python dependencies installed successfully."

deactivate

* **scripts/start\_app.sh**: This script will start your Streamlit application service after dependencies are installed.

Bash

#!/bin/bash

echo "Starting Streamlit app..."

sudo systemctl start streamlit\_app

echo "Streamlit app started."

**2.3. Create appspec.yml:** In the **root** of your local Git repository (the same directory as app.py and the scripts folder you just created), create a file named appspec.yml with the following content:

YAML

version: 0.0

os: linux

files:

- source: / # This means "all files from the root of the deployment package"

destination: /home/ec2-user/my\_photo\_app # Deploy them to this directory on the EC2 instance

hooks:

# This hook runs before the new application revision is copied to the instance.

# We stop the app to ensure no old processes interfere.

BeforeInstall:

- location: scripts/stop\_app.sh

timeout: 60

runas: root # This script needs root privileges for systemctl

# This hook runs after the application revision is copied.

# We install dependencies here as the new code is now in place.

AfterInstall:

- location: scripts/install\_dependencies.sh

timeout: 300 # Give enough time for pip install

runas: ec2-user # This script needs to run as ec2-user to access the virtual environment

# This hook runs after the new code is in place and dependencies are handled.

# We start the application service here.

ApplicationStart:

- location: scripts/start\_app.sh

timeout: 60

runas: root # This script needs root privileges for systemctl

**2.4. Set Executable Permissions and Push to Git:**

* Locally, in your terminal, navigate to your scripts directory:

Bash

cd scripts

* Make the scripts executable:

Bash

chmod +x stop\_app.sh install\_dependencies.sh start\_app.sh

* Go back to your repository root:

Bash

cd ..

* Add all the new files (appspec.yml and the scripts directory) to Git, commit, and push:

Bash

git add .

git commit -m "Add CodeDeploy appspec.yml and deployment scripts"

git push origin main # Or 'master' if that's your branch

**Part 3: Set Up AWS CodeDeploy**

Now you'll configure CodeDeploy to understand your application and how to deploy it.

**3.1. Create an IAM Service Role for CodeDeploy:** CodeDeploy needs permission to access your S3 bucket (where it stores deployment bundles) and to interact with EC2 instances.

* Go to the AWS IAM Console.
* Click **Roles** -> **Create role**.
* For "Trusted entity type", select **AWS service**.
* For "Use case", select **CodeDeploy** from the dropdown, then click **Next**.
* Attach permissions: The policy AWSCodeDeployRole should be pre-selected. Click **Next**.
* Give the role a name (e.g., CodeDeployServiceRole).
* Click **Create role**.

**3.2. Create CodeDeploy Application:**

* Go to the AWS CodeDeploy Console.
* Click **Applications** in the left navigation pane.
* Click **Create application**.
* **Application name:** my-photo-app-codedeploy (or a similar descriptive name).
* **Compute platform:** Select EC2/On-premises.
* Click **Create application**.

**3.3. Create CodeDeploy Deployment Group:**

* Once your application is created, click on it.
* Click **Create deployment group**.
* **Deployment group name:** my-photo-app-production (or my-photo-app-dev).
* **Service role:** Select the IAM role you created in step 3.1 (e.g., CodeDeployServiceRole).
* **Deployment type:** Choose In-place.
* **Environment configuration:**
  + Select Amazon EC2 instances.
  + Choose Tags and add the tag you use for your EC2 instance (e.g., Name = myphoto-app-ec2). This tells CodeDeploy which instances to deploy to.
* **Deployment settings:** You can leave the default CodeDeployDefault.AllAtOnce for now.
* **Load balancer:** Uncheck Enable load balancing (you are using Nginx directly on the instance).
* Click **Create deployment group**.

**Part 4: Set Up AWS CodePipeline for Automation**

This connects your Git repository to CodeDeploy, making the deployment automatic on every push.

**4.1. Create a New Pipeline:**

* Go to the AWS CodePipeline Console.
* Click **Pipelines** -> **Create pipeline**.
* **Pipeline name:** my-photo-app-pipeline
* **Service role:**
  + Choose New service role and keep the default role name (e.g., AWSCodePipelineServiceRole-your-region-your-pipeline-name).
  + Or, if you prefer, choose Existing service role and select a role with AWSCodePipelineFullAccess policy.
* **Advanced settings:** You can typically leave defaults.
* Click **Next**.

**4.2. Add Source Stage (Connect to GitHub):**

* **Source provider:** Select GitHub (Version 2) (recommended for newer setups for better integration).
  + **Important:** You will need to "Connect to GitHub" the first time, which involves authorizing AWS to access your GitHub account.
* **Repository name:** Select your repository (e.g., your-username/my\_photo\_app).
* **Branch name:** Select main (or master).
* **Output artifact format:** Choose CodePipeline default.
* **Change detection options:** Ensure Start the pipeline on source code change is checked.
* Click **Next**.

**4.3. Skip Build Stage (Optional for this app):**

* Since your Streamlit app is purely Python and doesn't require compilation (like Java or .NET apps), you can **Skip the build stage**.
* Click **Skip build stage?** -> **Skip**.

**4.4. Add Deploy Stage:**

* **Deploy provider:** Select AWS CodeDeploy.
* **Application name:** Select my-photo-app-codedeploy (the one you created in step 3.2).
* **Deployment group:** Select my-photo-app-production (the one you created in step 3.3).
* Click **Next**.

**4.5. Review and Create Pipeline:**

* Review all the settings.
* Click **Create pipeline**.

**Part 5: Set Up Email Notifications for CodePipeline**

This will send email alerts for pipeline success or failure.

1. **Go to the AWS SNS Console:**
   * Search for "SNS" in the AWS console search bar and click on "Simple Notification Service".
   * In the left navigation pane, click on **Topics**.
2. **Create a New Topic:**
   * Click **Create topic**.
   * **Type:** Select Standard.
   * **Name:** Give your topic a descriptive name, e.g., my-photo-app-deployment-notifications.
   * **Display name:** (Optional, short name for email subject line) e.g., PhotoAppDeploy.
   * Click **Create topic**.
3. **Subscribe Your Email Address to the SNS Topic:**
   * Select your newly created topic (my-photo-app-deployment-notifications).
   * Click on the **Subscriptions** tab.
   * Click **Create subscription**.
   * **Topic ARN:** This will be pre-filled.
   * **Protocol:** Select Email.
   * **Endpoint:** Enter the email address where you want to receive notifications.
   * Click **Create subscription**.
4. **Confirm the Subscription (CRITICAL STEP!):**
   * **Check your email inbox** for an email from "AWS Notifications" (it might go to spam).
   * The email will contain a link that says **"Confirm subscription"**.
   * **You MUST click this link** to activate the subscription. Until you confirm, you will not receive notifications.
   * After clicking, refresh the "Subscriptions" tab in the SNS console; the "Status" should change to Confirmed.
5. **Configure CodePipeline Notification Rule:**
   * Go to the AWS CodePipeline Console.
   * Select your pipeline (e.g., my-photo-app-pipeline).
   * At the top of the pipeline visualization, click on the **Notify** tab.
   * Click **Create notification rule**.
   * **Rule name:** Deployment-Status-Notifications (or similar).
   * **Detail type:** Leave as Basic.
   * **Events that trigger notifications:** Select the events you care about, typically:
     + Pipeline execution failed
     + Pipeline execution succeeded
   * **Targets:**
     + For "Target type", select SNS topic.
     + For "Topic", select your newly created SNS topic (e.g., my-photo-app-deployment-notifications).
   * Click **Submit**.

**Part 6: Test Your CI/CD Pipeline with Notifications**

1. **Monitor Initial Run:**
   * Once the pipeline is created, it will automatically start its first execution.
   * Watch the stages in the CodePipeline console.
2. **Make a Small Change and Push:**
   * On your local machine, make a tiny, harmless change to app.py (e.g., add a comment or change a string).
   * Save the file.
   * Commit the change: git commit -am "Triggering pipeline test for notifications"
   * Push to GitHub: git push origin main (or master)
3. **Observe Automatic Deployment and Email Notifications:**
   * Go back to the AWS CodePipeline console; you should see a new execution start automatically.
   * Monitor its progress.
   * **Check your email inbox.** You should receive email notifications when the pipeline starts, and then another one when it either succeeds or fails.
   * Finally, check your application in the browser using your EC2 instance's Public IP to confirm the change was deployed successfully.