GENERATION INDIA AWS RE/START PROGRAM

Generation
INDIA

A

PROJECT REPORT

ON

AWS STORAGE MANAGEMENT & AUTOMATION

Submitted By

MONALI DATTATRAY MAINDAD

Guided By

Prof.G.Nilesh

Vinsys IT Services Krvenagar,Pune
2025-2026

ABSTRACT

This project focuses on managing and protecting data using AWS storage services such as Amazon Elastic Block Store (EBS) and Amazon Simple Storage Service (S3). The objective of the lab was to automate EBS snapshot creation, manage snapshot retention, and synchronize data from EBS volumes to Amazon S3 using AWS Command Line Interface (AWS CLI).

In this lab, an AWS environment consisting of a Virtual Private Cloud (VPC) with EC2 instances named *Command Host* and *Processor* was used. The Command Host instance was utilized to administer AWS resources, while the Processor instance hosted the EBS volume whose data was managed. IAM roles were configured to provide secure access to AWS services.

Manual and automated EBS snapshots were created using AWS CLI, and Linux cron jobs were configured to schedule snapshot creation at regular intervals. A Python script was executed to retain only the two most recent snapshots, ensuring efficient storage management and cost optimization.

Additionally, an Amazon S3 bucket was created with versioning enabled to support data backup and recovery. Files from the EBS volume were synchronized to the S3 bucket using the aws s3 sync command. The project also demonstrated secure deletion of files and successful recovery of deleted files using S3 object versioning.

INDEX

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

Cloud storage management is a critical aspect of ensuring data availability, reliability, and security in modern cloud-based applications. Amazon Web Services (AWS) provides scalable and highly available storage services such as Amazon Elastic Block Store (EBS) and Amazon Simple Storage Service (S3) to efficiently manage and protect data.

This project, Managing Storage using AWS, focuses on implementing automated storage management techniques using AWS services and tools. The project demonstrates how EBS snapshots can be created, scheduled, and retained using AWS Command Line Interface (AWS CLI), Linux cron jobs, and Python scripting. It also highlights how data stored on EBS volumes can be synchronized with Amazon S3 for backup purposes.

Additionally, the project explores the use of Amazon S3 versioning to handle accidental file deletions and enable data recovery. By integrating IAM roles for secure access, this project follows AWS best practices for security and automation. Overall, the project provides hands-on experience in cloud storage automation, data protection, and recovery mechanisms in a real-world AWS environment.

1.2 AIM OF THIS PROJECT

The aim of this project is to design and implement an automated cloud storage management solution using Amazon Web Services (AWS). The project focuses on creating and managing Amazon EBS snapshots, scheduling automated backups, retaining only required snapshots, and synchronizing data from EBS volumes to Amazon S3. Additionally, the project aims to enable secure data backup, deletion handling, and recovery using Amazon S3 versioning while following AWS best practices for security and automation.

1.3 PROBLEM SATATEMENT

Manual management of Amazon EBS backups and cloud storage is inefficient and prone to errors. There is a need for an automated solution to create and manage EBS snapshots, synchronize data to Amazon S3, and recover deleted files securely. This project addresses these challenges by using

AWS CLI, automation, and S3 versioning to ensure reliable data backup and recovery.

1.4 PROBLEM SOLUTION

The problem is solved by implementing an automated storage management system using AWS services. Amazon EBS snapshots are created and scheduled using AWS CLI and Linux cron jobs to ensure regular backups. A Python script is used to retain only the required snapshots, optimizing storage usage. Data from EBS volumes is synchronized to Amazon S3, where versioning is enabled to support secure backup and easy recovery of deleted files. IAM roles are configured to provide secure and controlled access to AWS resources.

1.5 OBJECTIVE

- To automate Amazon EBS snapshot creation using AWS CLI and cron jobs.
- To manage snapshot retention using Python scripting for storage optimization.
- To synchronize data from EBS volumes to Amazon S3 securely.
- To enable data recovery using Amazon S3 versioning.
- To implement IAM roles for secure access to AWS storage services.

Chapter 2

LITERATURE SURVEY

2.1 RELATED WORK

Cloud computing has become a foundational technology for scalable and reliable data storage and backup solutions. Amazon Web Services (AWS) provides managed storage services such as Amazon Elastic Block Store (EBS) and Amazon Simple Storage Service (S3), which are widely adopted in cloud infrastructures. Amazon Web Services Documentation (2023) describes EBS snapshots as point-in-time backups stored in Amazon S3, enabling efficient data protection and disaster recovery strategies.[1]

Armbrust et al. (2010) discussed the benefits of cloud automation in their study *"A View of Cloud Computing"*, highlighting how automated backup and resource management reduce operational overhead and human errors in large-scale systems.[2] Similarly, **Buyya et al**. (2013) emphasized the importance of automation and scheduling mechanisms in cloud environments to improve reliability and cost efficiency.[3]

Linux cron scheduling has long been used for task automation in distributed systems. **Nemeth et al.** (2017), in *UNIX and Linux System Administration Handbook*, explain how cron jobs provide a reliable method for executing scheduled tasks such as automated backups.[4] Python scripting is frequently used for cloud automation due to its simplicity and extensive AWS SDK support, as discussed by Lutz (2013) in *Learning Python*.[5]

Amazon S3 versioning is recognized as a best practice for data protection. AWS Whitepapers (2022) state that enabling versioning allows recovery from accidental deletions and overwrites, ensuring data durability and business continuity.[6] Additionally, IAM-based access control is highlighted by **Ferraiolo et al**. (2016) as an effective method for enforcing security and least-privilege access in cloud systems.[7]

These studies collectively demonstrate that automated snapshot management, secure storage, and versioned backups are essential components of modern cloud storage architectures, forming the foundation for this project.

Chapter 3

SOFTWARE REQUIREMENTS SPECIFICATION

4.1 SYSTEM REQURIMENTS

4.1.1 Hardware Requirements

- Processor: Intel Core i5 or higher
- RAM: 8 GB or more
- Storage: 500 GB SSD
- Monitor: HD Display
- EC2 instance (minimum t2.micro)
- EBS volumes

4.1.2 Software Requirements

- AWS CLI v2
- Python 3.x
- Linux OS (Amazon Linux)

4.2 DEVELOPMENT PROCESS

1. **Requirement Gathering** – Identify user needs and recommendation techniques.
2. **Planning:** Requirements gathering, project scope definition, and technology selection.
3. **Design:** User interface design, database schema design, and system architecture planning.
4. **Development:** Frontend and backend development, database implementation, and integration testing.
5. **Testing:** Functional testing, usability testing, and bug fixing.
6. **Deployment:** Deployment to a production server for public access.

4.3 FUNCTIONAL REQUIREMENTS

1) EBS Snapshot Creation: The system shall create snapshots of specified EBS volumes using AWS

CLI.

2) Automated Snapshot Scheduling: The system shall schedule snapshot creation using cron jobs.

3) Snapshot Retention Management: The system shall retain only the latest snapshots using a Python script.

4) S3 Data Synchronization: The system shall synchronize EBS volume data to an S3 bucket.

5) S3 Versioning: The system shall enable versioning on the S3 bucket to support file recovery.

6) File Recovery: The system shall restore deleted files using S3 object version IDs.

7) Secure Access Control: The system shall use IAM roles to securely access AWS services

4.4 NON-FUNCTIONAL REQUIREMENTS

1) Security

- Access shall be controlled using IAM roles and policies.
- No hardcoded AWS credentials shall be used.

2) Reliability

- Automated snapshots shall ensure data availability.
- S3 versioning shall prevent permanent data loss.

3) Performance

- Snapshot creation and sync operations shall not impact EC2 performance significantly.

4) Scalability

- The system shall support multiple EBS volumes and large datasets.

5) Maintainability

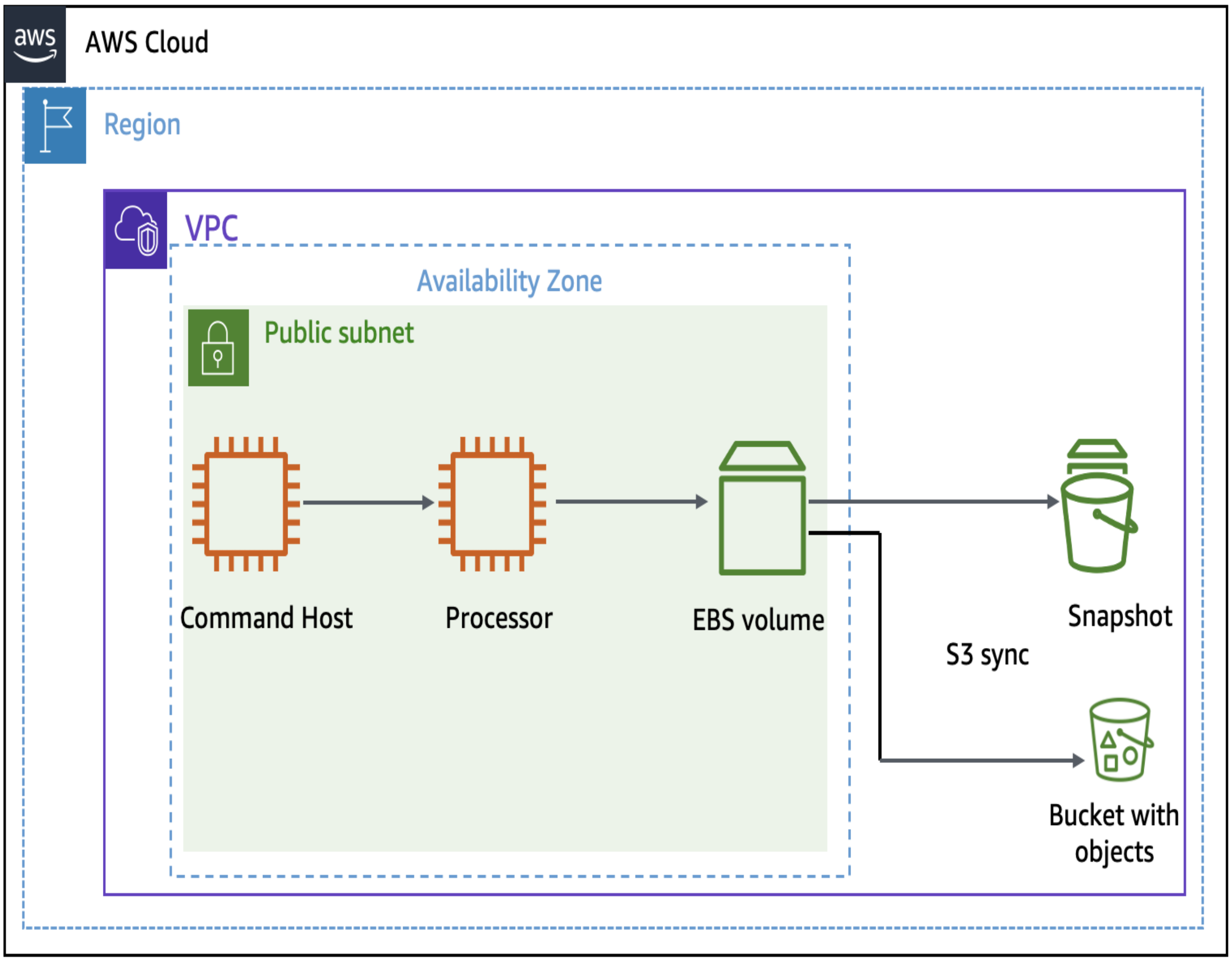
- Scripts shall be modular and easy to modify.

Chapter 4

SYSTEM ARCHITECTURE

5.1 SYSTEM ARCHITECTURE:-

The system is designed on **AWS cloud infrastructure** within a **Virtual Private Cloud (VPC)**. It uses two EC2 instances to manage storage operations, along with Amazon EBS for block storage and Amazon S3 for object storage and backup. Automation is achieved using **AWS CLI, Linux cron jobs, and Python scripts**.



1) ARCHITECTURAL COMPONENTS

1.1) Virtual Private Cloud (VPC)

A **Virtual Private Cloud (VPC)** is a logically isolated virtual network within the AWS cloud that allows users to launch AWS resources in a secure and controlled environment. In this project, the VPC serves as the **foundation of the system architecture**, ensuring network-level security and controlled access between cloud resources.

The VPC is configured with a **public subnet**, which enables EC2 instances to communicate with the internet. This public subnet is associated with a **route table** that directs outbound traffic to an **Internet Gateway**, allowing instances to access AWS services such as Amazon S3 and Amazon EBS through the AWS CLI.

The **Command Host** and **Processor EC2 instances** are deployed within this public subnet. Placing these instances in the same subnet ensures reliable communication between them while maintaining a simplified network design suitable for lab and learning environments. Security groups attached to the instances act as **virtual firewalls**, allowing only required inbound and outbound traffic such as SSH access.

By using a VPC, the architecture achieves:

- Network isolation from other AWS customers
- Secure communication between EC2 instances and AWS services
- Controlled internet access through defined routing rules
- Improved security and manageability of cloud resources

Overall, the VPC provides a **secure, scalable, and well-structured networking layer** that supports automated snapshot creation, S3 synchronization, and storage management operations in this project.

1.2) EC2 Instances

Amazon EC2 provides virtual machines where you can host the same kinds of applications that you might run on a traditional on-premises server. It provides secure, resizable compute capacity in the cloud. EC2 instances can support a variety of workloads.

a) Command Host

- Acts as the **administration server**.
- Used to:

    - Execute AWS CLI commands
    - Schedule cron jobs
    - Run Python scripts for snapshot retention

- Communicates with AWS services using IAM permissions.

b) Processor Instance

- Attached with an **EBS volume** that stores application data.
- Used to:
    - Store files
    - Sync data with Amazon S3
- Has an IAM role attached for secure S3 and EBS access.

1.3) Amazon Elastic Block Store (EBS)

Amazon Elastic Block Store (EBS) is a high-performance, persistent block storage service designed for use with Amazon EC2 instances. In this project, an **EBS volume is attached to the Processor EC2 instance**, where it serves as the primary storage location for application files and data.

EBS provides **data persistence**, meaning the data stored on the volume remains available even if the EC2 instance is stopped or restarted. This makes it suitable for storing critical files that require reliability and durability. The Processor instance uses this EBS volume to store files that are later synchronized with Amazon S3 for long-term storage and backup.

To ensure data protection and disaster recovery, **EBS volume snapshots are created at regular intervals**. These snapshots are **point-in-time backups** stored securely in Amazon S3. Snapshot creation is automated using the **AWS CLI and Linux cron scheduler**, reducing the need for manual intervention.

A Python-based snapshot retention script is used to manage storage costs by **retaining only the most recent snapshots and deleting older ones**. This approach ensures an optimal balance between data availability and cost efficiency.

Key benefits of using Amazon EBS in this project include:

- Reliable and persistent storage for EC2 instances
- Automated backup through snapshot scheduling
- Quick data recovery in case of data loss or failure
- Integration with AWS automation tools such as AWS CLI and Python

1.4) Amazon EBS Snapshots

Amazon EBS Snapshots are **point-in-time backups** of EBS volumes stored in Amazon S3. In this project:

- **Automated Creation:** Snapshots are created using **AWS CLI** and scheduled with **cron jobs** on the Command Host EC2 instance.
- **Retention Management:** A **Python script** (snapshotter_v2.py) deletes older snapshots, keeping only the latest two to save storage and costs.
- **Benefits:** Provides reliable backups, supports quick recovery, reduces manual effort, and ensures data durability for the Processor EC2 instance.

1.5) Amazon Simple Storage Service (S3)

Amazon S3 is a **scalable object storage service** that provides secure, durable, and highly available storage for data. In this project, S3 serves as the **long-term storage and backup solution** for files stored on the Processor EC2 instance's EBS volume.

a) Storing Synchronized Data

- Files from the EBS volume are **synchronized to an S3 bucket** using the **AWS CLI `s3 sync` command**.
- Ensures that all important files on the Processor instance are backed up in the cloud.

b) Versioning Enabled

- **S3 Versioning** is activated for the bucket to maintain multiple versions of files.
- Allows recovery of accidentally deleted or overwritten files by restoring previous versions.

c) File Recovery Support

- Deleted files can be recovered using their **version ID** with `aws s3api get-object`.
- Ensures data integrity and protects against accidental deletion.

d) Long-Term Backup Storage

- Acts as a **durable, off-instance backup** that is independent of EC2 and EBS.
- Provides **high availability and reliability**, as S3 automatically stores data across multiple facilities.

e) Benefits in the Project

- Automates file backup and synchronization from EBS to S3.

- Ensures **data durability and disaster recovery**.
- Integrates seamlessly with **AWS CLI and EC2 IAM roles** for secure access.
- Simplifies management by providing a **centralized storage solution**

1.6) IAM (Identity and Access Management)

AWS Identity and Access Management (IAM) is a service that allows secure control of access to AWS resources. In this project, IAM is used to provide **secure and controlled access** for the EC2 instances to perform storage operations without using hardcoded credentials.

a) IAM Roles for EC2 Instances

- **Processor EC2** and **Command Host EC2** are assigned **IAM roles** with specific permissions.
- These roles allow the instances to access AWS services such as:
  - **Amazon S3** (for file synchronization and retrieval)
  - **Amazon EBS** (for snapshot creation and management)

b) Secure, Temporary Credentials

- IAM roles provide **temporary security credentials** that are automatically rotated by AWS.
- This eliminates the need to store AWS access keys on EC2 instances, enhancing security.

c) Principle of Least Privilege

- Each role grants only the permissions needed for its tasks:
  - The Processor instance can read/write to S3 and manage its EBS volumes.
  - The Command Host instance can schedule snapshots and run automation scripts.
- This minimizes security risks and prevents unauthorized access to other AWS resources.

1.7) Automation Tools

Automation plays a key role in this project by reducing manual effort and ensuring consistent backup and recovery operations. The following tools are used to automate storage management tasks:

a) AWS Command Line Interface (AWS CLI)

- AWS CLI is used to interact with AWS services directly from the EC2 instances.
- It enables:
  - Creation of **EBS snapshots**.
  - Synchronization of files from **EBS to Amazon S3** using aws s3 sync.
  - Retrieval of deleted files from S3 using **object versioning**.
- AWS CLI allows faster execution of tasks compared to manual console operations.

b) Cron Jobs

- **Cron** is a Linux-based scheduling utility used to automate recurring tasks.
- In this project, cron jobs are configured to:
    - Automatically create **EBS snapshots at regular intervals**.
    - Run snapshot commands without human intervention.
- Ensures backups are taken consistently and on time.

c) Python Script (Snapshot Retention Management)

- A Python script is used to **control snapshot retention**.
- The script:
    - Lists all snapshots associated with an EBS volume.
    - Sorts snapshots based on creation time.
    - Deletes older snapshots and **retains only the most recent ones**.
- This prevents unnecessary storage usage and reduces backup costs.

Chapter 5

IMPLEMENTATION & RESULT

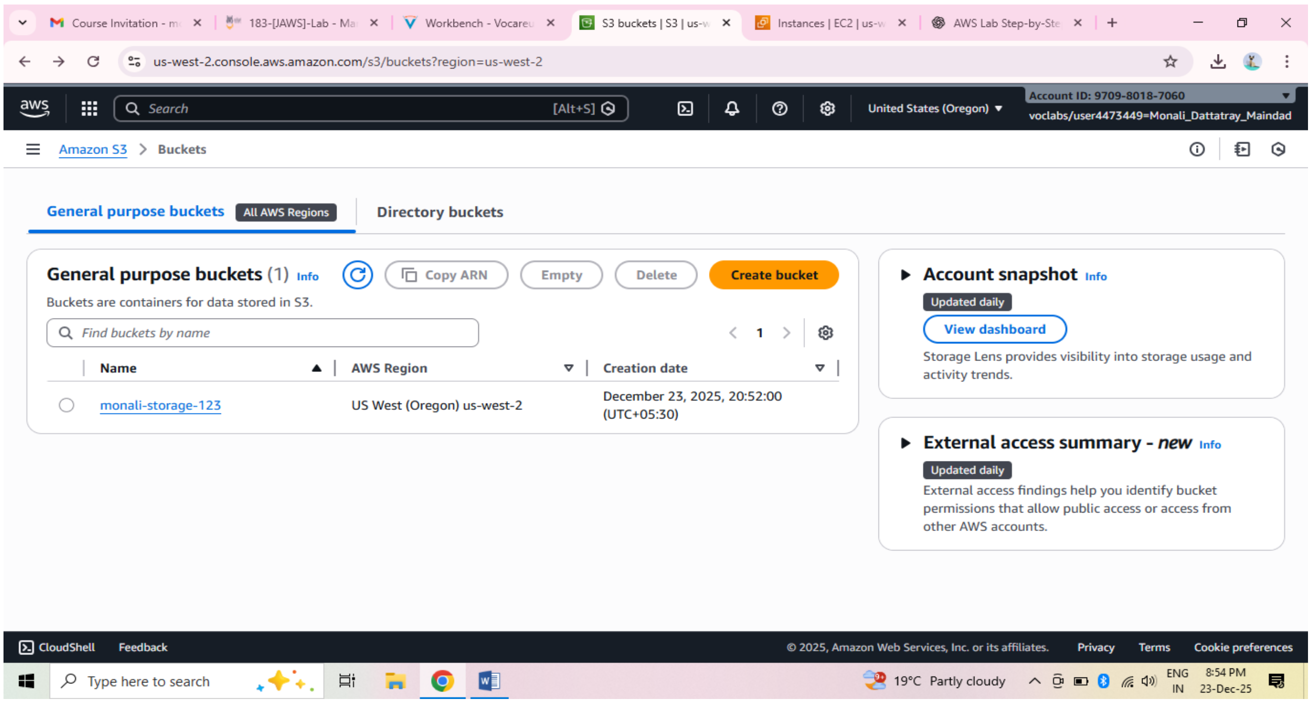
The project was implemented using AWS cloud services and automation tools to manage storage, backups, and data recovery efficiently. The implementation was carried out in multiple phases as described below:

5.1 CREATING AND CONFIGURING RESOURCES

a) Create an S3 bucket

In this task, you create an S3 bucket to sync files from an EBS volume

- On the console, choose **Create bucket**.
- In the **Create bucket** section, configure the following:
- **Bucket name**: Enter a bucket name. Use a combination of characters and numbers to keep it unique.
- This will be referred to as "s3-bucket-name" throughout the lab.
- **Region**: Leave as default.
- Scroll and choose **Create bucket**.



us-west-2.console.aws.amazon.com/s3/buckets?region=us-west-2

Account ID: 9709-8018-7060
voclabs/user4473449=Monali_Dattatray_Maindad

Amazon S3 > Buckets

General purpose buckets (1) Info

Buckets are containers for data stored in S3.

Name	AWS Region	Creation date
monali-storage-123	US West (Oregon) us-west-2	December 23, 2025, 20:52:00 (UTC+05:30)

Account snapshot Info
Updated daily
Storage Lens provides visibility into storage usage and activity trends.

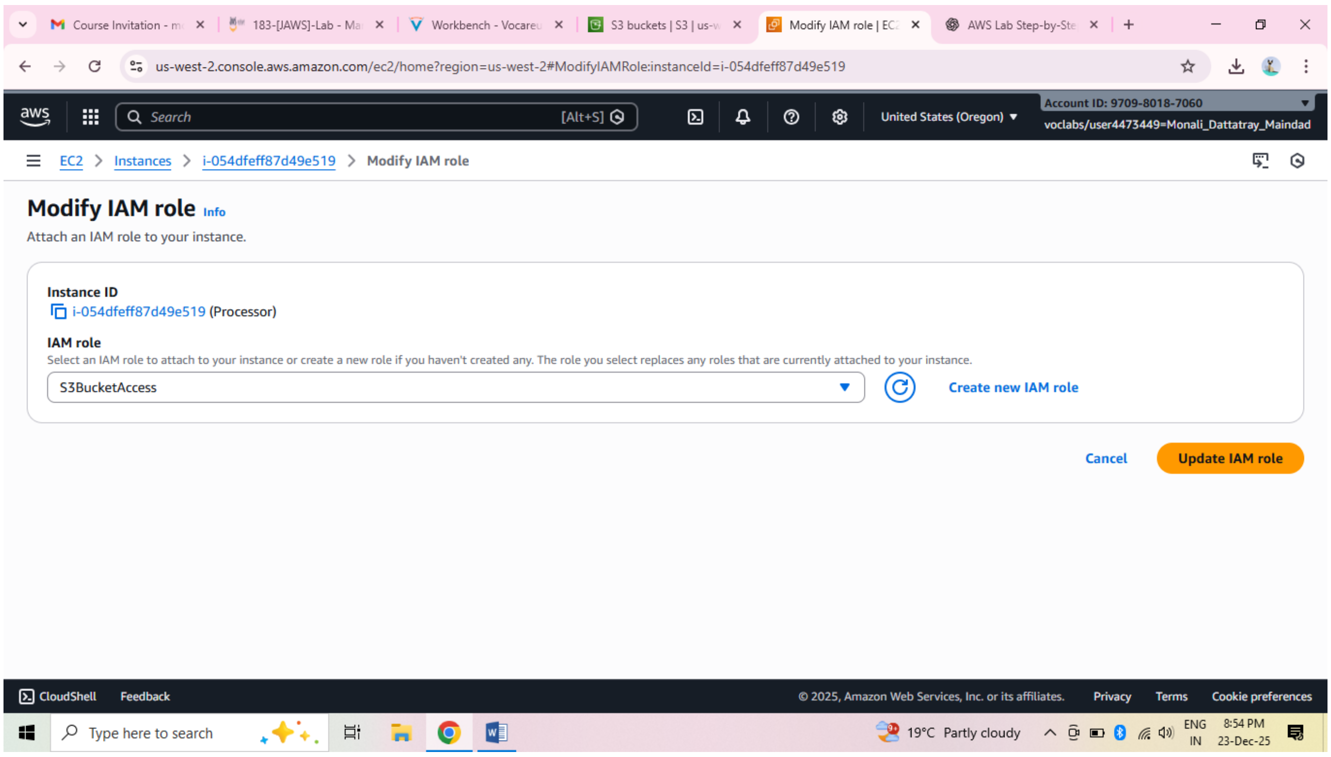
External access summary - new Info
Updated daily
External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.

© 2025, Amazon Web Services, Inc. or its affiliates.

b) Attach instance profile to Processor

In this task, you attach a pre-created IAM role as an instance profile to the EC2 instance "Processor," giving it the permissions to interact with other AWS services such as EBS volumes and S3 buckets.

- In the navigation pane, choose **Instances**.
- Choose **Processor** from the list of EC2 instances.
- Choose **Actions** > **Security** > **Modify IAM role**.
- Choose the S3BucketAccess role in the **IAM role** dropdown list.
- Choose **Update IAM role**.



Modify IAM role

Attach an IAM role to your instance.

Instance ID: i-054dfeff87d49e519 (Processor)

IAM role: Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

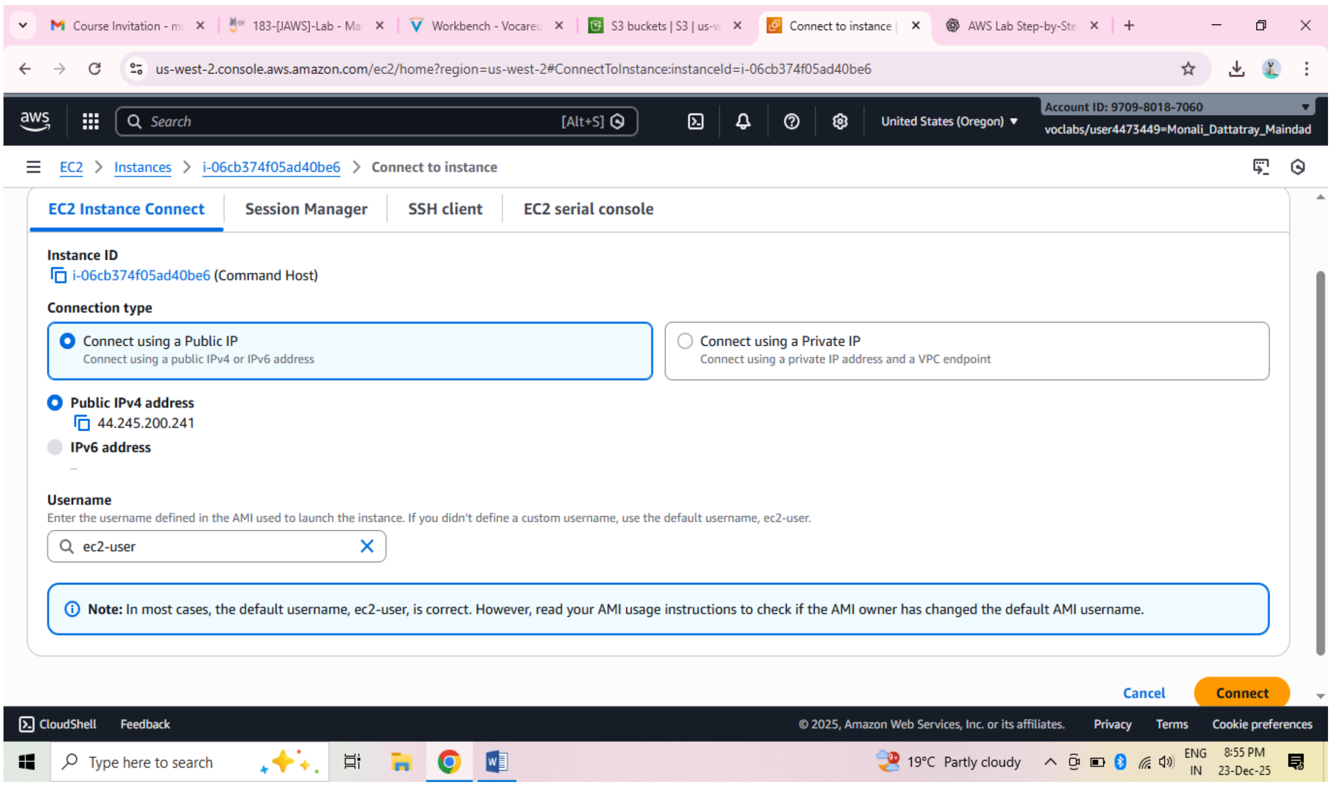
S3BucketAccess

5.2 TAKING SNAPSHOTS OF YOUR INSTANCE

In this section, you use the AWS Command Line Interface (AWS CLI) to manage the processing of snapshots of an instance.

a) Connecting to the Command Host EC2 instance

- In the navigation pane, choose **Instances**.
- From the list of instances, choose **Command Host**.
- Choose **Connect**.
- On the **EC2 Instance Connect** tab, choose **Connect**.



Instance ID
i-06cb374f05ad40be6 (Command Host)

Connection type
[✓ Connect using a Public IP — Connect using a public IPv4 or IPv6 address] Connect using a Private IP — Connect using a private IP address and a VPC endpoint

[✓ Public IPv4 address] 44.245.200.241
IPv6 address

Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.
ec2-user

Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

b) Taking an initial snapshot

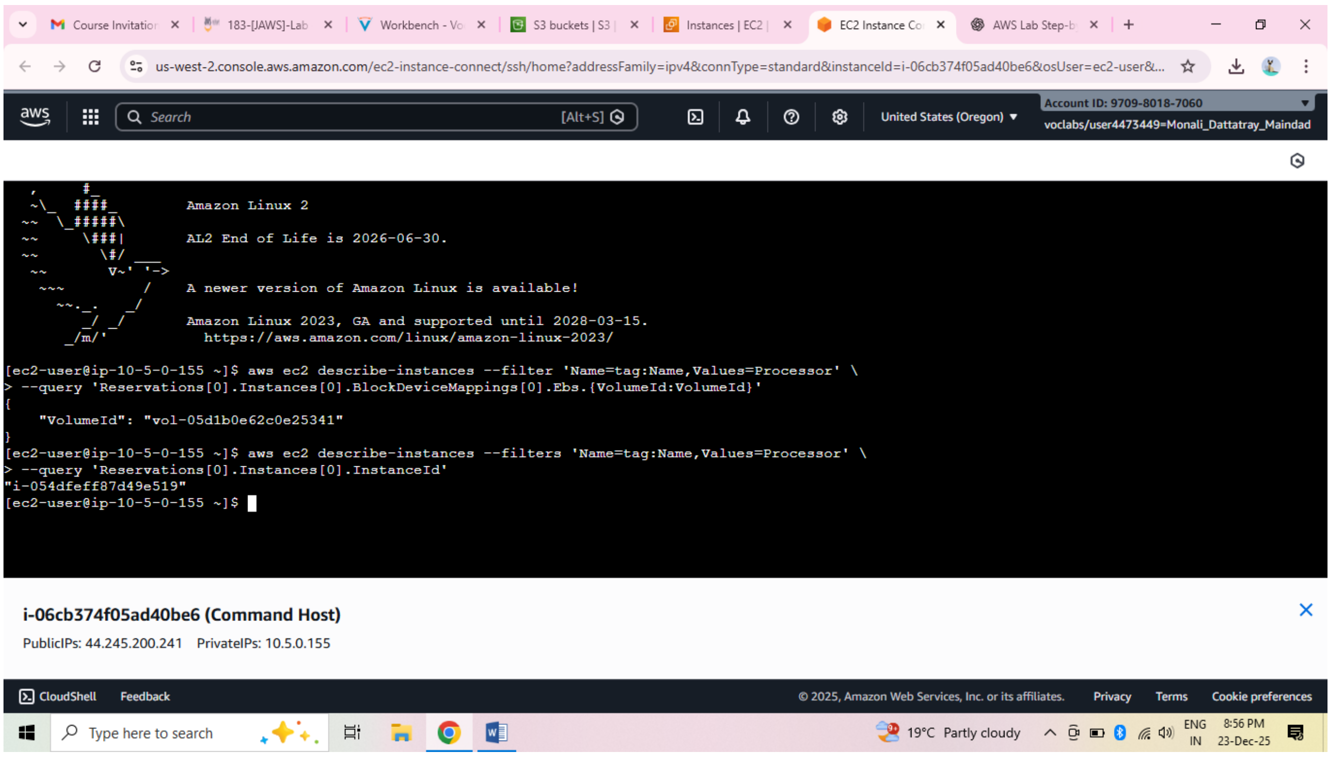
In this task, you identify the EBS volume that's attached to the "Processor" instance and take an initial snapshot. To do so, you run commands in the **EC2 Instance Connect terminal window**. You can copy the command output to a text editor for subsequent use.

- To display the EBS volume-id, run the following command:

```
aws ec2 describe-instances --filter 'Name=tag:Name,Values=Processor' --query
'Reservations[0].Instances[0].BlockDeviceMappings[0].Ebs.{VolumeId:VolumeId}'
```

- Next, you take snapshot of this volume. Prior to this, you shut down the "Processor" instance, which requires its instance ID. Run the following command to obtain the instance ID:

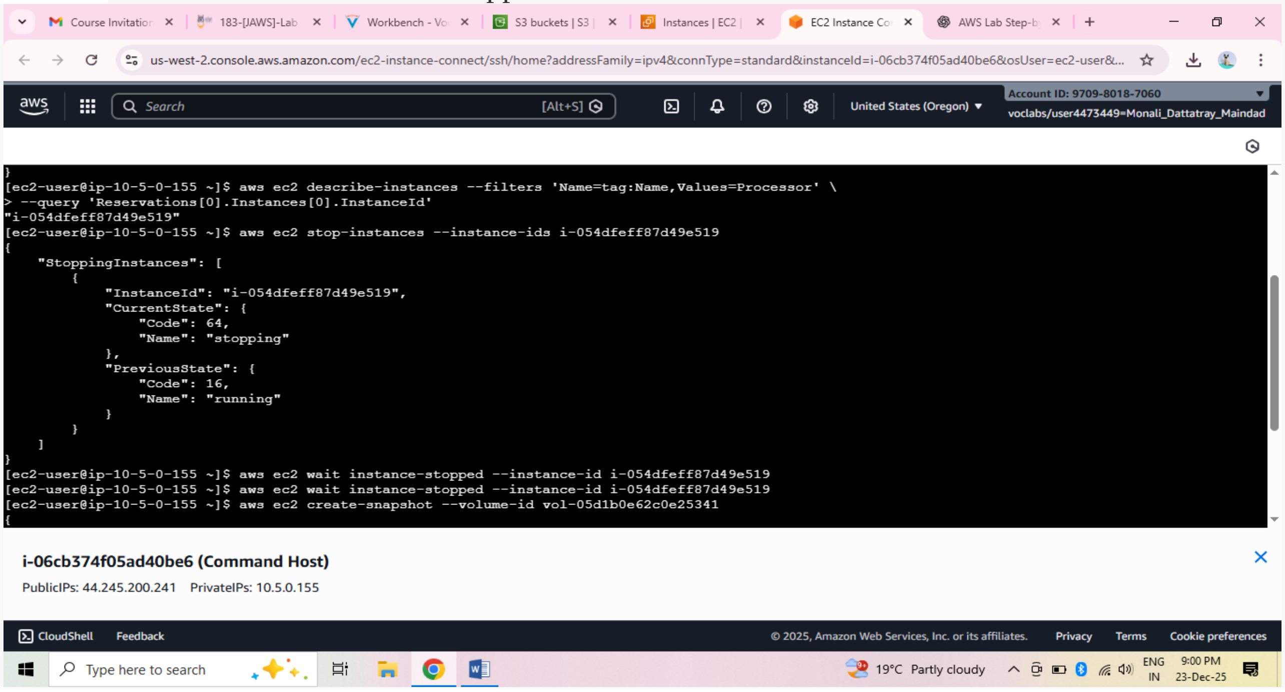
```
aws ec2 describe-instances --filters 'Name=tag:Name,Values=Processor' --query
'Reservations[0].Instances[0].InstanceId'
```


us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-06cb374f05ad40be6&osUser=ec2-user&...
Account ID: 9709-8018-7060
voclabs/user4473449=Monali_Dattatray_Maindad
United States (Oregon)

```
       #_
  ~\_  ####_        Amazon Linux 2
 ~~  \_#####\
 ~~     \###|       AL2 End of Life is 2026-06-30.
 ~~       \#/ ___
  ~~       V~' '->
   ~~~         /    A newer version of Amazon Linux is available!
     ~~._.   _/
        _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
      _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-10-5-0-155 ~]$ aws ec2 describe-instances --filter 'Name=tag:Name,Values=Processor' \
> --query 'Reservations[0].Instances[0].BlockDeviceMappings[0].Ebs.{VolumeId:VolumeId}'
{
    "VolumeId": "vol-05d1b0e62c0e25341"
}
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 describe-instances --filters 'Name=tag:Name,Values=Processor' \
> --query 'Reservations[0].Instances[0].InstanceId'
"i-054dfeff87d49e519"
[ec2-user@ip-10-5-0-155 ~]$
```

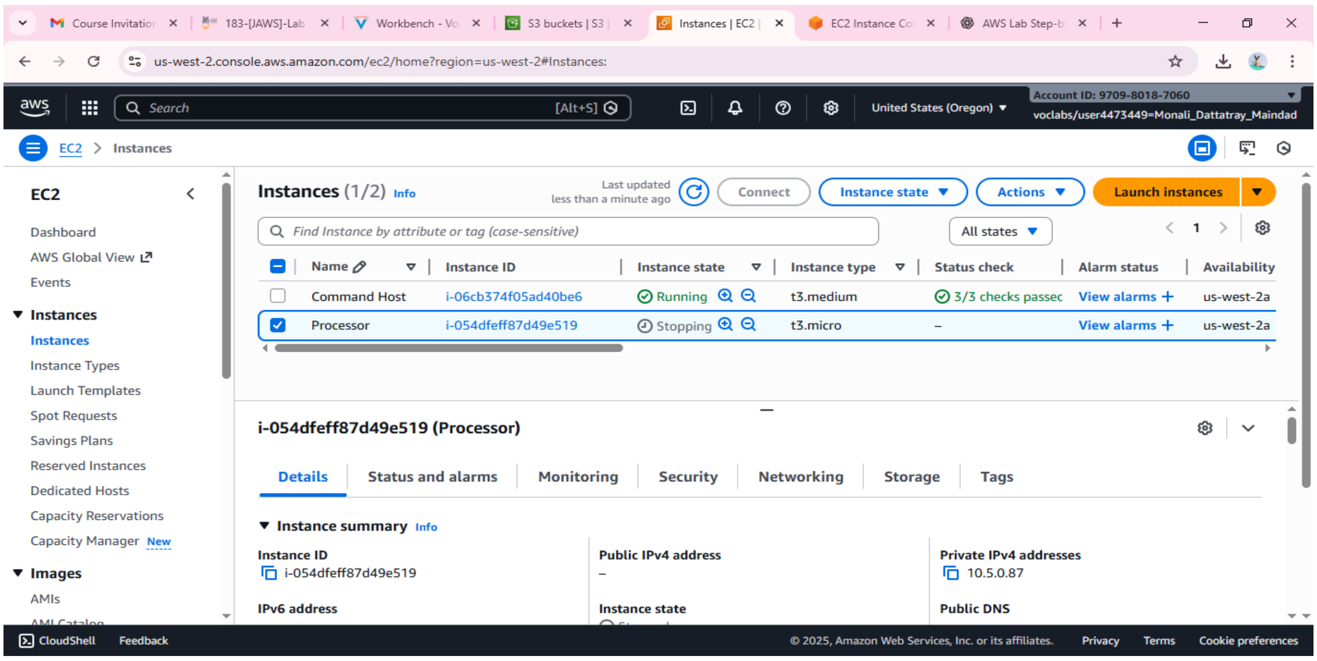
i-06cb374f05ad40be6 (Command Host)
PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155

- To shut down the "Processor" instance, run the following command and replace "INSTANCE-ID" with the instance-id that you retrieved earlier:
  - aws ec2 stop-instances --instance-ids INSTANCE-ID
- To verify that the "Processor" instance stopped, run the following command and replace "INSTANCE-ID" with your instance id.
  - aws ec2 wait instance-stopped --instance-id INSTANCE-ID


us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-06cb374f05ad40be6&osUser=ec2-user&...
Account ID: 9709-8018-7060
voclabs/user4473449=Monali_Dattatray_Maindad
United States (Oregon)

```
}
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 describe-instances --filters 'Name=tag:Name,Values=Processor' \
> --query 'Reservations[0].Instances[0].InstanceId'
"i-054dfeff87d49e519"
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 stop-instances --instance-ids i-054dfeff87d49e519
{
    "StoppingInstances": [
        {
            "InstanceId": "i-054dfeff87d49e519",
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 wait instance-stopped --instance-id i-054dfeff87d49e519
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 wait instance-stopped --instance-id i-054dfeff87d49e519
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 create-snapshot --volume-id vol-05d1b0e62c0e25341
{
```

i-06cb374f05ad40be6 (Command Host)
PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
☐	Command Host	i-06cb374f05ad40be6	Running	t3.medium	3/3 checks passec	View alarms +	us-west-2a
☑	Processor	i-054dfeff87d49e519	Stopping	t3.micro	–	View alarms +	us-west-2a

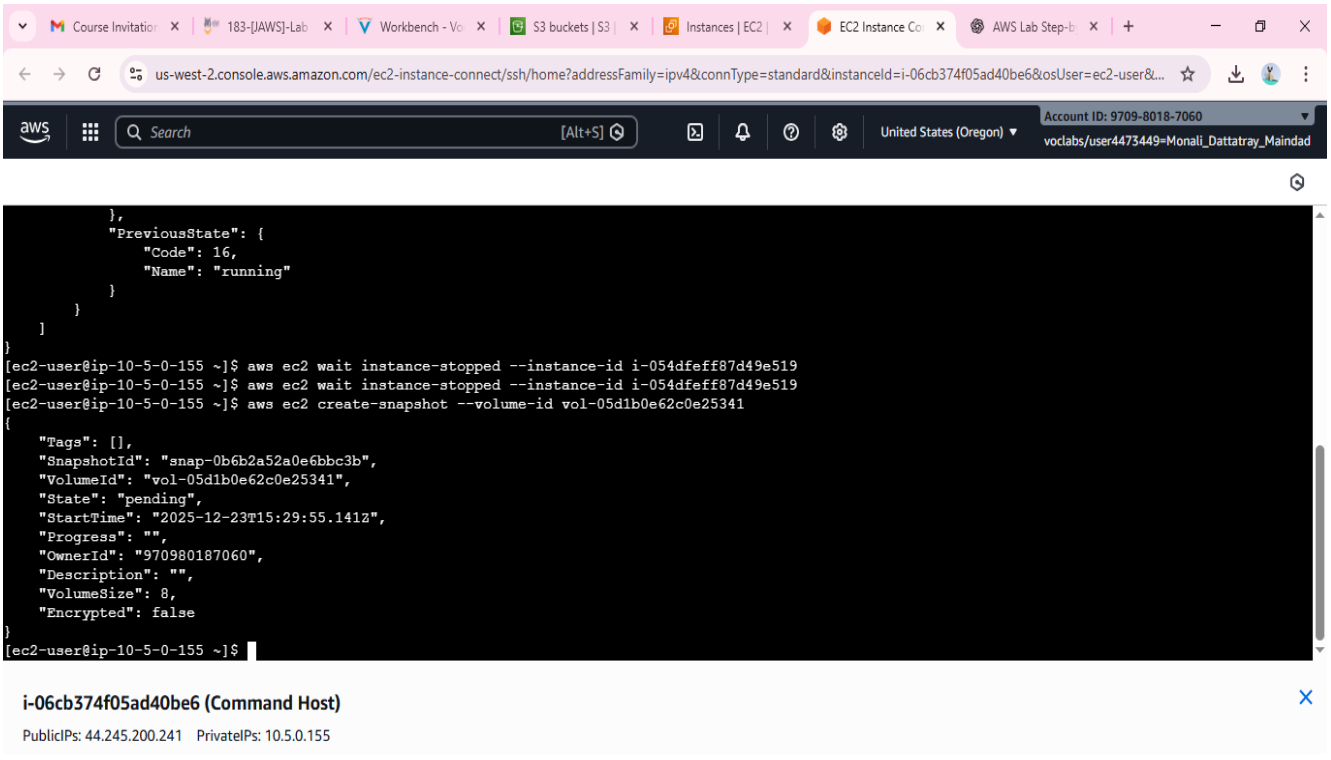
i-054dfeff87d49e519 (Processor)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-054dfeff87d49e519	–	10.5.0.87
IPv6 address	Instance state	Public DNS

- To create your first snapshot of the volume of your "Processor" instance, run the following command and replace "VOLUME-ID" with the VolumeId that you retrieved earlier:
  - aws ec2 create-snapshot --volume-id VOLUME-ID
- To check the status of your snapshot, run the following command and replace "SNAPSHOT-ID" with the SnapshotId that you retrieved earlier:
  - aws ec2 wait snapshot-completed --snapshot-id SNAPSHOT-ID



```
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 wait instance-stopped --instance-id i-054dfeff87d49e519
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 wait instance-stopped --instance-id i-054dfeff87d49e519
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 create-snapshot --volume-id vol-05d1b0e62c0e25341
{
    "Tags": [],
    "SnapshotId": "snap-0b6b2a52a0e6bbc3b",
    "VolumeId": "vol-05d1b0e62c0e25341",
    "State": "pending",
    "StartTime": "2025-12-23T15:29:55.141Z",
    "Progress": "",
    "OwnerId": "970980187060",
    "Description": "",
    "VolumeSize": 8,
    "Encrypted": false
}
[ec2-user@ip-10-5-0-155 ~]$
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155

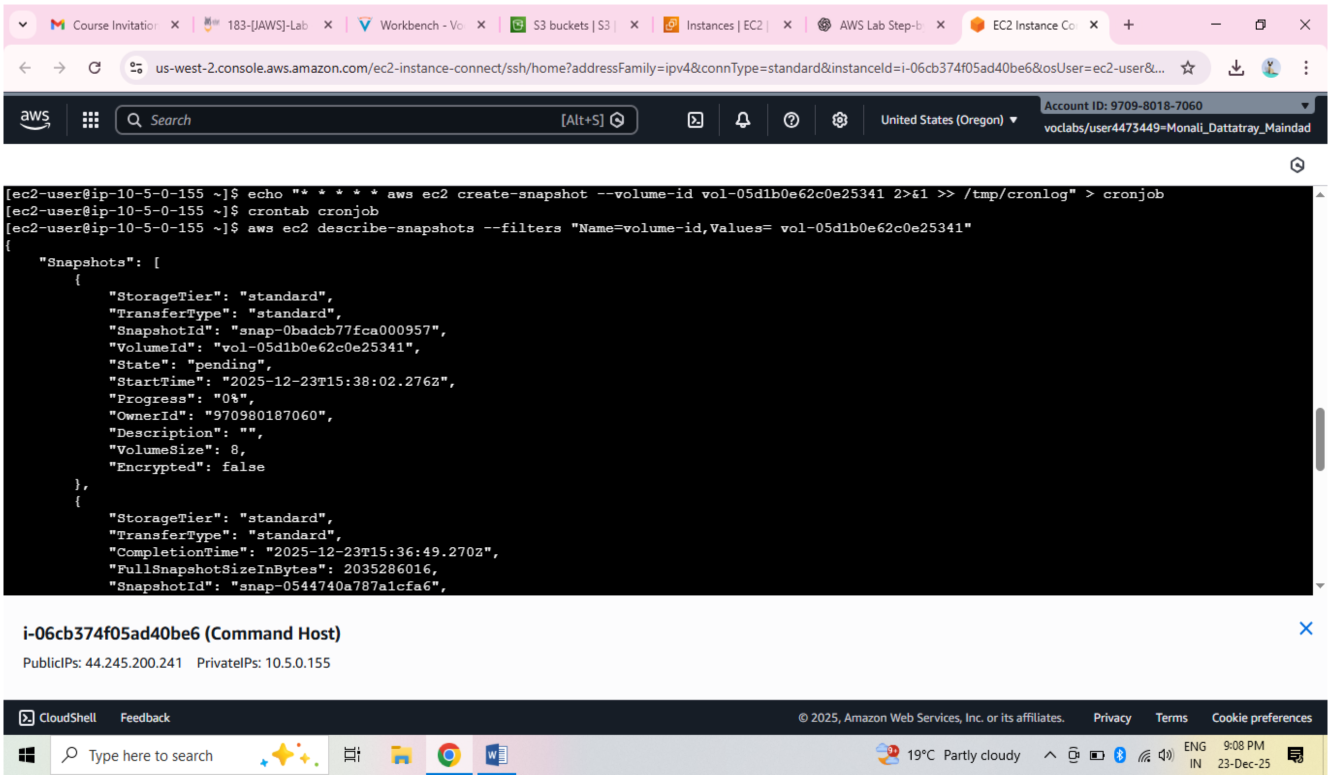
- To restart the "Processor" instance, run the following command and replace "INSTANCE-ID" with the instance-id that you retrieved earlier:
  - aws ec2 start-instances --instance-ids INSTANCE-ID

c) Scheduling the creation of subsequent snapshots

Using the Linux scheduling system (cron), you can set up a recurring snapshot process so that new snapshots of your data are taken automatically.

In this task, you create a cron job to manage the number of snapshots that are maintained for a volume.

- To create and schedule a cron entry that runs a job every minute, run the following command and replace "VOLUME-ID" with the VolumeId that you retrieved earlier:
  - echo "* * * * * aws ec2 create-snapshot --volume-id VOLUME-ID 2>&1 >> /tmp/cronlog" >
  - cronjob
  - crontab cronjob
- To verify that subsequent snapshots are being created, run the following command and replace "VOLUME-ID" with the VolumeId that you retrieved earlier:
  - aws ec2 describe-snapshots --filters "Name=volume-id,Values=VOLUME-ID"
- Wait a few minutes so that a few more snapshots are generated before beginning the next task.

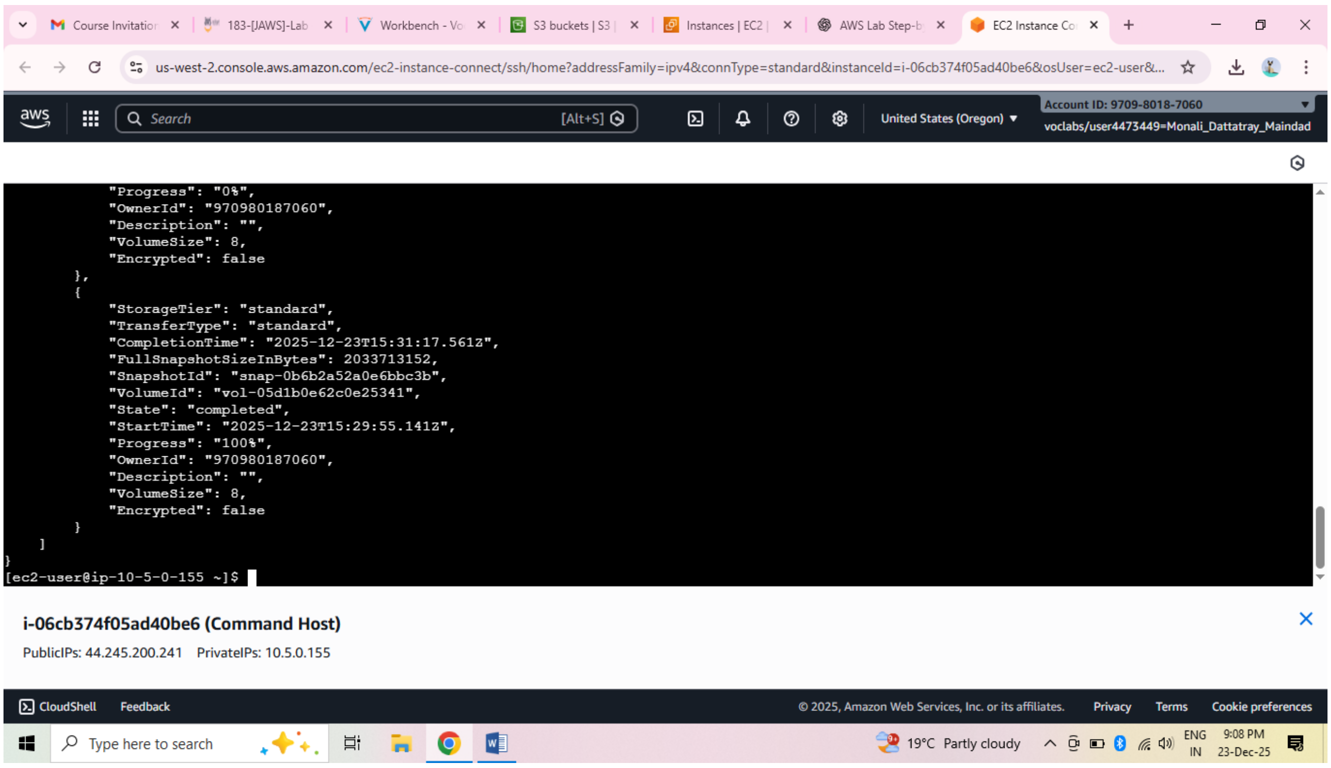


```
[ec2-user@ip-10-5-0-155 ~]$ echo "* * * * * aws ec2 create-snapshot --volume-id vol-05d1b0e62c0e25341 2>&1 >> /tmp/cronlog" > cronjob
[ec2-user@ip-10-5-0-155 ~]$ crontab cronjob
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 describe-snapshots --filters "Name=volume-id,Values= vol-05d1b0e62c0e25341"
{
    "Snapshots": [
        {
            "StorageTier": "standard",
            "TransferType": "standard",
            "SnapshotId": "snap-0badcb77fca000957",
            "VolumeId": "vol-05d1b0e62c0e25341",
            "State": "pending",
            "StartTime": "2025-12-23T15:38:02.276Z",
            "Progress": "0%",
            "OwnerId": "970980187060",
            "Description": "",
            "VolumeSize": 8,
            "Encrypted": false
        },
        {
            "StorageTier": "standard",
            "TransferType": "standard",
            "CompletionTime": "2025-12-23T15:36:49.270Z",
            "FullSnapshotSizeInBytes": 2035286016,
            "SnapshotId": "snap-0544740a787a1cfa6",
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155

us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-06cb374f05ad40be6&osUser=ec2-user&...
Account ID: 9709-8018-7060
United States (Oregon)
voclabs/user4473449=Monali_Dattatray_Maindad
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
19°C Partly cloudy
9:08 PM
23-Dec-25


Course Invitation
183-[AWS]-Lab
Workbench - Vo
S3 buckets | S3
Instances | EC2
AWS Lab Step-b
EC2 Instance Co
us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-06cb374f05ad40be6&osUser=ec2-user&...
Search
[Alt+S]
United States (Oregon)
Account ID: 9709-8018-7060
voclabs/user4473449=Monali_Dattatray_Maindad

```
            "Progress": "0%",
            "OwnerId": "970980187060",
            "Description": "",
            "VolumeSize": 8,
            "Encrypted": false
        },
        {
            "StorageTier": "standard",
            "TransferType": "standard",
            "CompletionTime": "2025-12-23T15:31:17.561Z",
            "FullSnapshotSizeInBytes": 2033713152,
            "SnapshotId": "snap-0b6b2a52a0e6bbc3b",
            "VolumeId": "vol-05d1b0e62c0e25341",
            "State": "completed",
            "StartTime": "2025-12-23T15:29:55.141Z",
            "Progress": "100%",
            "OwnerId": "970980187060",
            "Description": "",
            "VolumeSize": 8,
            "Encrypted": false
        }
    ]
}
[ec2-user@ip-10-5-0-155 ~]$
```

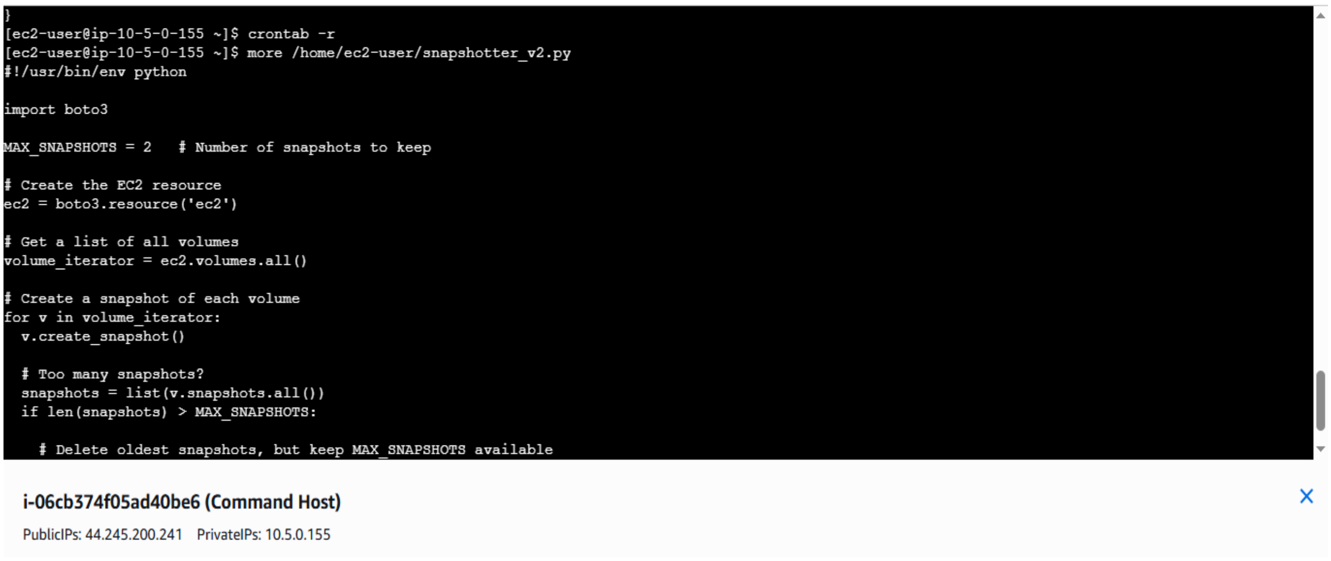
i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241   PrivateIPs: 10.5.0.155

d) Retaining the last two snapshots

In this task, you run a Python script that maintains only the last two snapshots for any given EBS volume.

- To stop the cron job, run the following command:
    - crontab -r
- To examine the contents of the Python script "snapshotter_v2.py", run the following command:
    - more /home/ec2-user/snapshotter_v2.py



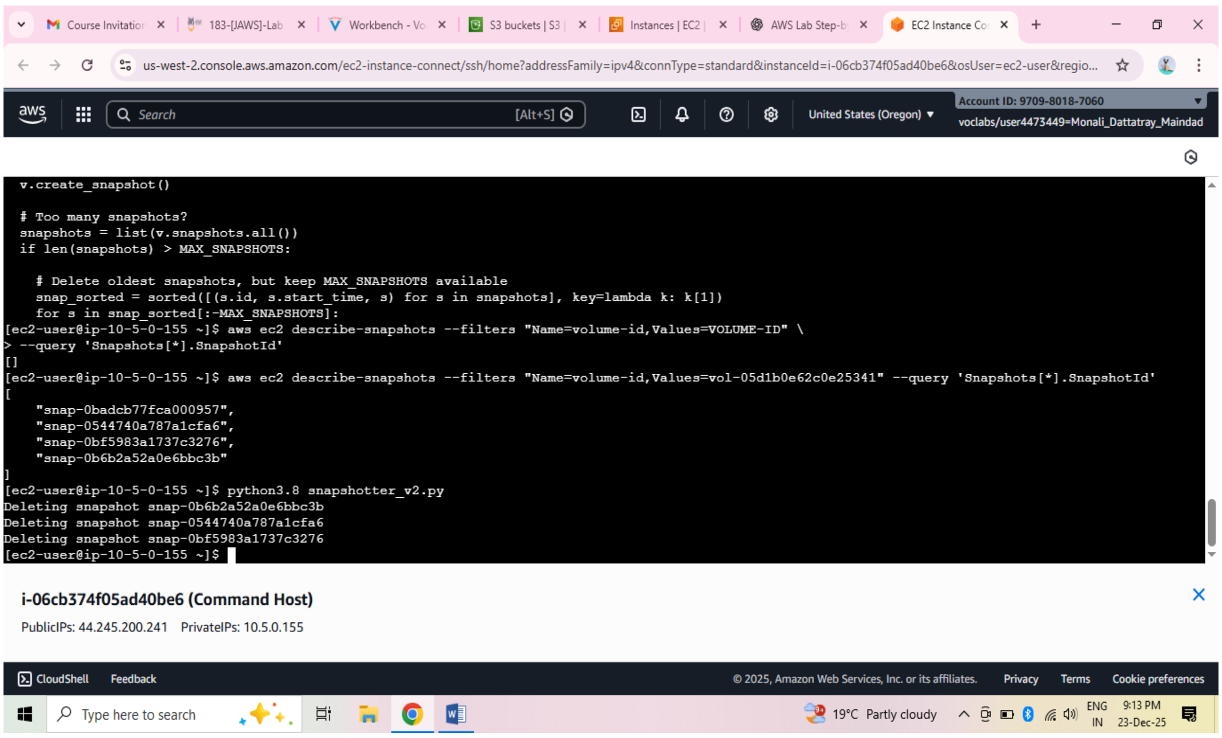
```
}
[ec2-user@ip-10-5-0-155 ~]$ crontab -r
[ec2-user@ip-10-5-0-155 ~]$ more /home/ec2-user/snapshotter_v2.py
#!/usr/bin/env python

import boto3

MAX_SNAPSHOTS = 2   # Number of snapshots to keep

# Create the EC2 resource
ec2 = boto3.resource('ec2')

# Get a list of all volumes
volume_iterator = ec2.volumes.all()

# Create a snapshot of each volume
for v in volume_iterator:
  v.create_snapshot()

  # Too many snapshots?
  snapshots = list(v.snapshots.all())
  if len(snapshots) > MAX_SNAPSHOTS:

    # Delete oldest snapshots, but keep MAX_SNAPSHOTS available
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241   PrivateIPs: 10.5.0.155

The script finds all EBS volumes that are associated with the current user's account and takes snapshots. It then examines the number of snapshots that are associated with the volume, sorts the snapshots by date, and removes all but the two most recent snapshots.

- Before running snapshotter_v2.py, run the following command and replace "VOLUME-ID" with the VolumeId that you retrieved earlier:
    - aws ec2 describe-snapshots --filters "Name=volume-id, Values=VOLUME-ID" --query 'Snapshots[*].SnapshotId'



```
  v.create_snapshot()

  # Too many snapshots?
  snapshots = list(v.snapshots.all())
  if len(snapshots) > MAX_SNAPSHOTS:

    # Delete oldest snapshots, but keep MAX_SNAPSHOTS available
    snap_sorted = sorted([(s.id, s.start_time, s) for s in snapshots], key=lambda k: k[1])
    for s in snap_sorted[:-MAX_SNAPSHOTS]:
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 describe-snapshots --filters "Name=volume-id,Values=VOLUME-ID" \
> --query 'Snapshots[*].SnapshotId'
[]
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 describe-snapshots --filters "Name=volume-id,Values=vol-05d1b0e62c0e25341" --query 'Snapshots[*].SnapshotId'
[
    "snap-0badcb77fca000957",
    "snap-0544740a787a1cfa6",
    "snap-0bf5983a1737c3276",
    "snap-0b6b2a52a0e6bbc3b"
]
[ec2-user@ip-10-5-0-155 ~]$ python3.8 snapshotter_v2.py
Deleting snapshot snap-0b6b2a52a0e6bbc3b
Deleting snapshot snap-0544740a787a1cfa6
Deleting snapshot snap-0bf5983a1737c3276
[ec2-user@ip-10-5-0-155 ~]$
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155

us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-06cb374f05ad40be6&osUser=ec2-user&regio...
Account ID: 9709-8018-7060
voclabs/user4473449=Monali_Dattatray_Maindad
United States (Oregon)
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
19°C Partly cloudy
9:13 PM
23-Dec-25

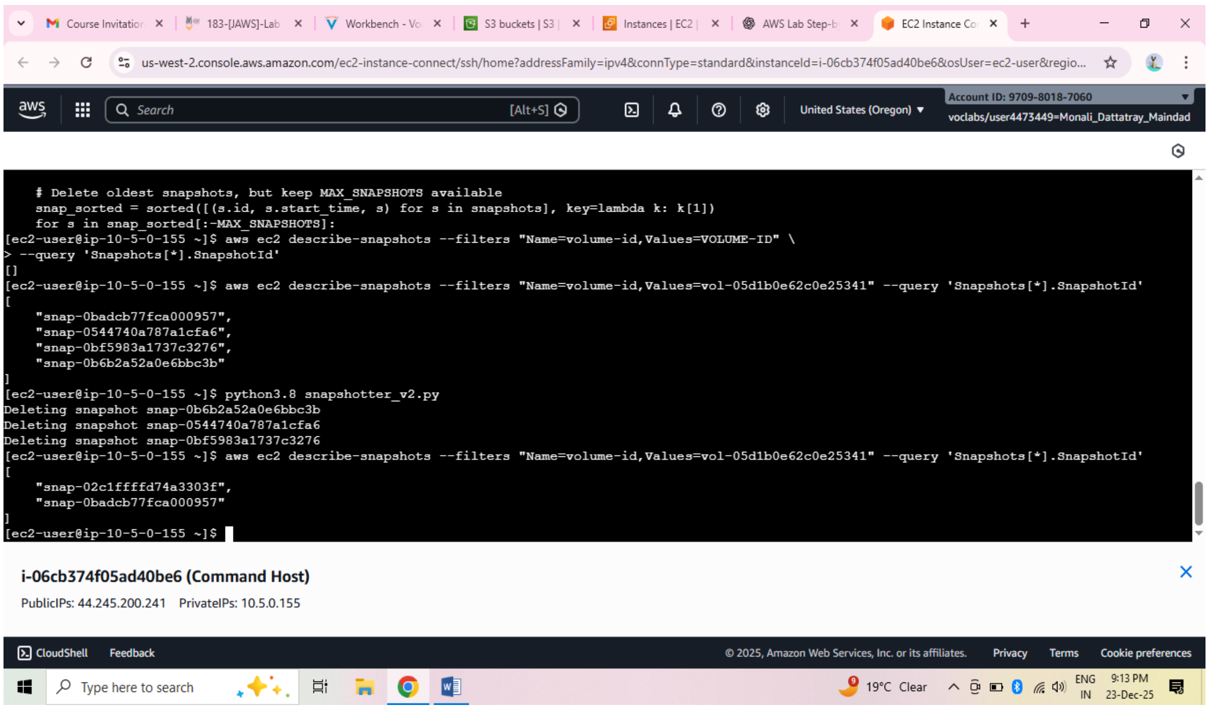
The command returns the multiple snapshot IDs that were returned for the volume. These are the snapshots that were created by your cron job before you stopped it.

- Run the the "snapshotter_v2.py" script using following command:
    - python3.8 snapshotter_v2.py

The script runs for a few seconds, and then it returns a list of all of the snapshots that it deleted:

- To examine the new number of snapshots for the current volume, re-run the following command from an earlier step:
    - aws ec2 describe-snapshots --filters "Name=volume-id, Values=VOLUME-ID" --query 'Snapshots[*].SnapshotId'

The command returns only **two** snapshot IDs.



```
    # Delete oldest snapshots, but keep MAX_SNAPSHOTS available
    snap_sorted = sorted([(s.id, s.start_time, s) for s in snapshots], key=lambda k: k[1])
    for s in snap_sorted[:-MAX_SNAPSHOTS]:
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 describe-snapshots --filters "Name=volume-id,Values=VOLUME-ID" \
> --query 'Snapshots[*].SnapshotId'
[]
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 describe-snapshots --filters "Name=volume-id,Values=vol-05d1b0e62c0e25341" --query 'Snapshots[*].SnapshotId'
[
    "snap-0badcb77fca000957",
    "snap-0544740a787a1cfa6",
    "snap-0bf5983a1737c3276",
    "snap-0b6b2a52a0e6bbc3b"
]
[ec2-user@ip-10-5-0-155 ~]$ python3.8 snapshotter_v2.py
Deleting snapshot snap-0b6b2a52a0e6bbc3b
Deleting snapshot snap-0544740a787a1cfa6
Deleting snapshot snap-0bf5983a1737c3276
[ec2-user@ip-10-5-0-155 ~]$ aws ec2 describe-snapshots --filters "Name=volume-id,Values=vol-05d1b0e62c0e25341" --query 'Snapshots[*].SnapshotId'
[
    "snap-02c1ffffd74a3303f",
    "snap-0badcb77fca000957"
]
[ec2-user@ip-10-5-0-155 ~]$
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155

5.3 CHALLENGE: SYNCHRONIZE FILES WITH AMAZON S3

In this task, you are challenged to sync the contents of a directory with the Amazon S3 bucket that you created earlier.

- Run the following command in the terminal to download a sample set of files:

```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-100-RSJAWS-3-124627/183-lab-JAWS-managing-storage/s3/files.zip
```

- Activate versioning for your Amazon S3 bucket.
- Use a single AWS CLI command to sync the contents of your unzipped folder with your Amazon S3 bucket.
- Modify the command so that it deletes a file from Amazon S3 when the corresponding file is deleted locally on your instance.
- Recover the deleted file from Amazon S3 using versioning.

a) Downloading and unzipping sample files

The sample file package contains a folder with three text files: file1.txt, file2.txt, and file3.txt. These are the files that you will sync with your Amazon S3 bucket.

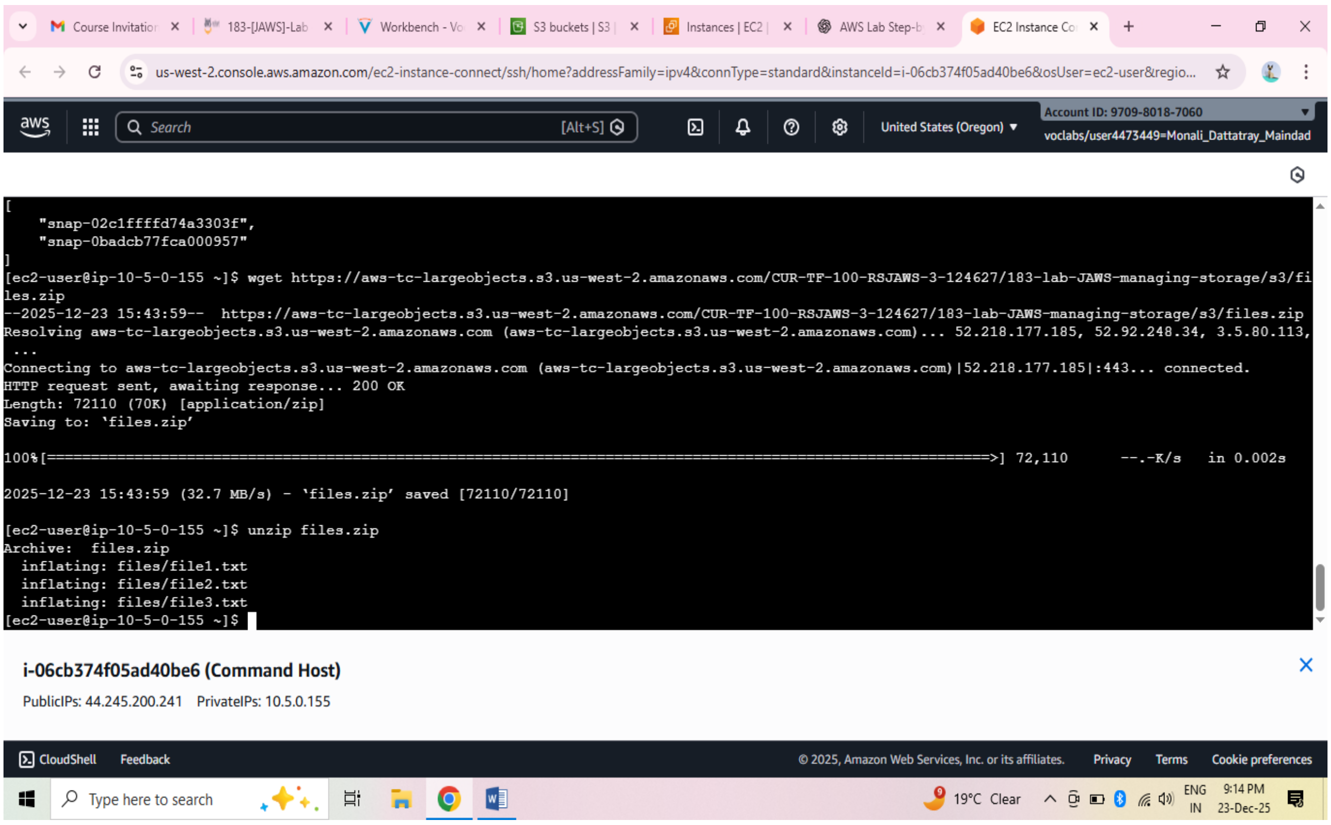
- Connect to the "Processor" instance using EC2 Instance Connect.

Note: Refer to the earlier steps that you used to connect to the "Command Host" instance.

You run following AWS CLI commands in the EC2 Instance Connect terminal window.

- To download the sample files on the "Processor" instance, run the following command from within your instance:
    - wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-100-RSJAWS-3-124627/183-lab-JAWS-managing-storage/s3/files.zip

- To unzip the directory, use the following command:
    - unzip files.zip



```
[
    "snap-02c1ffffd74a3303f",
    "snap-0badcb77fca000957"
]
[ec2-user@ip-10-5-0-155 ~]$ wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-100-RSJAWS-3-124627/183-lab-JAWS-managing-storage/s3/fi
les.zip
--2025-12-23 15:43:59--  https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-100-RSJAWS-3-124627/183-lab-JAWS-managing-storage/s3/files.zip
Resolving aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)... 52.218.177.185, 52.92.248.34, 3.5.80.113,
 ...
Connecting to aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)|52.218.177.185|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 72110 (70K) [application/zip]
Saving to: 'files.zip'

100%[====================================================================================================>] 72,110      --.-K/s   in 0.002s

2025-12-23 15:43:59 (32.7 MB/s) - 'files.zip' saved [72110/72110]

[ec2-user@ip-10-5-0-155 ~]$ unzip files.zip
Archive:  files.zip
  inflating: files/file1.txt
  inflating: files/file2.txt
  inflating: files/file3.txt
[ec2-user@ip-10-5-0-155 ~]$
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241   PrivateIPs: 10.5.0.155

us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-06cb374f05ad40be6&osUser=ec2-user&regio...
Account ID: 9709-8018-7060
United States (Oregon)
voclabs/user4473449=Monali_Dattatray_Maindad
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

b) Syncing files

Before syncing content with your Amazon S3 bucket, you need to activate versioning on your bucket.

- Run the following command and replace "S3-BUCKET-NAME" with your bucket name:
    - aws s3api put-bucket-versioning --bucket S3-BUCKET-NAME --versioning-configuration Status=Enabled

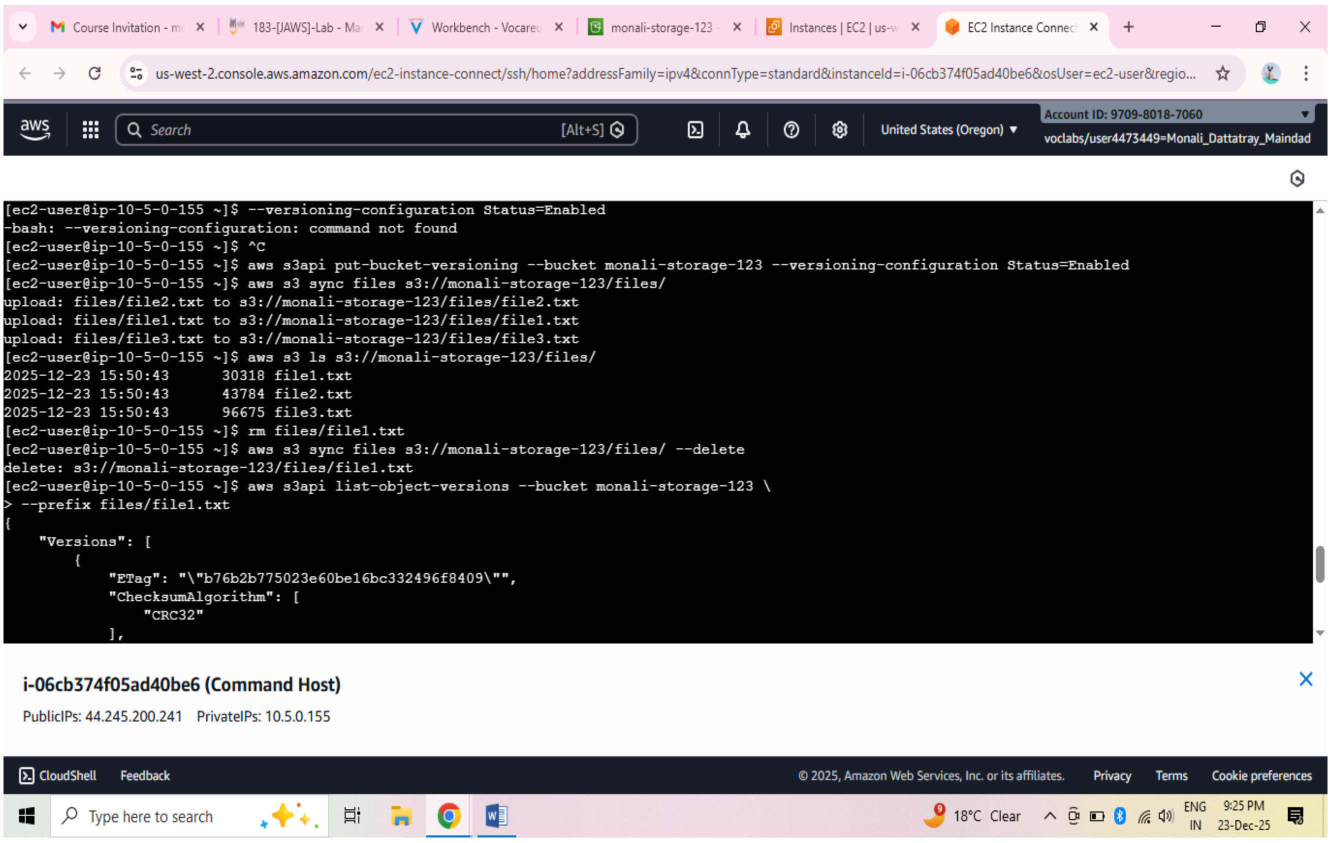
- To sync the contents of the files folder with your Amazon S3 bucket, run the following command and replace "S3-BUCKET-NAME" with your bucket name:
  - aws s3 sync files s3://S3-BUCKET-NAME/files/

  The command confirms that three files were uploaded to your S3 bucket.

- To confirm the state of your files, run the following command and replace "S3-BUCKET-NAME" with your bucket name:
  - aws s3 ls s3://S3-BUCKET-NAME/files/
- To delete one of the files on the local drive, run the following command:
  - rm files/file1.txt

  To delete the same file from the S3 bucket, use the `--delete` option with the `aws s3 sync` command.

- Run the following command and replace "S3-BUCKET-NAME" with your bucket name:
  - aws s3 sync files s3://S3-BUCKET-NAME/files/ --delete


Course Invitation - m
183-[JAWS]-Lab - Ma
Workbench - Vocare
monali-storage-123
Instances | EC2 | us-w
EC2 Instance Connect
us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-06cb374f05ad40be6&osUser=ec2-user&regio...
Search
[Alt+S]
United States (Oregon)
Account ID: 9709-8018-7060
voclabs/user4473449=Monali_Dattatray_Maindad

```
[ec2-user@ip-10-5-0-155 ~]$ --versioning-configuration Status=Enabled
-bash: --versioning-configuration: command not found
[ec2-user@ip-10-5-0-155 ~]$ ^C
[ec2-user@ip-10-5-0-155 ~]$ aws s3api put-bucket-versioning --bucket monali-storage-123 --versioning-configuration Status=Enabled
[ec2-user@ip-10-5-0-155 ~]$ aws s3 sync files s3://monali-storage-123/files/
upload: files/file2.txt to s3://monali-storage-123/files/file2.txt
upload: files/file1.txt to s3://monali-storage-123/files/file1.txt
upload: files/file3.txt to s3://monali-storage-123/files/file3.txt
[ec2-user@ip-10-5-0-155 ~]$ aws s3 ls s3://monali-storage-123/files/
2025-12-23 15:50:43      30318 file1.txt
2025-12-23 15:50:43      43784 file2.txt
2025-12-23 15:50:43      96675 file3.txt
[ec2-user@ip-10-5-0-155 ~]$ rm files/file1.txt
[ec2-user@ip-10-5-0-155 ~]$ aws s3 sync files s3://monali-storage-123/files/ --delete
delete: s3://monali-storage-123/files/file1.txt
[ec2-user@ip-10-5-0-155 ~]$ aws s3api list-object-versions --bucket monali-storage-123 \
> --prefix files/file1.txt
{
    "Versions": [
        {
            "ETag": "\"b76b2b775023e60be16bc332496f8409\"",
            "CheckSumAlgorithm": [
                "CRC32"
            ],
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155

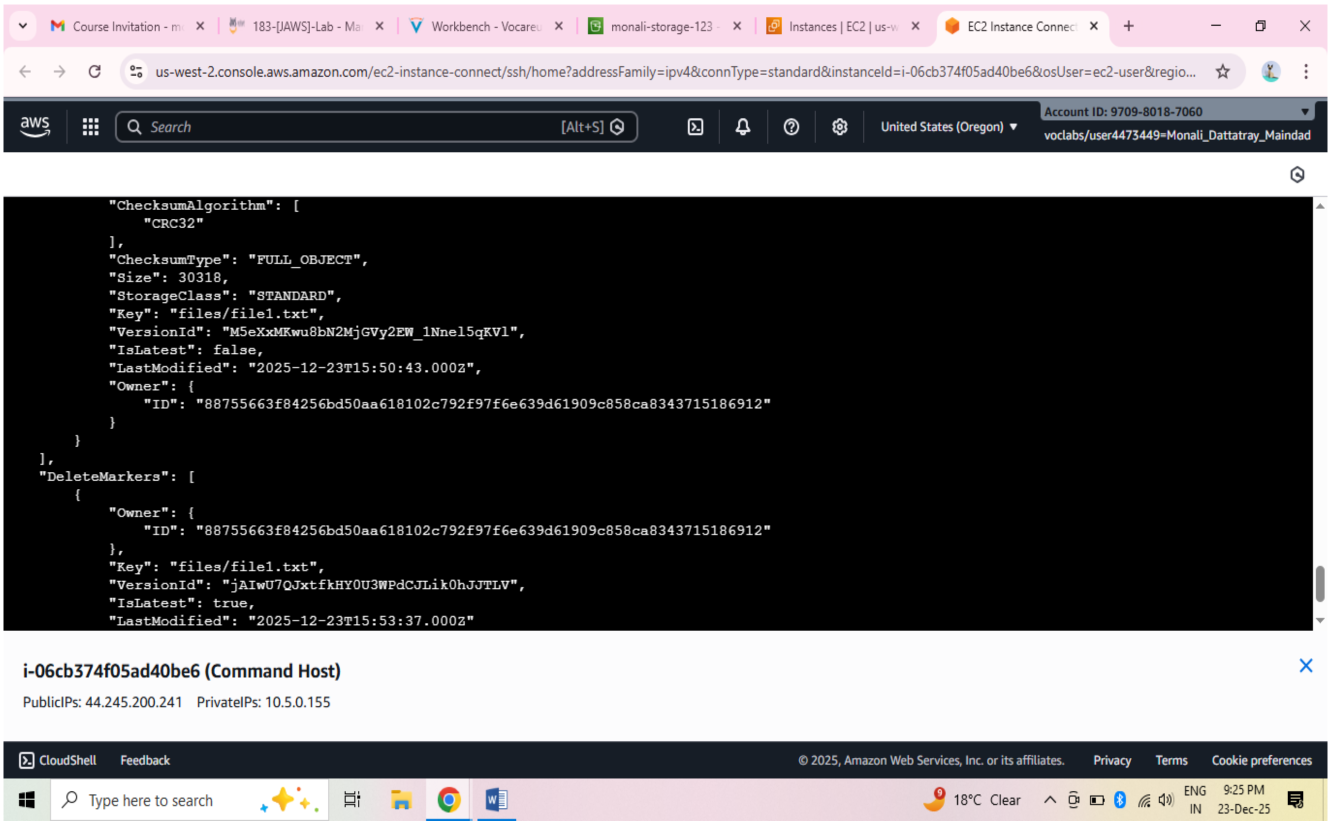
- To verify that the file was deleted from the bucket, run the following command and replace "S3-BUCKET-NAME" with your bucket name:

- aws s3 ls s3://S3-BUCKET-NAME/files/

- Now, try to recover the old version of file1.txt. To view a list of previous versions of this file, run the following command and replace "S3-BUCKET-NAME" with your bucket name:
  - aws s3api list-object-versions --bucket S3-BUCKET-NAME --prefix files/file1.txt

The output contains a `DeleteMarkers` and a `Versions` block. `DeleteMarkers` indicates where the delete marker is. For example, if you perform an `aws s3 rm` operation (or an `aws s3 sync` operation with the `--delete` option), this is the next version that the file will revert to.The `Versions` block contains a list of all available versions. You should have only a single versions entry. Note the value for `VersionId` for use later.Because there's no direct command to restore an older version of an Amazon S3 object to its own bucket, you need to re-download the old version and sync again to Amazon S3.

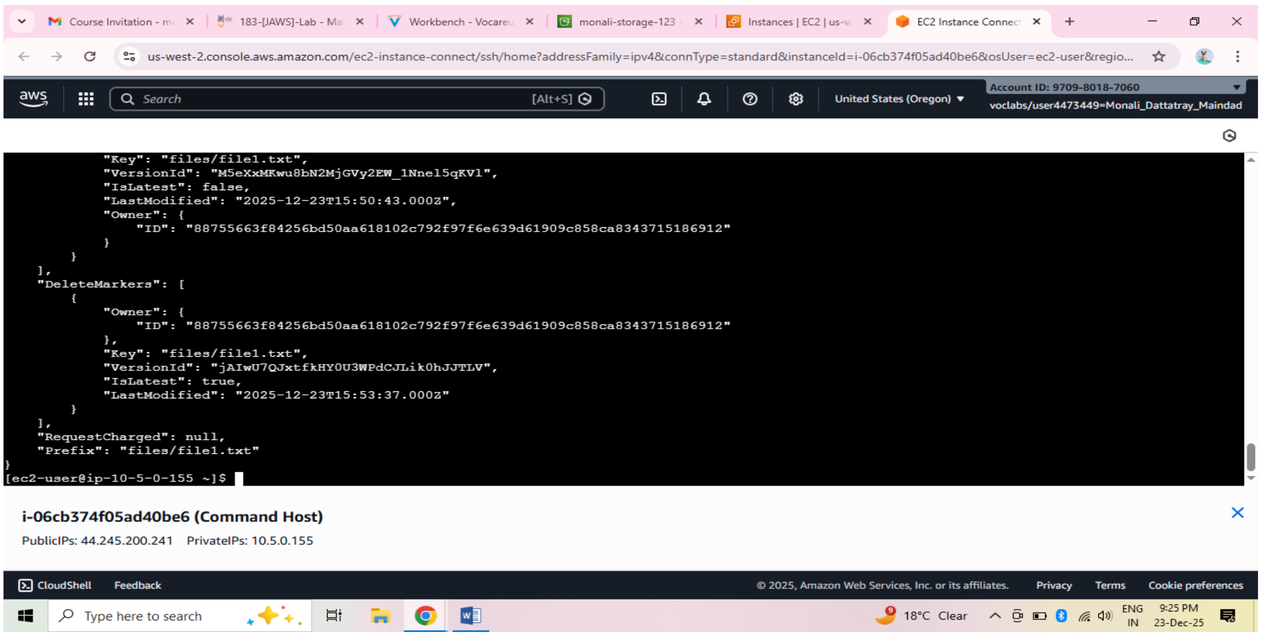
- To download the previous version of file1.txt, run the following command and replace "S3-BUCKET-NAME" with your bucket name:
  - aws s3api get-object --bucket S3-BUCKET-NAME --key files/file1.txt --version-id VERSION-ID files/file1.txt


Course Invitation - m
183-[JAWS]-Lab - Ma
Workbench - Vocareu
monali-storage-123 -
Instances | EC2 | us-w
EC2 Instance Connec
us-west-2.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-06cb374f05ad40be6&osUser=ec2-user&regio...
Search [Alt+S]
United States (Oregon)
Account ID: 9709-8018-7060
voclabs/user4473449=Monali_Dattatray_Maindad
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
18°C Clear
ENG IN
9:25 PM 23-Dec-25

```
            "ChecksumAlgorithm": [
                "CRC32"
            ],
            "ChecksumType": "FULL_OBJECT",
            "Size": 30318,
            "StorageClass": "STANDARD",
            "Key": "files/file1.txt",
            "VersionId": "M5eXxMKwu8bN2MjGVy2EW_1Nnel5qKVl",
            "IsLatest": false,
            "LastModified": "2025-12-23T15:50:43.000Z",
            "Owner": {
                "ID": "88755663f84256bd50aa618102c792f97f6e639d61909c858ca8343715186912"
            }
        }
    ],
    "DeleteMarkers": [
        {
            "Owner": {
                "ID": "88755663f84256bd50aa618102c792f97f6e639d61909c858ca8343715186912"
            },
            "Key": "files/file1.txt",
            "VersionId": "jAIwU7QJxtfkHY0U3WPdCJLik0hJJTLV",
            "IsLatest": true,
            "LastModified": "2025-12-23T15:53:37.000Z"
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155



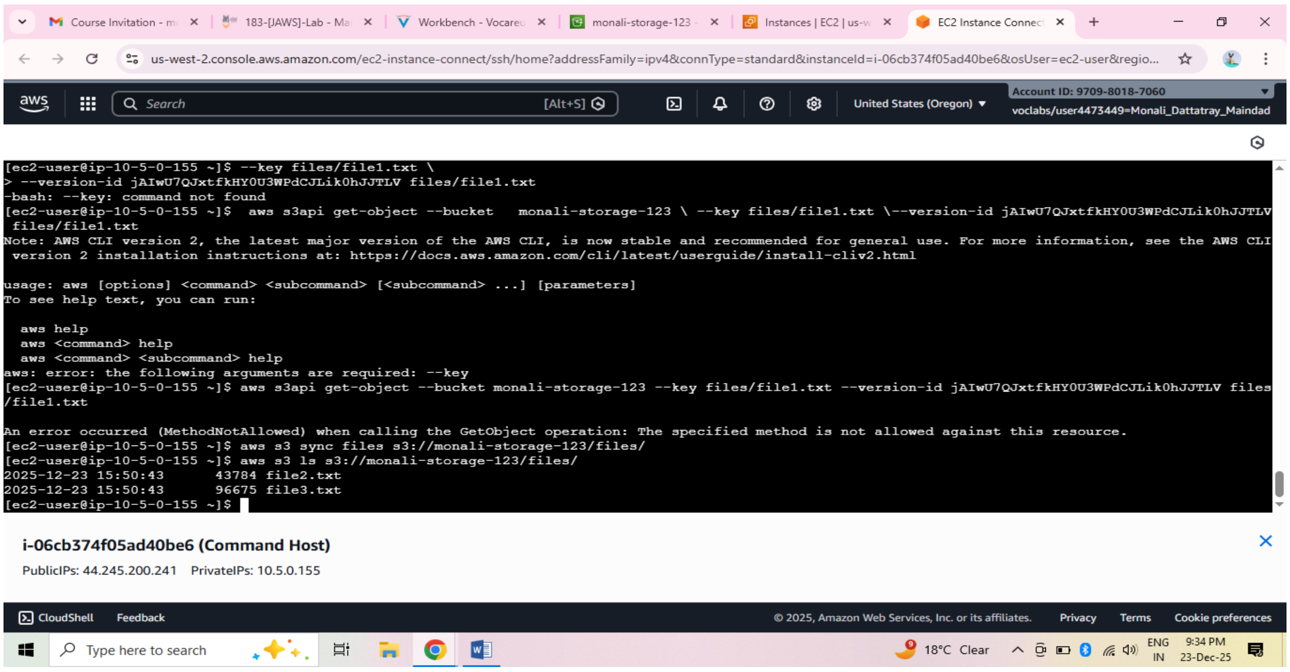
```
            "Key": "files/file1.txt",
            "VersionId": "M5eXxMKwu8bN2MjGVy2EW_1Nnel5qKVl",
            "IsLatest": false,
            "LastModified": "2025-12-23T15:50:43.000Z",
            "Owner": {
                "ID": "88755663f84256bd50aa618102c792f97f6e639d61909c858ca8343715186912"
            }
        }
    ],
    "DeleteMarkers": [
        {
            "Owner": {
                "ID": "88755663f84256bd50aa618102c792f97f6e639d61909c858ca8343715186912"
            },
            "Key": "files/file1.txt",
            "VersionId": "jAIwU7QJxtfkHY0U3WPdCJLik0hJJTLV",
            "IsLatest": true,
            "LastModified": "2025-12-23T15:53:37.000Z"
        }
    ],
    "RequestCharged": null,
    "Prefix": "files/file1.txt"
}
[ec2-user@ip-10-5-0-155 ~]$
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155

- To verify that the file was restored locally, run the following command:
  - ls files

  The command shows all three files listed.
- To re-sync the contents of the files/ folder to Amazon S3, run the following command from within your instance and replace "S3-BUCKET-NAME" with your bucket name:
  - aws s3 sync files s3://S3-BUCKET-NAME/files/
- Finally, to verify that a new version of file1.txt was pushed to the S3 bucket, run the following command and replace "S3-BUCKET-NAME" with your bucket name:
  - aws s3 ls s3://S3-BUCKET-NAME/files/



```
[ec2-user@ip-10-5-0-155 ~]$ --key files/file1.txt \
> --version-id jAIwU7QJxtfkHY0U3WPdCJLik0hJJTLV files/file1.txt
-bash: --key: command not found
[ec2-user@ip-10-5-0-155 ~]$  aws s3api get-object --bucket   monali-storage-123 \ --key files/file1.txt \--version-id jAIwU7QJxtfkHY0U3WPdCJLik0hJJTLV
 files/file1.txt
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for general use. For more information, see the AWS CLI
 version 2 installation instructions at: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

  aws help
  aws <command> help
  aws <command> <subcommand> help
aws: error: the following arguments are required: --key
[ec2-user@ip-10-5-0-155 ~]$ aws s3api get-object --bucket monali-storage-123 --key files/file1.txt --version-id jAIwU7QJxtfkHY0U3WPdCJLik0hJJTLV files
/file1.txt

An error occurred (MethodNotAllowed) when calling the GetObject operation: The specified method is not allowed against this resource.
[ec2-user@ip-10-5-0-155 ~]$ aws s3 sync files s3://monali-storage-123/files/
[ec2-user@ip-10-5-0-155 ~]$ aws s3 ls s3://monali-storage-123/files/
2025-12-23 15:50:43      43784 file2.txt
2025-12-23 15:50:43      96675 file3.txt
[ec2-user@ip-10-5-0-155 ~]$
```

i-06cb374f05ad40be6 (Command Host)

PublicIPs: 44.245.200.241 PrivateIPs: 10.5.0.155

Chapter 6
SECURITY ANALYSIS

Security is a critical aspect of cloud storage and backup management. This project implements multiple AWS security best practices to ensure **data protection, controlled access, and secure automation**.

6.1 IAM ROLE-BASED ACCESS CONTROL

- IAM roles are assigned to EC2 instances instead of using static AWS access keys.
- These roles provide **temporary security credentials** that are automatically rotated by AWS.
- Ensures that EC2 instances can securely access:
  - Amazon EBS for snapshot operations
  - Amazon S3 for file synchronization and recovery
- Prevents credential leakage and unauthorized access.

6.2 PRINCIPLE OF LEAST PRIVILEGE

- IAM policies are configured to allow only the **minimum required permissions**.
- EC2 instances are restricted to:
  - Creating and deleting snapshots
  - Accessing specific S3 buckets only
- Reduces the risk of misuse or accidental modification of other AWS resources.

6.3 SECURE SNAPSHOT MANAGEMENT

- EBS snapshots are created using AWS-managed APIs through the AWS CLI.
- Snapshots are stored securely in Amazon EBS infrastructure.
- Snapshot retention is controlled using a Python script to remove unnecessary older backups, reducing exposure of unused data.

6.4 S3 BUCKET SECURITY

- S3 bucket access is restricted using IAM roles.
- Versioning is enabled to protect against accidental or malicious deletion of files.
- Deleted files are not permanently lost and can be recovered using previous versions.

6.5 DATA PROTECTION AND RECOVERY

- Automated snapshots ensure consistent backups of EBS volumes.
- S3 versioning ensures that data integrity is maintained even after deletion.
- This approach supports **disaster recovery** and prevents data loss.

6.6 NETWORK-LEVEL SECURITY

- EC2 instances are deployed inside a VPC, providing network isolation.
- Access to instances is controlled via AWS security groups.
- Only authorized users can connect using EC2 Instance Connect.

6.7.AUTOMATION SECURITY

- Automation tasks (cron jobs and Python scripts) run locally on EC2 instances using IAM roles.
- No sensitive credentials are stored in scripts or configuration files.
- Logs are maintained to monitor automated operations.

Chapter 7
ADVANTAGES & LIMITATIONS

7.1 ADVANTAGES

1. **Automated Backup and Recovery**
    - EBS snapshots are created automatically using AWS CLI and cron jobs.
    - Reduces the risk of data loss caused by human error.
2. **Reduced Manual Effort**
    - Automation eliminates the need for manual snapshot creation and file backups.
    - Administrators can focus on other tasks instead of repetitive backup operations.
    - Improves operational efficiency and reliability.
3. **Cost-Efficient Snapshot Management**
    - Snapshot retention is managed using a Python script that deletes older snapshots.
    - Helps control storage costs by retaining only the required number of snapshots.
    - Prevents unnecessary accumulation of outdated backups.
4. **High Data Durability Using Amazon S3**
    - Amazon S3 provides extremely high durability by storing data across multiple facilities.
    - Versioning ensures that previous file versions can be recovered easily.
    - Acts as a reliable long-term backup solution independent of EC2 and EBS.

7.2 LIMITATIONS

1. **Fixed Snapshot Scheduling**
    - Snapshots are scheduled at fixed intervals using cron.
    - Does not dynamically adjust based on data changes or workload activity.
    - May create unnecessary snapshots during low-activity periods.
2. **Single Region Dependency**
    - All resources (EC2, EBS, S3) are located in a single AWS region.
    - In case of a regional outage, recovery options are limited.
    - Cross-region replication is not implemented.
3. **Manual Monitoring Required**
    - Cron job failures or snapshot errors require manual checking of logs.
    - No automated alerting mechanism is configured.

Chapter 8

CONCLUSIONS AND FUTURE WORK

8.1 CONCLUSIONS

This project successfully demonstrates an automated approach to managing cloud storage and backups using AWS services. The objectives of creating automated EBS snapshots, synchronizing data to Amazon S3, and enabling secure data recovery through versioning were fully achieved. By using AWS CLI, cron jobs, and Python scripts, the project minimizes manual intervention and improves operational efficiency.

Automation plays a vital role in modern cloud environments by ensuring data reliability, reducing human errors, and optimizing storage costs. The integration of security best practices, scalable storage, and automation tools makes this solution robust, efficient, and suitable for real-world cloud infrastructure management.

8.2 FUTURE SCOPE

1. **Use AWS Lambda Instead of Cron Jobs**
   - Replace cron-based scheduling with **AWS Lambda functions** for snapshot automation.
   - Enables a **serverless architecture**, eliminating the need to manage EC2-based schedulers.
   - Improves scalability and reliability of snapshot operations.
2. **Enable Amazon CloudWatch Alarms**
   - Integrate **CloudWatch monitoring** to track snapshot creation, failures, and storage usage.
   - Configure alarms to notify administrators in case of snapshot failures.
3. **Cross-Region Snapshot Replication**
   - Enable replication of EBS snapshots to another AWS region.
   - Provides **disaster recovery** in case of regional outages.
   - Improves data availability and business continuity.
4. **S3 Lifecycle Policies**
   - Implement lifecycle rules to move older data to **S3 Glacier or Glacier Deep Archive**.
   - Automatically delete obsolete data after a defined period.
   - Reduces long-term storage costs while maintaining compliance.

Chapter 9

REFERENCES

[1] Amazon Web Services, Amazon EBS Documentation, 2023.

[2] M. Armbrust et al., “A View of Cloud Computing,” Communications of the ACM, 2010.

[3] R. Buyya et al., Cloud Computing: Principles and Paradigms, Wiley, 2013.

[4] E. Nemeth et al., UNIX and Linux System Administration Handbook, Pearson, 2017.

[5] M. Lutz, Learning Python, O’Reilly Media, 2013.

[6] Amazon Web Services, Amazon S3 Best Practices and Whitepapers, 2022.

[7] D. Ferraiolo et al., Role-Based Access Control, Artech House, 2016.