**Step-by-Step Guide to Set Up AWS CodeDeploy with GitHub Actions**

**Prerequisites**

1. **AWS Account**: Ensure you have an AWS account.
2. **IAM User with Required Permissions**: Create an IAM user with permissions for CodeDeploy, EC2, and S3 (if necessary).
3. **GitHub Repository**: Have your code repository set up on GitHub.
4. **EC2 Instance**: Have an EC2 instance set up and running.
5. **Docker Hub Account**: Have a Docker Hub account for hosting your Docker images.

**Step 1: Set Up EC2 Instance**

1. **Launch EC2 Instance**:
   * Open the Amazon EC2 console.
   * Choose "Launch Instance."
   * Select an Amazon AMI(e.g., Amazon Linux 2).
   * Choose an instance type, Eg. t2.micro.
   * Configure instance details, add storage, add tags, and configure security group (allow SSH and HTTP/HTTPS).
2. **Add User Data Script**: Add the following user data script to install Docker and AWS CodeDeploy agent:
3. #!/bin/bash
4. # Update the package list and install necessary packages
5. sudo yum update -y
6. sudo yum install -y ruby wget
7. # Install Docker
8. sudo yum install docker -y
9. sudo usermod -aG docker ec2-user
10. sudo service docker start
11. # Enable Docker to start on boot
12. sudo systemctl enable docker
13. # Install AWS CodeDeploy Agent
14. cd /home/ec2-user
15. wget https://aws-codedeploy-us-west-2.s3.us-west-2.amazonaws.com/latest/install
16. chmod +x ./install
17. sudo ./install auto
18. # Start the CodeDeploy agent
19. sudo service codedeploy-agent start
20. # Enable CodeDeploy agent to start on boot
21. sudo systemctl enable codedeploy-agent
22. # Print versions to verify installation
23. docker --version
24. codedeploy-agent --version
25. **Launch the Instance**: Review and launch the instance. Ensure you download the key pair (.pem file) for SSH access.

**Step 2: Create IAM Role for CodeDeploy**

1. **Open the IAM Console**:
   * Navigate to the AWS Management Console.
   * Go to the IAM (Identity and Access Management) service.
2. **Create a New Role**:
   * In the left navigation pane, select "Roles."
   * Click on the "Create role" button.
3. **Select Trusted Entity**:
   * Choose the type of trusted entity as "AWS service."
   * Under "Use case," select "EC2" and click "Next: Permissions."
4. **Attach Permissions Policies**:
   * Search for and select the following policies:
     + **AWSCodeDeployFullAccess**
     + **AmazonS3ReadOnlyAccess** (if your CodeDeploy application accesses S3 for deployment packages)
   * Click "Next: Tags" (you can add tags if needed) and then "Next: Review."
5. **Name the Role**:
   * Enter a role name (e.g., **EC2CodeDeployRole**).
   * Add a description if desired.
   * Click "Create role."

**Step 3: Attach IAM Role to EC2 Instance**

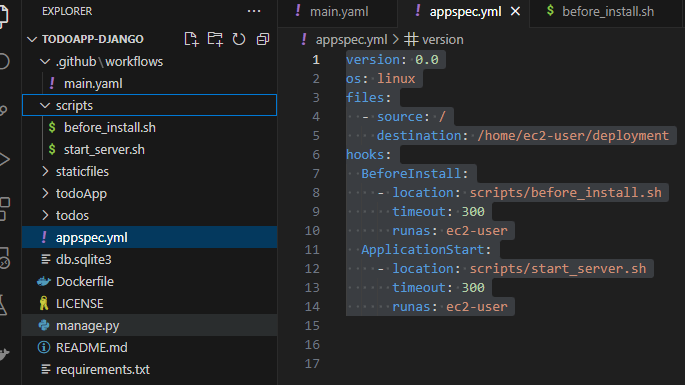
1. **Open the EC2 Console**:
   * Navigate to the AWS Management Console.
   * Go to the EC2 service.
2. **Select the Instance**:
   * In the left navigation pane, select "Instances."
   * Find and select the instance you want to add the IAM role to.
3. **Attach IAM Role**:
   * With your instance selected, click the "Actions" button.
   * Navigate to "Security" > "Modify IAM role."
   * In the "IAM role" dropdown, select the role you created earlier (e.g., **EC2CodeDeployRole**).
   * Click "Update IAM role."

**Step 4: Configure AWS CodeDeploy**

1. **Create CodeDeploy Application**:
   * Open the AWS CodeDeploy console.
   * Choose "Create application."
   * Enter an application name (e.g., **todoapp**).
   * Choose the compute platform as "EC2/On-premises."
2. **Create Deployment Group**:
   * Choose the application you created.
   * Choose "Create deployment group."
   * Enter a name for the deployment group (e.g., **todoapp-deployment-group**).
   * Select the service role with CodeDeploy permissions.
   * In "Deployment type," select "In-place."
   * In "Environment configuration," choose "Amazon EC2 instances" and select your EC2 instances using tags or manually.
   * Uncheck load balancer settings .
   * Click "Create deployment group."

**Step 5: Prepare GitHub Repository**

1. **Create Deployment Scripts**:
   * In your GitHub repository, create the following directory structure:



Create a directory named “scripts” in your root directory

1. **Add Deployment Scripts**:
   * **scripts/before\_install.sh**:

#!/bin/bash

docker pull abudev22/todoapp:latest

docker stop my-website || true

docker rm my-website || true

make to replace with your docker username and image name

* + **deploy/scripts/start\_server.sh**:

#!/bin/bash

docker run -d --name my-website -p 8000:8000 abudev22/todoapp:latest

1. **create an AppSpec File in your root directory**:
   * **appspec.yml**:
2. version: 0.0
3. os: linux
4. files:
5. - source: /
6. destination: /home/ec2-user/deployment
7. hooks:
8. BeforeInstall:
9. - location: scripts/before\_install.sh
10. timeout: 300
11. runas: ec2-user
12. ApplicationStart:
13. - location: scripts/start\_server.sh
14. timeout: 300
15. runas: ec2-user

**Step 6: Set Up GitHub Actions Workflow**

1. **Create Workflow File**:
   * In your GitHub repository, create the following directory structure:

.github/ workflows/ deploy.yml

1. **Add Workflow Configuration**:

Make sure to change the application name and application group to be the same as the one you created in

* + **main.yml**:

1. name: Deploy website to EC2 using AWS CodeDeploy
2. on:
3. push:
4. branches:
5. - devops
6. jobs:
7. deploy:
8. runs-on: ubuntu-latest
9. steps:
10. - name: Checkout code
11. uses: actions/checkout@v3
12. - name: Login to Docker Hub
13. uses: docker/login-action@v3
14. with:
15. username: ${{ secrets.DOCKER\_HUB\_USERNAME }}
16. password: ${{ secrets.DOCKER\_HUB\_PASSWORD }}
17. - name: Build and push Docker image
18. run: |
19. docker build -t ${{ secrets.DOCKER\_HUB\_USERNAME }}/todoapp .
20. docker push ${{ secrets.DOCKER\_HUB\_USERNAME }}/todoapp:latest
22. - name: Deploy to AWS CodeDeploy
23. run: |
24. aws deploy create-deployment \
25. --application-name todoapp \
26. --deployment-group-name todoapp-deployment-group \
27. --github-location repository=${{ github.repository }},commitId=${{ github.sha }}
28. env:
29. AWS\_ACCESS\_KEY\_ID: ${{ secrets.AWS\_ACCESS\_KEY\_ID }}
30. AWS\_SECRET\_ACCESS\_KEY: ${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}
31. AWS\_REGION: us-east-2

**Step 7: Set Up GitHub Secrets**

1. **Add GitHub Secrets**:
   * In your GitHub repository, go to "Settings" > "Secrets" > "Actions."
   * Add the following secrets:
     + **DOCKER\_HUB\_USERNAME**: Your Docker Hub username.
     + **DOCKER\_HUB\_PASSWORD**: Your Docker Hub password.
     + **AWS\_ACCESS\_KEY\_ID**: Your AWS access key ID.
     + **AWS\_SECRET\_ACCESS\_KEY**: Your AWS secret access key.

**Step 8: Push Changes to GitHub**

1. **Push Changes**:
   * Commit and push your changes to the **main** branch in your GitHub repository.
2. **Monitor Deployment**:
   * Monitor the GitHub Actions workflow for any issues.
   * Monitor the AWS CodeDeploy console for the deployment status.

By following these steps, you will have a complete setup for deploying your application from GitHub to EC2 using AWS CodeDeploy.

Congrats!