Title:

Understanding the Working of AWS - EC2 Instance Creation and S3 Object Storage

Objective: In this lab, I focused on working with AWS services, specifically EC2 for creating virtual servers and S3 for managing object storage. The goal was to set up an EC2 instance, configure it, and securely store data using S3 buckets. By completing these tasks, I gained hands-on experience in leveraging AWS for cloud computing needs.

Note: Screenshots illustrating various steps and outcomes are attached in the screenshots section.

Prerequisites:

Before starting, I ensured I had the following ready:

- An AWS Account
- A basic understanding of cloud computing concepts
- A stable internet connection and a web browser

Theory:

AWS (Amazon Web Services) is a comprehensive cloud computing platform that offers on-demand computing resources and services. Two key services I worked with in this lab were **EC2** and **S3**.

1. **EC2 (Elastic Compute Cloud):**
   EC2 provides scalable compute capacity in the cloud, allowing users to launch virtual servers, known as instances. These instances can run a variety of workloads, from web applications to large-scale computations. EC2 is highly configurable, offering various instance types, AMIs (Amazon Machine Images), and storage options tailored for different needs. Security, network configuration, and monitoring are all integral parts of setting up and managing an EC2 instance.
2. **S3 (Simple Storage Service):**
   S3 is an object storage service designed to store and retrieve any amount of data from anywhere on the web. It’s known for its high durability, availability, and scalability, making it ideal for a wide range of use cases such as data backups, media storage, and content distribution. S3 uses a flat structure (unlike traditional hierarchical file systems), where objects are stored in buckets. Each object in S3 can have customizable permissions, encryption, and versioning to ensure data is securely managed.

AWS allows users to dynamically scale resources based on demand, offering flexibility, cost-efficiency, and reliability for businesses and developers alike.

Materials and Equipment:

For this lab, the following materials and equipment were required:

- **AWS Account:**
  An AWS account was necessary to access the AWS Management Console and experiment with the various services provided.
- **Stable Internet Connection:**
  A reliable internet connection was essential for accessing the AWS console, launching instances, and uploading files to S3.
- **Web Browser:**
  I used a web browser (Google Chrome) to interact with the AWS Management Console, where I managed the EC2 and S3 services.
- **SSH Terminal/Command Line Tool:**
  I used an SSH terminal on my computer to connect to the EC2 instance.

Procedure:

Step 1: Creating an EC2 Instance

1. **Logging into AWS Account:** I began by logging into my AWS account, where I navigated to the AWS Management Console.
2. **Navigating to EC2 Dashboard:** In the AWS Console, I searched for “EC2” and opened the EC2 dashboard. This allowed me to manage and create instances (virtual servers).
3. **Launching a New Instance:** I clicked on "Launch Instance" and chose an Amazon Machine Image (AMI). I opted for the **Amazon Linux** AMI for this lab. Then, I selected the **t3.micro** instance type, which is suitable for low-cost and general-purpose workloads.
4. **Configuring Instance Settings:** After selecting the instance type, I configured various settings, including:
   - Network settings: I used the default VPC (VirtualPrivate Cloud).

- IAM Roles: I didn’t attach any specific IAM roles for this lab.
- Security Group: I configured the new security group with default settings.

5. **Launching the Instance:** Once everything was configured, I reviewed my settings and launched the instance. I created a new key pair for secure SSH access to the instance. This key pair is crucial for securely accessing the EC2 instance later.
6. **Connecting to the EC2 Instance:** After the instance was in a "running" state, I selected it from the EC2 dashboard and clicked on the "Connect" button. I followed the on-screen instructions, using the key pair to SSH into the instance. I used a terminal to connect, and once connected, I was able to interact with the instance’s Linux operating system.

Step 2: Managing Object Storage with S3

1. **Accessing the S3 Service:** Next, I shifted my focus to AWS S3, the storage service. I searched for "S3" in the AWS Management Console and opened the S3 dashboard.
2. **Creating a New S3 Bucket:** In the S3 dashboard, I clicked on “Create bucket.” I provided a unique name for my bucket.
3. **Uploading Objects to the Bucket:** With my bucket created, I selected it and proceeded to upload a new text file. I used the "Upload" button to choose a file from my local machine.
4. **Managing Files in the Bucket:** After the upload process completed, I was able to view and manage the objects within the bucket. I tested the process of downloading one of the files back to my local machine and ensured the storage and retrieval worked as expected.

Connecting to AWS EC2 Instance Using Git Bash

Instead of converting the `.pem` key to a `.ppk` format (as required by PuTTY), I opted for a simpler method using **Git Bash** on my Windows machine. This allowed me to directly connect to the EC2 instance without additional steps.

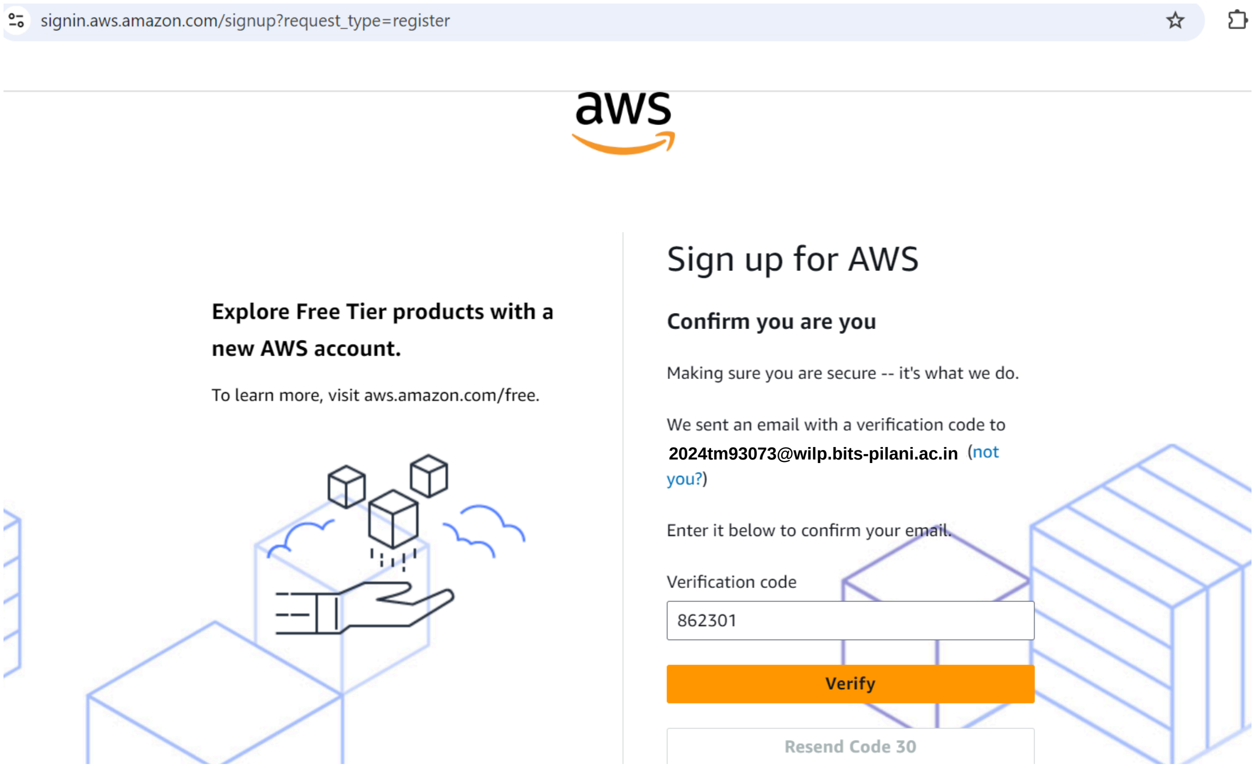
Steps I Followed:

1. **Installing Git Bash**: I downloaded and installed **Git Bash** from the official Git website. This tool allows the use of Linux-style commands directly on a Windows machine.
2. **Opening Git Bash**: After installation, I opened Git Bash from the Start menu.
3. **Navigating to the `.pem` File Location**: In Git Bash, I used the `cd` command to navigate to the directory where my `.pem` key file was stored:

   ```
   a. cd C:\Users\ws_htu1470\Downloads
   ```
4. **Connecting to the EC2 Instance**: Finally, I used the `ssh` command to connect to my EC2 instance, using the `.pem` key file directly. The command I used was:

   ```
   ssh -i bitsfirstkeypair.pem ec2-user@13.60.21.24
   ```
5. For my Amazon Linux instance, I used `ec2-user` as the username.
6. **Successful Connection**: This command successfully connected me to the EC2 instance, allowing me to manage and interact with the server directly from the terminal.

Screenshots:



signin.aws.amazon.com/signup?request_type=register

Explore Free Tier products with a new AWS account.

To learn more, visit aws.amazon.com/free.

Sign up for AWS

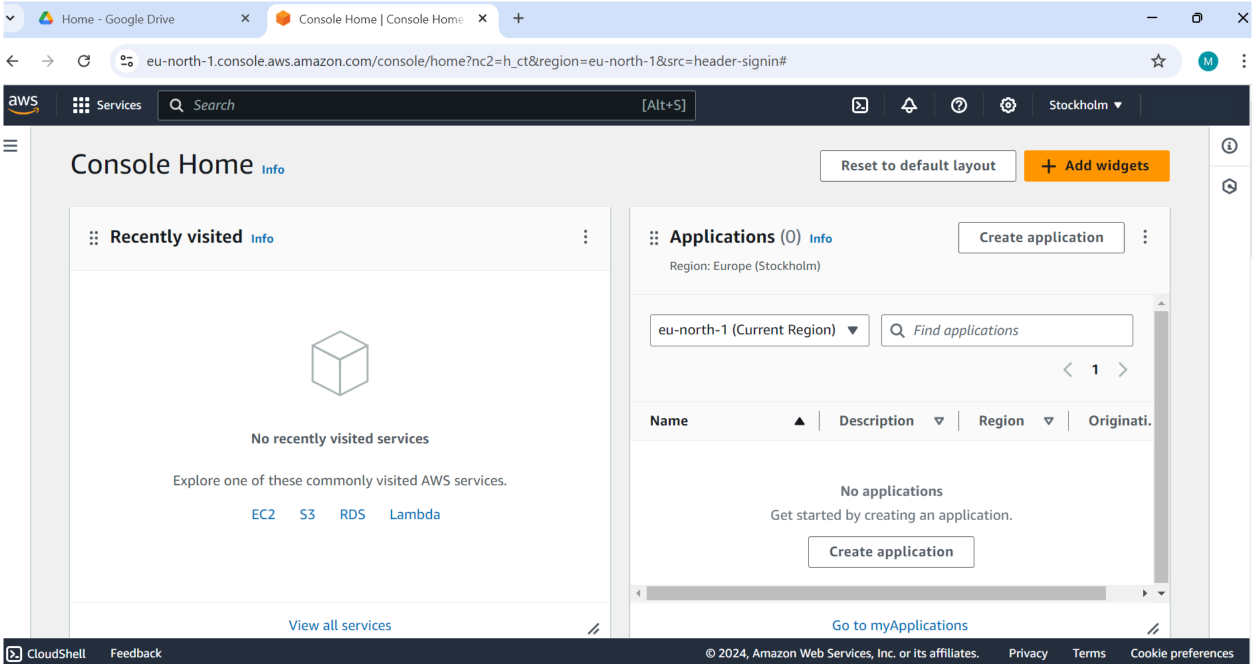
Confirm you are you

Making sure you are secure -- it's what we do.

We sent an email with a verification code to **2024tm93073@wilp.bits-pilani.ac.in** (not you?)

Enter it below to confirm your email.

Verification code: 862301



eu-north-1.console.aws.amazon.com/console/home?nc2=h_ct&region=eu-north-1&src=header-signin#

Console Home Info

Recently visited Info

No recently visited services

Explore one of these commonly visited AWS services.

Applications (0) Info

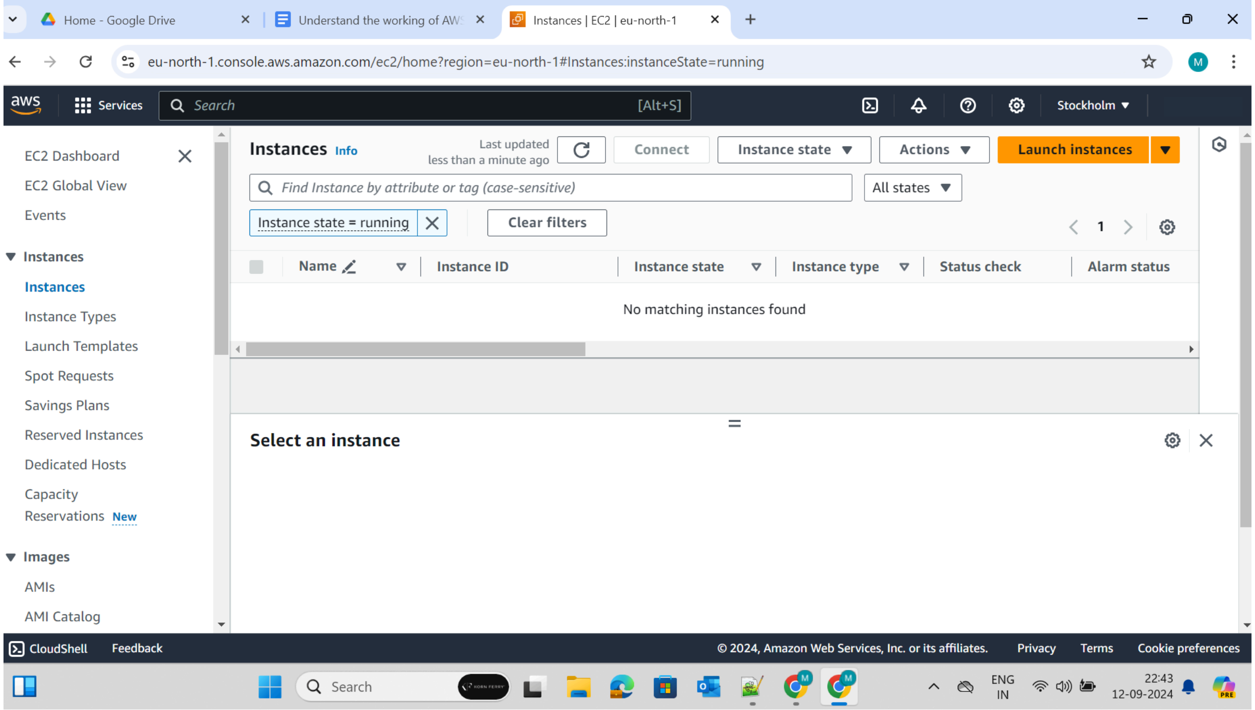
Region: Europe (Stockholm)

Name	Description	Region	Originati.

No applications

Get started by creating an application.

© 2024, Amazon Web Services, Inc. or its affiliates.



eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#Instances:instanceState=running

Instances

Last updated less than a minute ago

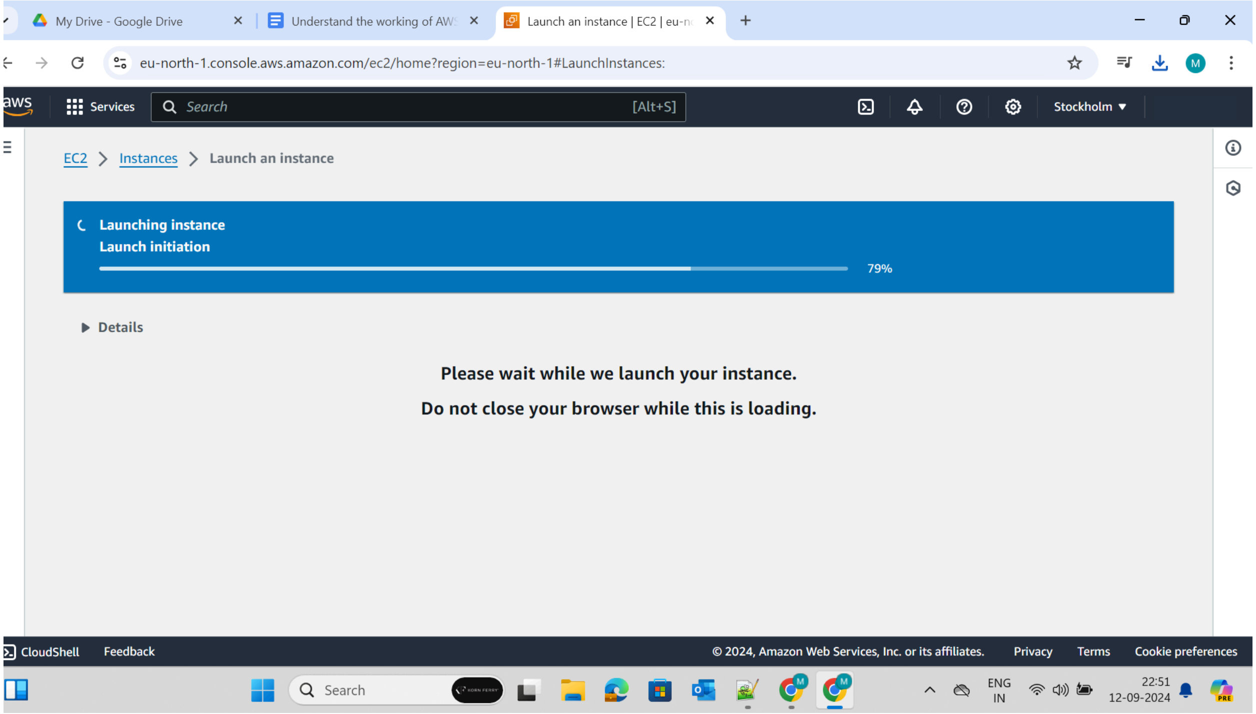
Find Instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status

No matching instances found

Select an instance

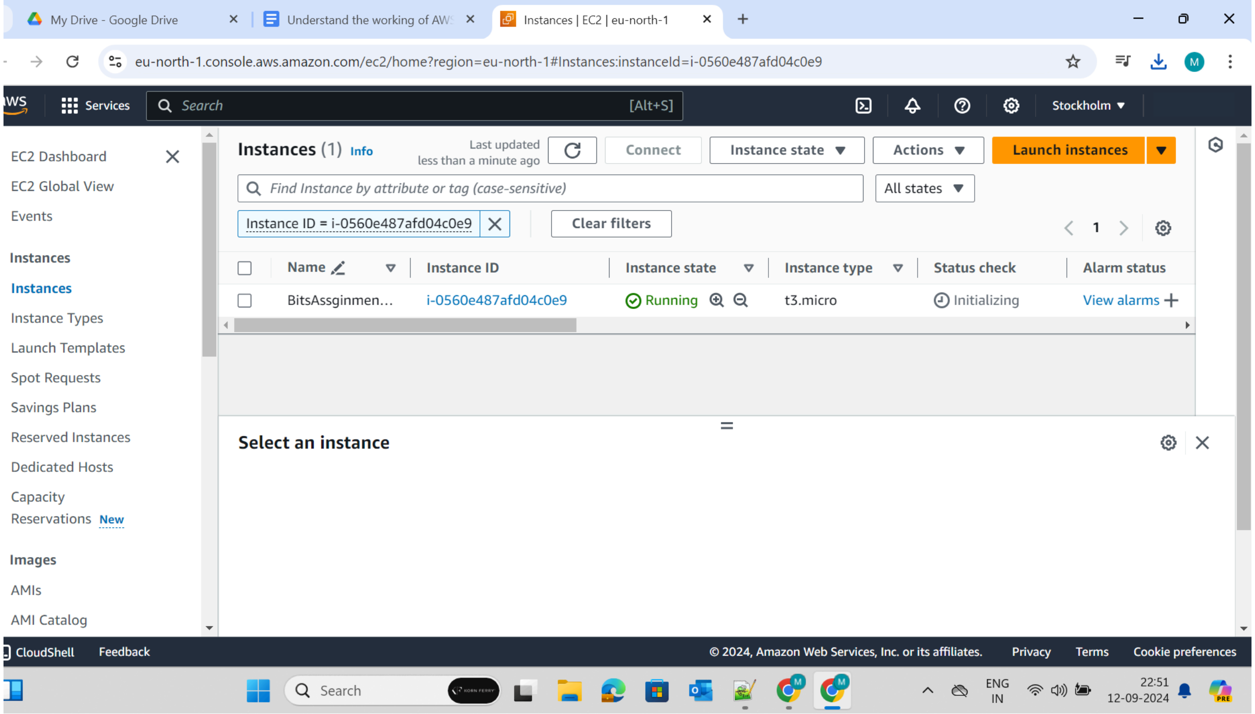


eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#LaunchInstances:

Launching instance
Launch initiation
79%

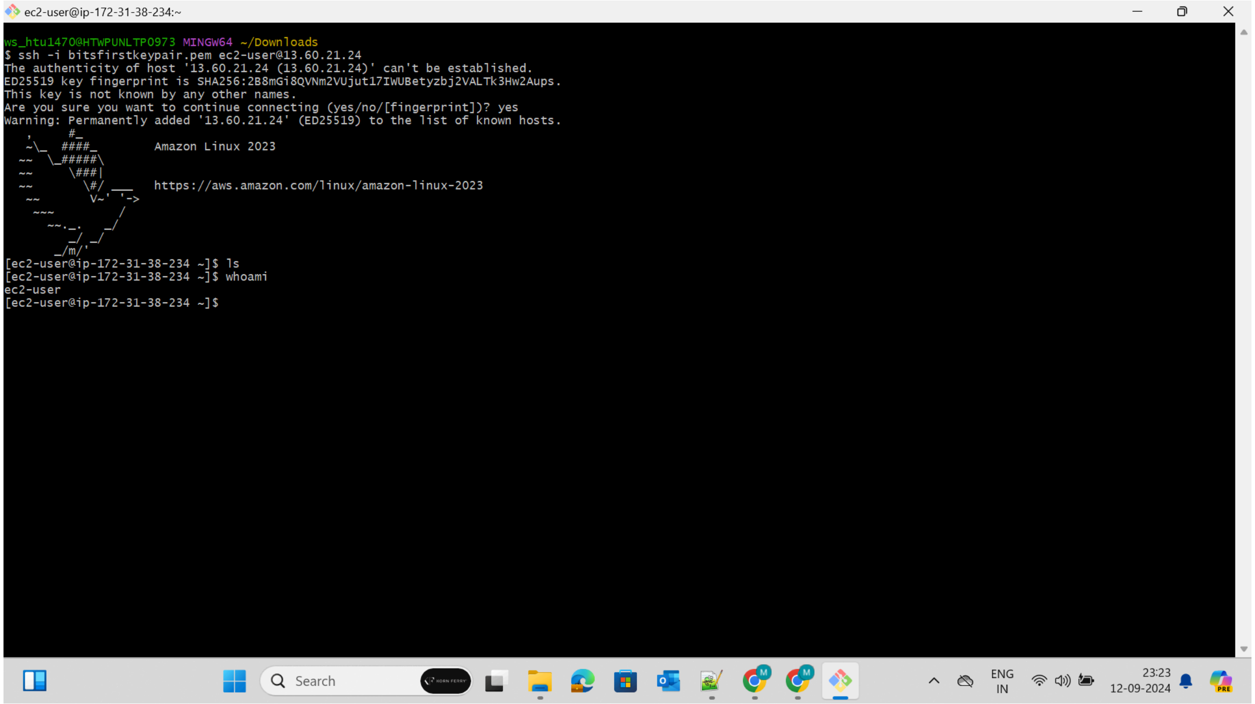
Please wait while we launch your instance.

Do not close your browser while this is loading.

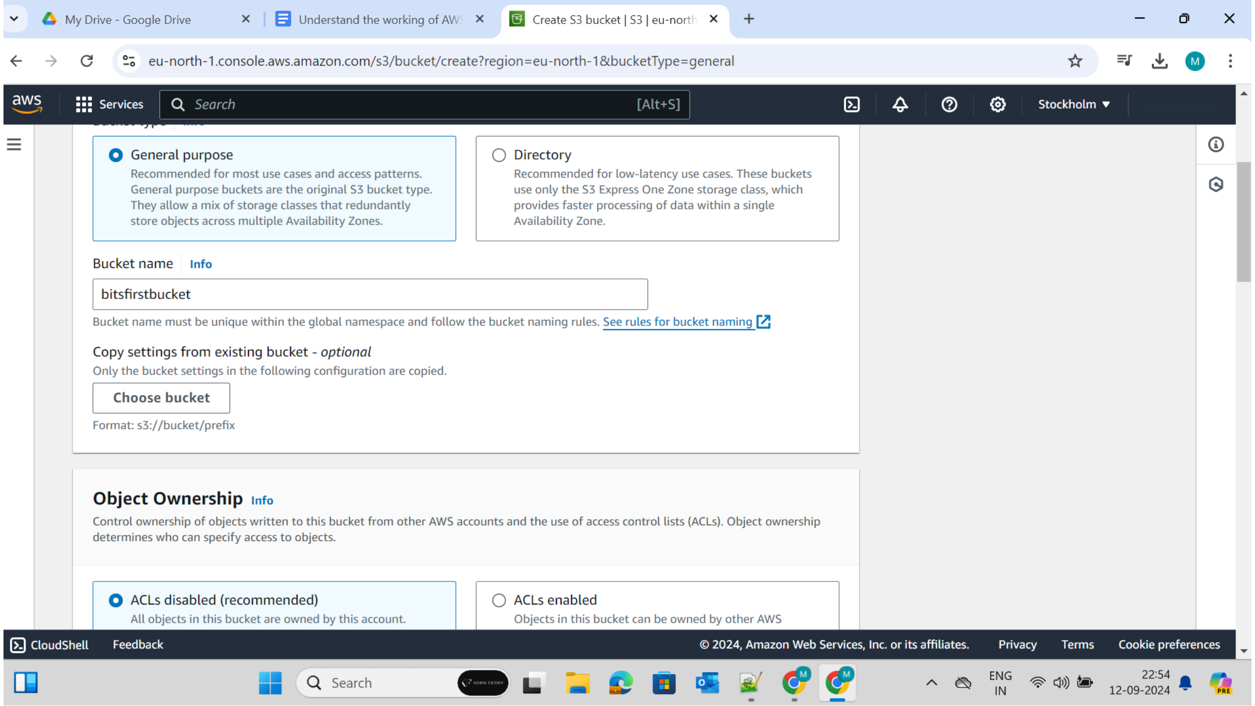


Instance ID = i-0560e487afd04c0e9

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
□	BitsAssginmen…	i-0560e487afd04c0e9	✓ Running	t3.micro	Initializing	View alarms +



```
ws_htu1470@HTWPUNLTP0973 MINGW64 ~/Downloads
$ ssh -i bitsfirstkeypair.pem ec2-user@13.60.21.24
The authenticity of host '13.60.21.24 (13.60.21.24)' can't be established.
ED25519 key fingerprint is SHA256:2B8mGi8QVNm2VUjut17IWUBetyzbj2VALTk3Hw2Aups.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.60.21.24' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-38-234 ~]$ ls
[ec2-user@ip-172-31-38-234 ~]$ whoami
ec2-user
[ec2-user@ip-172-31-38-234 ~]$
```


My Drive - Google Drive
Understand the working of AW
Create S3 bucket | S3 | eu-north
eu-north-1.console.aws.amazon.com/s3/bucket/create?region=eu-north-1&bucketType=general
aws
Services
Search
[Alt+S]
Stockholm ▼
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Search
ENG
IN
22:54
12-09-2024

● General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

○ Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

bitsfirstbucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

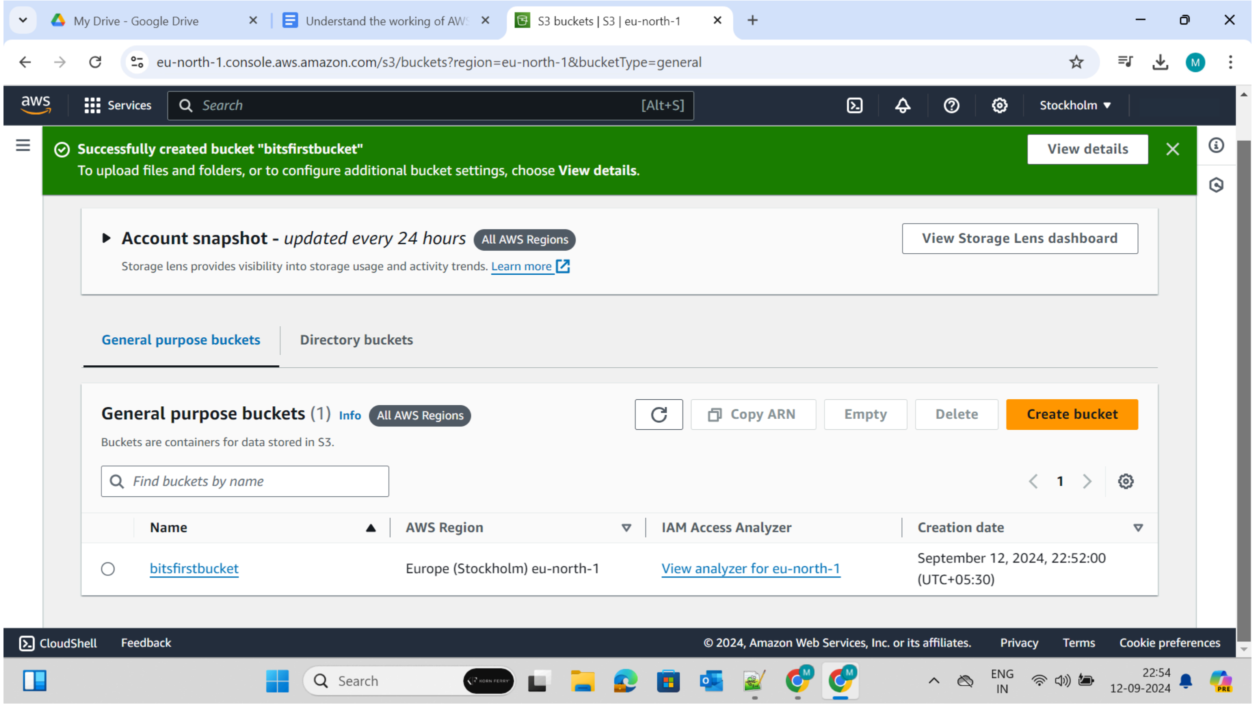
Format: s3://bucket/prefix

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

● ACLs disabled (recommended)
All objects in this bucket are owned by this account.

○ ACLs enabled
Objects in this bucket can be owned by other AWS


My Drive - Google Drive
Understand the working of AW
S3 buckets | S3 | eu-north-1
eu-north-1.console.aws.amazon.com/s3/buckets?region=eu-north-1&bucketType=general
aws
Services
Search
[Alt+S]
Stockholm ▼
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Search
ENG
IN
22:54
12-09-2024

Successfully created bucket "bitsfirstbucket"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

View details

Account snapshot - *updated every 24 hours* All AWS Regions

Storage lens provides visibility into storage usage and activity trends. Learn more

View Storage Lens dashboard

General purpose buckets | Directory buckets

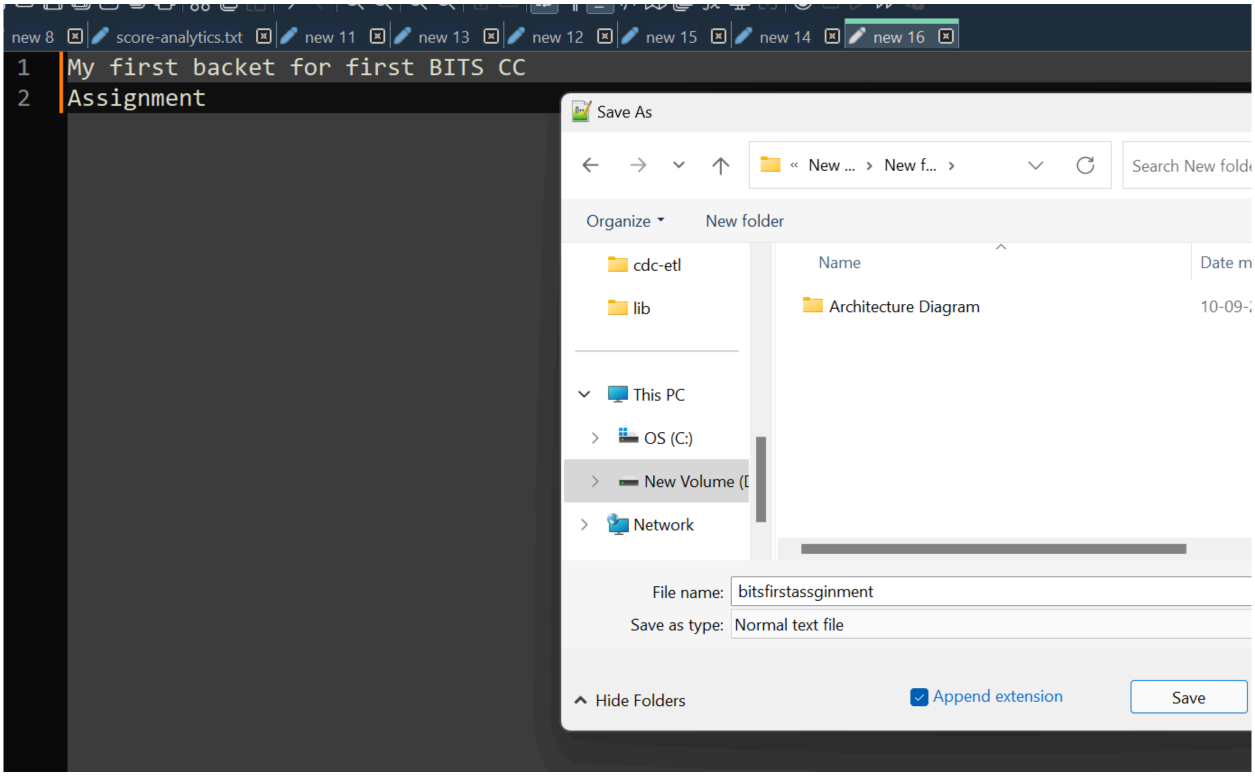
General purpose buckets (1) Info All AWS Regions

Copy ARN | Empty | Delete | Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

	Name ▲	AWS Region	IAM Access Analyzer	Creation date
○	bitsfirstbucket	Europe (Stockholm) eu-north-1	View analyzer for eu-north-1	September 12, 2024, 22:52:00 (UTC+05:30)

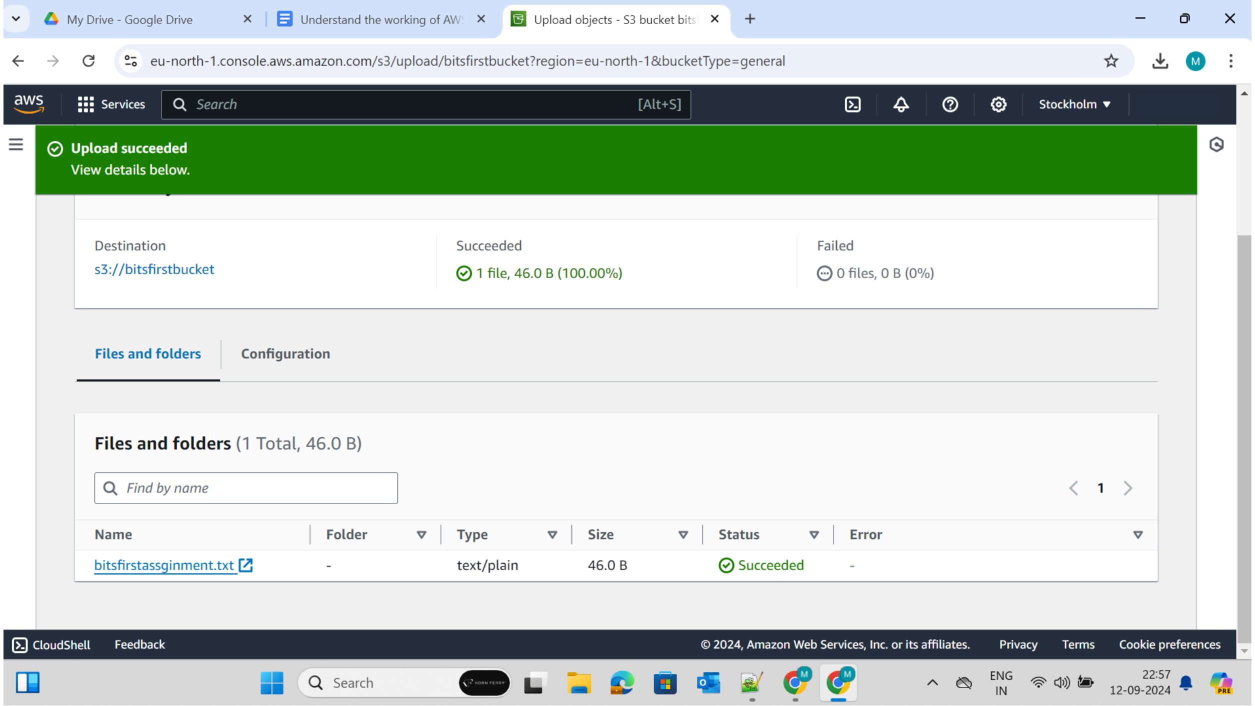


```
My first backet for first BITS CC
Assignment
```

File name: bitsfirstassginment

Save as type: Normal text file

☑ Append extension



Upload succeeded
View details below.

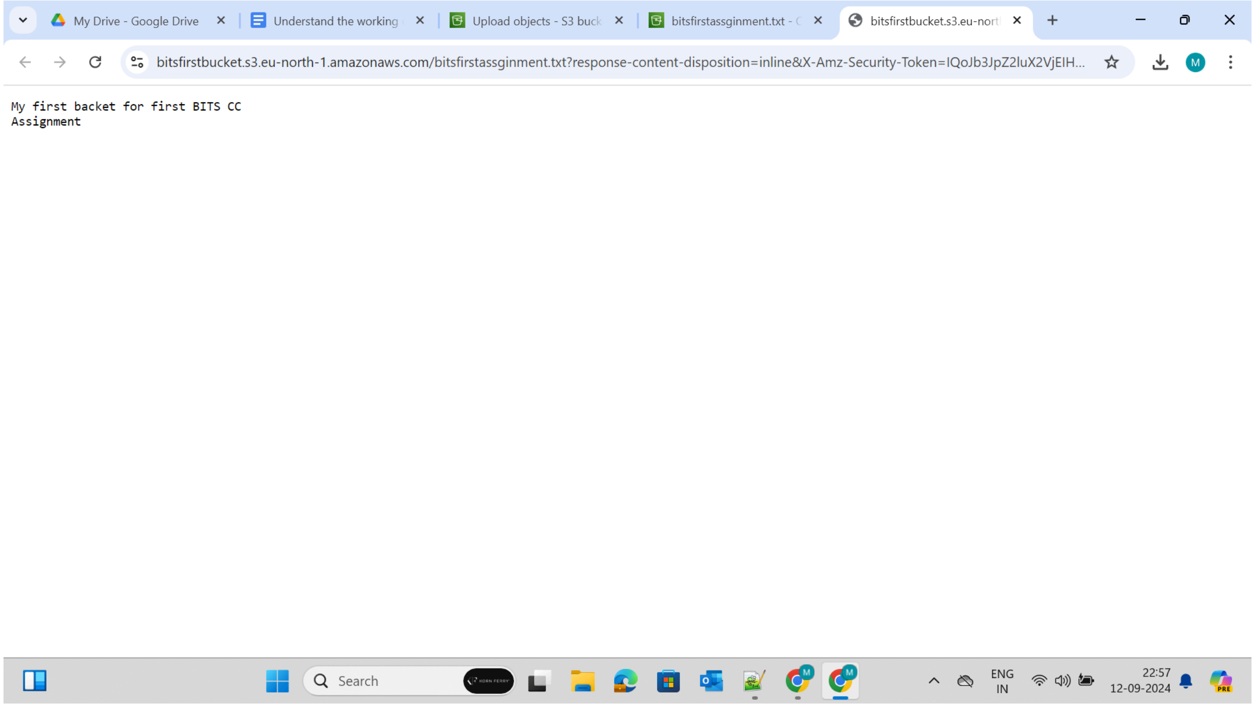
Destination
s3://bitsfirstbucket

Succeeded
1 file, 46.0 B (100.00%)

Failed
0 files, 0 B (0%)

Files and folders (1 Total, 46.0 B)

Name	Folder	Type	Size	Status	Error
bitsfirstassginment.txt	-	text/plain	46.0 B	Succeeded	-



bitsfirstbucket.s3.eu-north-1.amazonaws.com/bitsfirstassginment.txt?response-content-disposition=inline&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEIH...

```
My first backet for first BITS CC
Assignment
```

Results and Observations:

EC2 Instance Creation:

During the instance creation process, I noted how straightforward it was to configure and launch an EC2 instance. The instance was up and running in just a few minutes, and connecting to it via SSH worked seamlessly using the key pair I had set up.

S3 Object Storage:
Creating the S3 bucket was a simple process, and the interface provided all necessary options to manage security and storage settings. Uploading files was quick, and I appreciated how easy it was to manage permissions and access settings for each object..

Innovative Insights:

Cost Efficiency:

AWS's free-tier offerings and the ability to start small with minimal resources (like the t3.micro instance) provided insight into how startups and small projects can leverage the cloud without heavy upfront costs. AWS enables users to only pay for what they use, making it cost-effective and scalable.

Security Configurations:

I learned the importance of configuring security settings accurately. From setting up key pairs for secure SSH access to ensuring S3 objects are protected with the right permissions, AWS emphasizes security at every step, which is crucial for data protection and system integrity.

Challenges Encountered:

During the lab, I faced a few challenges:

1. **SSH Key Pair Issues:** Initially, I encountered difficulty connecting to the EC2 instance via SSH. This was because I had forgotten to select the correct key pair when launching the instance. To resolve this, I had to terminate the instance and launch a new one with the appropriate key pair.
2. **Security Settings Issue:** I blocked my own access to the instance by mistake. I fixed this by changing the security settings to allow access.

Conclusion: By the end of the lab, I had successfully created and launched an EC2 instance and connected to it via SSH. Additionally, I set up a secure S3 bucket and managed file uploads within it. This exercise helped me understand the fundamental processes involved in setting up a cloud server and managing cloud-based object storage using AWS.