**NAME: LIKHITH SASANK UPPALAPATI VENKATA**

**DAY: 1**

1. **AWS IAM Setup:**

**Create IAM users, groups, and roles.**

**Assign permissions and policies.**

**Test login using IAM sign-in link.**

1. **Billing Alert and CloudWatch Alarm:**

**Enable billing alerts in the AWS Billing Console.**

**Create a CloudWatch alarm to track monthly charges.**

**Set up an SNS topic to send cost alerts via email.**

1. **Amazon S3 Bucket:**

**Create a new S3 bucket.**

**Upload files of different types (PDF, image, CSV).**

**Enable bucket versioning and upload the same file again to check multiple versions.**

1. **GitHub Setup:**

**Create a new repository for OUBT bootcamp tasks.**

**Clone the repository locally.**

**Add the Day 1 Word document and screenshots.**

**Commit and push the files to GitHub.**

**1. Introduction**

This week focused on AWS fundamentals, IAM, S3, GitHub basics, and Python for Data Engineering. The goal was to understand access management, cloud storage, and data handling concepts.

**2. IAM Concepts Summary**

**IAM Overview:**  
IAM – It is a security’s service that will control who can access our AWS account

**IAM – USERS:**

Users means one individual person that needs an access

**Example** : Let’s say we are a team of developers, admin, testers. Each person needs a particular access so we create users as Likhith.Dev, Sasank.Admin, Venkat.Tester and give them roles and permissions as shown below.

Likhith.Dev -> **Developer** -> S3, Glue, RDS, Lambda

Sasank.Admin -> **Admin** -> Full Access

Venkat.Tester -> **Tester** -> Read Only

Here we attach policies to Users.

**IAM – GROUPS:**

Groups are the collection of Users who share same permissions.

Here we do assign policies to groups not for users separately as shown below,

**Developers** -> Access to S3, Glue, RDS, Lambda -> Members (Likhith, Sasank)

**Admin** -> Full Access -> Members (Venkat)

**Tester** -> Read Only Access (Durga)

**IAM ROLES:**

Role is a temporary identity with specific permissions.

* AWS service assumes a role
* Users assume a role
* Group assumes a role

Example : Suppose I have a microservice or may be a spring boot app deployed in EC2 that basically needs to read and write to RDS, So instead of hardcoding credentials, I can create a role and assign to our EC2 instance for access to RDS, So it gets permission automatically.

**IAM Policy:**

Policy defines permissions.

It is a json object that says what is allowed and what is not allowed.

**Billing Alerts:**

This helps to avoid surprise charges .

* We can use cloudwatch alerts / Billing alarms to notify us when usage exceeds a limit.

Here Am showing all my hand-ON PRACTICE,

**Task 1: Created IAM User**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Successfully created user,

A screenshot of a computer

AI-generated content may be incorrect.

**Task 2: Created S3 Bucket:**

A screenshot of a computer

AI-generated content may be incorrect.

A close-up of a computer screen

AI-generated content may be incorrect.

Successfully created S3-BUCKET,

A screenshot of a computer

AI-generated content may be incorrect.

**Task 3: Billing Alert**

Set up CloudWatch Billing Alarm for $1 threshold with email alert.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Finally created an alarm,

A screenshot of a computer

AI-generated content may be incorrect.

**S3 - Section**

Here I have uploaded the files to S3,

A screenshot of a computer

AI-generated content may be incorrect.

**S3 Bucket Versioning Test**

I uploaded the same PDF file twice to my S3 bucket. With versioning enabled, both versions of the file are saved. The screenshot below shows the **current version** and the **previous version**. This confirms that versioning is working correctly and older files can be restored if needed.

**Screenshot inserted below.**

A screenshot of a computer

AI-generated content may be incorrect.

**Summary :**

Today I learned how AWS **Identity and Access Management (IAM)** helps control and secure user access across AWS services. I created **IAM users, groups, and roles**, assigned appropriate permissions, and understood the concept of least privilege for secure account management.

I also explored **Amazon S3**, where I created a bucket, uploaded multiple file types, and enabled **versioning** to maintain file history and data recovery. In addition, I configured **AWS CloudWatch billing alerts and alarms** to track usage and receive cost notifications, ensuring proactive budget management.

Finally, I set up a **GitHub repository** to store and manage all my bootcamp work, learned the basics of commits and pushing files, and organized my Day 1 deliverables systematically. Overall, I gained hands-on experience in foundational AWS services, account monitoring, and version control — key building blocks for cloud and data engineering.