Akilandeshwari Srinivasan(451036)

CLCM3504 – Lab 3

Due by : 03- 12-2023

1. **AWS Command Line Interface (CLI):**

   The AWS Command Line Interface (CLI) is a set of tools provided by Amazon Web Services (AWS) to interact with AWS services using a command-line interface. Its primary purpose is to enable users to manage and automate various AWS resources and services directly from the command line. The AWS CLI allows users to perform tasks such as creating and configuring AWS resources, deploying applications, and accessing AWS services, providing a flexible and efficient way to manage AWS environments.

2. **Configuration the AWS CLI**

   To configure the AWS CLI on the local machine, follow these steps:

   - Install AWS CLI: Download and install the AWS CLI on your machine by following the instructions provided for your operating system.

   - AWS Access Key and Secret Key: Obtain your AWS Access Key ID and Secret Access Key from the AWS Management Console under the IAM (Identity and Access Management) section.

   - Open a Terminal or Command Prompt: Launch the command-line interface on your machine.

   - Run `aws configure`: Enter the following command and provide the requested information:
     aws configure

   You will be prompted to enter your AWS Access Key ID, Secret Access Key, default region name, and default output format.

- Verify Configuration: Confirm that the configuration is successful by running a simple AWS CLI command, such as:

aws s3 ls

This command lists the available S3 buckets, and if successful, it indicates that your AWS CLI is configured correctly.

Now your AWS CLI is set up on your local machine, allowing you to interact with AWS services using the command line.

3. **Command for creating S3 bucket**

   To create a new S3 bucket using the AWS CLI, use the following command:

   aws s3api create-bucket --bucket BUCKET_NAME --region REGION

   Replace `BUCKET_NAME` with the desired name for new S3 bucket and `REGION` with the AWS region where we want to create the bucket. This command creates a new S3 bucket in the specified region.

4. **Purpose of AWS CLI profiles and how to use them:**

   AWS CLI profiles are useful for managing multiple sets of AWS credentials and configurations on the same machine. They allow me to switch between different AWS accounts or IAM (Identity and Access Management) roles seamlessly.

   To use AWS CLI profiles, I can configure them by adding the `--profile` flag when running AWS CLI commands.

   aws configure --profile myprofile

   This command sets up a new profile named "myprofile." Subsequently, when running AWS CLI commands, I can specify the profile to use:

   aws s3 ls --profile myprofile

This ensures that the command uses the credentials and configurations associated with the "myprofile" profile. Profiles enhance flexibility and security, allowing me to compartmentalize AWS access for different tasks within the context of my college assignments.

5. **Git vs Github:**

GitHub is a web-based platform for version control and collaborative software development. It provides a hosting service for Git repositories along with additional features like issue tracking, pull requests, code review, and project management tools. GitHub simplifies collaboration among developers by offering a centralized platform where they can contribute to projects, track changes, and manage code repositories.

Git, on the other hand, is a distributed version control system designed for tracking changes in source code during software development. It operates locally on a user's machine and allows multiple developers to work on a project simultaneously. Git enables versioning, branching, merging, and tracking changes efficiently.

6. **Github Actions:**

GitHub Actions is a feature of GitHub that allows you to automate various workflows directly within your GitHub repository. It helps you build, test, and deploy your code without relying on external CI/CD (Continuous Integration/Continuous Deployment) services.

Purpose of GitHub Actions:
1. Automated Workflows: GitHub Actions enables the creation of automated workflows triggered by events like code pushes, pull requests, or issue updates.

2. CI/CD: It is commonly used for Continuous Integration and Continuous Deployment, automating tasks such as testing code changes and deploying applications.

3. Task Automation: Beyond CI/CD, GitHub Actions can automate various tasks, like running tests, updating documentation, or notifying team members.

Example Use Case:
Let's consider a scenario of a web application hosted on GitHub. Upon a new pull request being opened, a GitHub Action workflow can be triggered to:

- Run Tests: Automatically execute a suite of tests to ensure that the code changes do not introduce new issues.

- Code Quality Checks: Perform static code analysis or other code quality checks.

- Notify Teams: Notify relevant teams or individuals about the pull request.

- Automated Deployment: If the pull request is merged into the main branch, GitHub Actions can automatically deploy the updated code to a staging environment for further testing.

7. Difference between git commit and git push commands in Git:

git commit:

This command is used to record changes to the local repository. When you make changes to your code and want to save those changes in the version history of your local repository, you use git commit.
Function: Commits create a snapshot of the changes you've made, along with a commit message that describes what you did. This allows you to track the progress of your project and revert to previous states if needed.
git commit -m "Your commit message here"

git push:

This command is used to update a remote repository with the changes you've made in your local repository. After committing changes locally, you may want to share those changes with others or update the central repository (e.g., on GitHub or GitLab).
Function: git push sends your committed changes to the remote repository, making them accessible to others working on the project.
git push origin branch_name