Git Documentation for AWS Projects

Introduction

This document explains how to use Git when working on AWS-based projects. It includes best practices, common commands, and integration tips for GitHub and AWS services.

Git Setup for AWS

- Install Git on your system (https://git-scm.com/downloads)
- Configure Git:
  git config --global user.name "Your Name"
  git config --global user.email "you@example.com"

Initializing a Repository

- Navigate to your AWS project directory
- Run:
  git init
- Add files:
  git add .
- Commit:
  git commit -m "Initial commit"

Connecting to GitHub

- Create a new repository on GitHub
- Link it with your local repo:
  git remote add origin https://github.com/yourusername/your-repo.git
- Push changes:
  git push -u origin main

Using Git in AWS Projects

- Track infrastructure code (e.g., Terraform, CloudFormation)
- Track Lambda functions, APIs, S3-hosted websites
- Keep a `.gitignore` to avoid uploading AWS credentials

GitHub Actions with AWS

- Use GitHub Actions to automate deployment to AWS
- Store AWS credentials in GitHub Secrets (e.g., `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`)
- Sample GitHub Actions Workflow:

```yaml
name: Deploy to AWS
on: [push]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3
    - uses: aws-actions/configure-aws-credentials@v2
      with:
        aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
        aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
        aws-region: us-east-1
    - run: aws s3 sync ./site s3://your-bucket-name
```

Best Practices

- Use branches for new features or experiments
- Never commit AWS credentials
- Use `.gitignore` to exclude `node_modules`, `env` files, etc.
- Regularly push changes to remote repo for backup and collaboration