REPORT OF CLOUD COMPUTING

BY SACHCHITANAND YADAV

CLOUD COMPUTING MODULE - 19

Learning Objectives -

- Summarize Cloud Computing Concepts
- Explain Cloud Hacking Methodology
- AWS – Practical Hands-On
- Cloud Bucket Enumeration
- Docker Image Vulnerability Assessment
- Cloud Computing Countermeasures

Table of Contents

4.1 Importance of Assessment
4.2 Trivy Overview
4.3 Scanning Process & Steps
4.4 Key Findings & Remediation

5. Cloud Computing Countermeasures

5.1 Identity and Access Management (IAM)
5.2 Data Protection & Encryption
5.3 Secure Configuration Management
5.4 Network Security Controls
5.5 API and Application Security
5.6 Monitoring, Logging & Visibility
5.7 Workload & Container Security
5.8 Patch & Vulnerability Management
5.9 Backup, Recovery & Availability
5.10 Compliance, Governance & Policies

6. Module Summary

6.1 Overview of Cloud Computing Concepts
6.2 Threats, Attacks & Misconfigurations
6.3 Defensive Strategies & Tools
6.4 Foundation for Advanced Cloud Security

Summarize Cloud Computing Concepts: -

Cloud Computing

Cloud computing is the silent engine of today's digital world. It delivers computing power, storage, and software over the internet, removing the need for users to own or maintain physical hardware. What once required bulky servers now lives in vast data centers, humming quietly in the background.

Definition

Cloud computing is the use of internet-based resources to store, manage, and process data instead of relying on a local computer or on-premise servers. The user interacts through a device, while the cloud handles computation and storage remotely.

Common Cloud Service Examples

- Streaming videos on YouTube
- Storing documents on Google Drive
- Hosting applications and websites on Amazon Web Services (AWS)

Core Components of Cloud Computing

Component	Description
Compute	Virtual machines and containers used to execute applications
Storage	Cloud-based storage for files, backups, and databases
Networking	Global connectivity enabling communication between services
Security	Encryption, identity management, firewalls, and threat protection

Advantages of Cloud Computing

Benefit	Description
Cost Efficiency	Pay-as-you-use pricing model
Scalability	Resources can be scaled up or down instantly
Accessibility	Services available anytime, anywhere
Reliability	Built-in redundancy and backups
Managed Infrastructure	No physical hardware maintenance required

Real-World Analogy

Cloud computing is like renting a house instead of building one. You use the space, while the owner manages repairs, upgrades, and utilities. Simple, efficient, no headaches.

Types of Cloud Computing

Cloud computing models are classified into **deployment models** and **service models**.

A. Deployment Models

Model	Description	Example
Public Cloud	Shared cloud services delivered over the internet	Hosting a website on AWS
Private Cloud	Dedicated infrastructure for a single organization	Bank's internal ERP system
Hybrid Cloud	Combination of public and private cloud	Sensitive data private, rest public
Community Cloud	Shared by organizations with similar needs	Healthcare institutions sharing infrastructure

B. Service Models

Model	Full Form	Functionality	Example
IaaS	Infrastructure as a Service	Virtual servers, networking	AWS EC2
PaaS	Platform as a Service	Application development platforms	Google App Engine
SaaS	Software as a Service	Ready-to-use software	Gmail, Zoom
FaaS	Function as a Service	Serverless function execution	AWS Lambda

---

Containers in Cloud Computing

A container is a lightweight, portable unit that packages an application along with all its dependencies. It ensures consistent execution across development, testing, and production environments.

Definition

Containers are self-contained environments that allow applications to run reliably across different computing platforms.

Key Characteristics of Containers

Feature	Description
Lightweight	Minimal resource usage
Portable	Runs across OS and cloud platforms
Fast Startup	Launches within seconds
Isolated	Independent execution
Scalable	Easily replicated under load

Containers vs Virtual Machines

Aspect	Container	Virtual Machine
Startup Time	Seconds	Minutes
Size	Megabytes	Gigabytes
OS	Shares host OS	Full OS per VM
Performance	High	Moderate
Use Case	Microservices, DevOps	Legacy or OS-specific apps

Popular Container Tools

- Docker
- Podman
- containerd
- CRI-O

Containers power modern cloud-native systems such as Kubernetes, Amazon ECS, Azure AKS, and Google GKE.

Docker

Docker is an open-source containerization platform that enables developers to build, ship, and run applications consistently across environments.

Definition

Docker packages applications into standardized containers, ensuring reliability and portability.

Why Docker Is Widely Used

Feature	Description
Portability	Runs on any system or cloud
Consistency	Eliminates environment mismatch
Resource Efficiency	Uses fewer resources than VMs
Rapid Deployment	Containers start instantly
Version Control	Dockerfiles enable repeatable builds

Core Docker Components

Component	Description
Docker Engine	Runtime for building and running containers
Docker Image	Read-only template
Docker Container	Running instance of an image
Dockerfile	Script defining image build steps
Docker Hub	Public image registry

Docker Engine

Docker Engine is the core service responsible for building, running, and managing containers using a client-server model.

Components

Component	Description
Docker Daemon	Manages containers in the background
Docker Client	Command-line interface
REST API	Communication layer

Working Process

1. User executes a command such as `docker run nginx`
2. The client sends the request to the daemon
3. The daemon pulls the image and runs the container

Kubernetes (K8s)

Kubernetes is an open-source container orchestration platform developed by Google and maintained by the Cloud Native Computing Foundation (CNCF).

Definition

Kubernetes automates deployment, scaling, and management of containerized applications across clusters.

Why Kubernetes Matters

Challenge	Kubernetes Solution
Manual deployments	Automated rollouts
Scaling complexity	Auto-scaling
Failure recovery	Self-healing
Traffic management	Built-in load balancing

Key Kubernetes Components

Component	Description
Pod	Smallest deployment unit
Node	Machine running pods
Cluster	Collection of nodes
Deployment	Blueprint for pod management
Service	Stable network endpoint
Ingress	External traffic control
Namespace	Logical separation
Kubelet	Node-level agent
kubectl	Management CLI

Serverless Computing

Serverless computing allows developers to run code without managing servers. The cloud provider handles infrastructure, scaling, and maintenance.

Definition

In serverless computing, developers focus only on code while the platform manages execution.

Working Model

Functions are uploaded → triggered by events → executed instantly → billed per execution.

Key Characteristics

Feature	Description
No Infrastructure Management	Fully managed backend
Event-Driven	Trigger-based execution
Auto-Scaling	Adjusts to traffic
Cost Efficient	No idle cost

Popular Serverless Platforms

- AWS Lambda
- Azure Functions
- Google Cloud Functions
- IBM Cloud Functions
- Cloudflare Workers

Cloud Security

Cloud security focuses on protecting data, applications, and infrastructure in cloud environments.

OWASP Top 10 Cloud Security Risks (CNAS-2023)

Rank	Code	Risk
1	CNA01	Cloud provider misconfiguration
2	CNA02	Poor change management
3	CNA03	Excessive privileges
4	CNA04	Insecure APIs
5	CNA05	Lack of monitoring
6	CNA06	Weak workload configurations
7	CNA07	Poor segmentation
8	CNA08	Exposed secrets

Rank	Code	Risk
9	CNA09	Supply chain vulnerabilities
10	CNA10	Denial of Service attacks

Major Cloud Threats

- Data breaches
- Account hijacking
- Misconfigured storage
- Insider threats
- Weak IAM policies
- DDoS attacks
- Vendor lock-in
- Supply chain attacks

Threat Mitigation Techniques

Category	Controls
Data Security	Encryption, DLP, classification
Identity Management	Least privilege, MFA
Network Security	VPCs, WAF, security groups
Monitoring	SIEM, logs, alerts
Compliance	ISO 27001, GDPR, HIPAA
API Protection	OAuth2, rate limiting

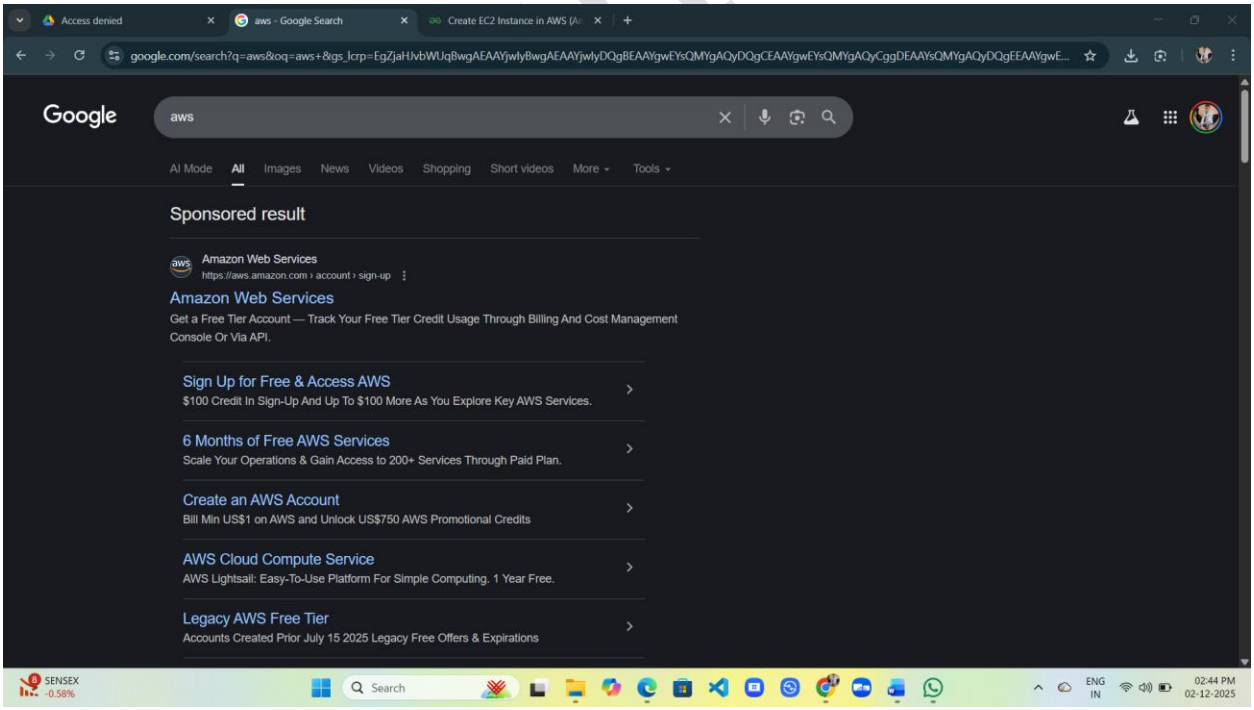
Create EC2 Instance in AWS (Amazon)

Definition: -

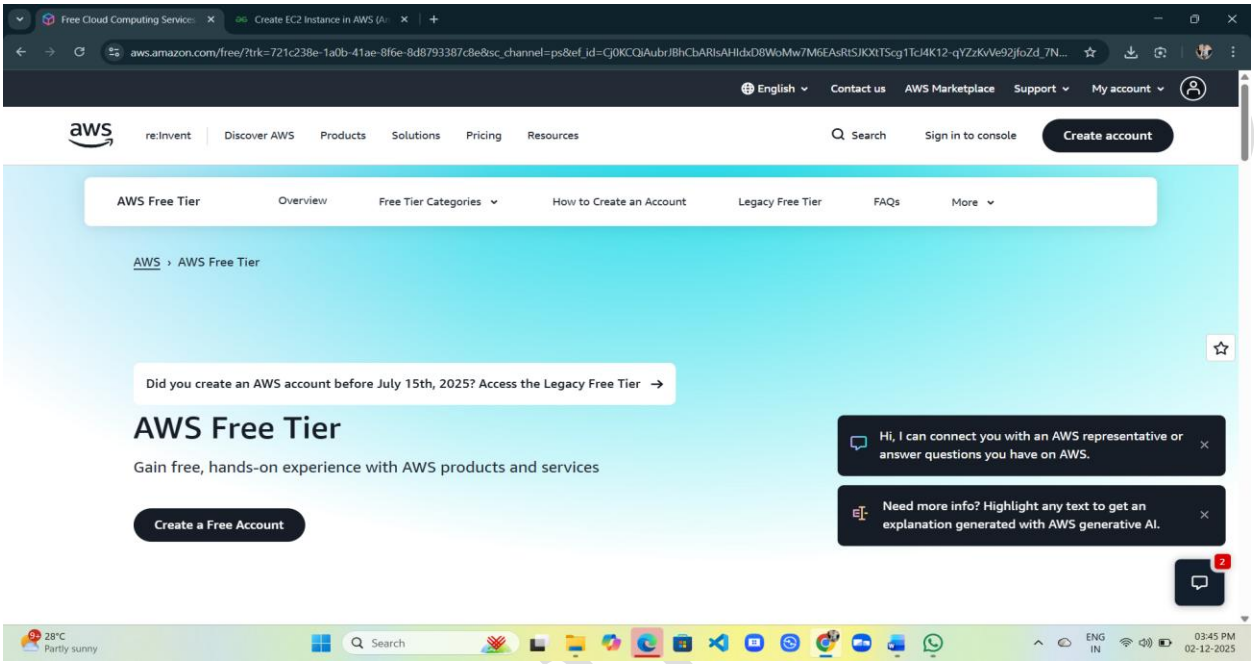
Launching an EC2 instance in AWS allows you to set up a virtual server in the cloud within minutes. This virtual server can host your web apps, run software, or perform automated tasks all without needing a physical machine.

How to use -

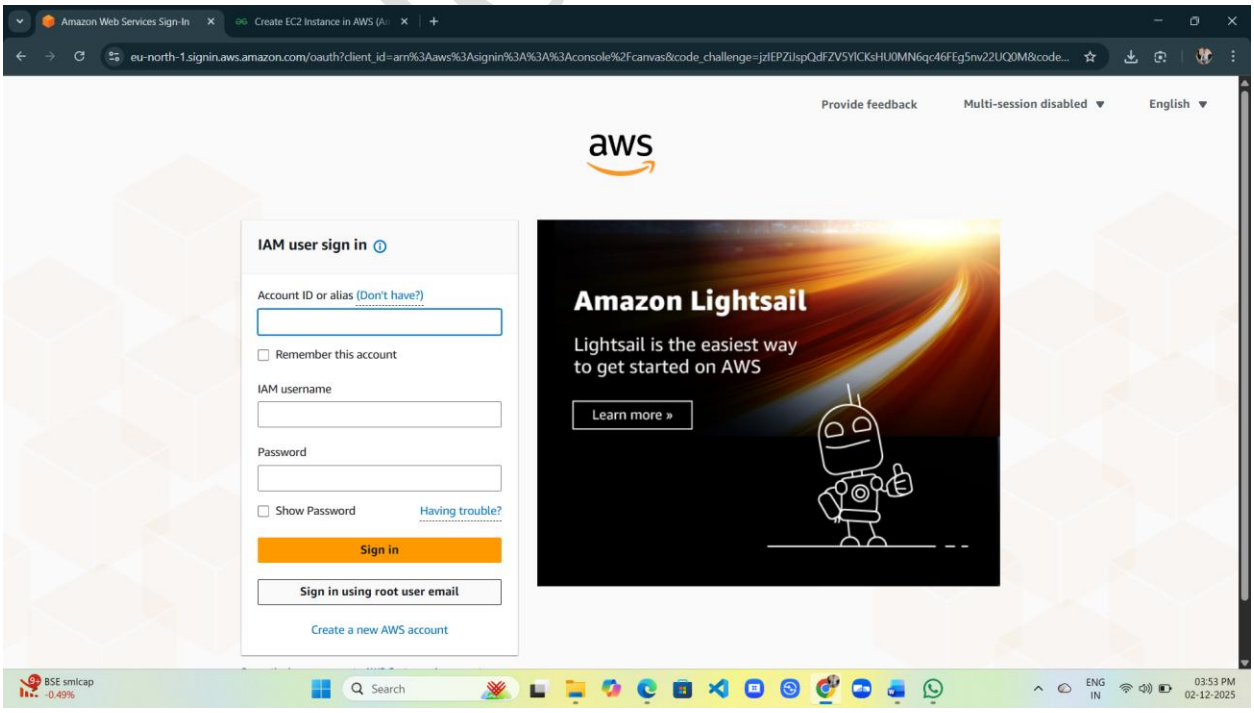
- Open browser and search AWS
- Click on the first link



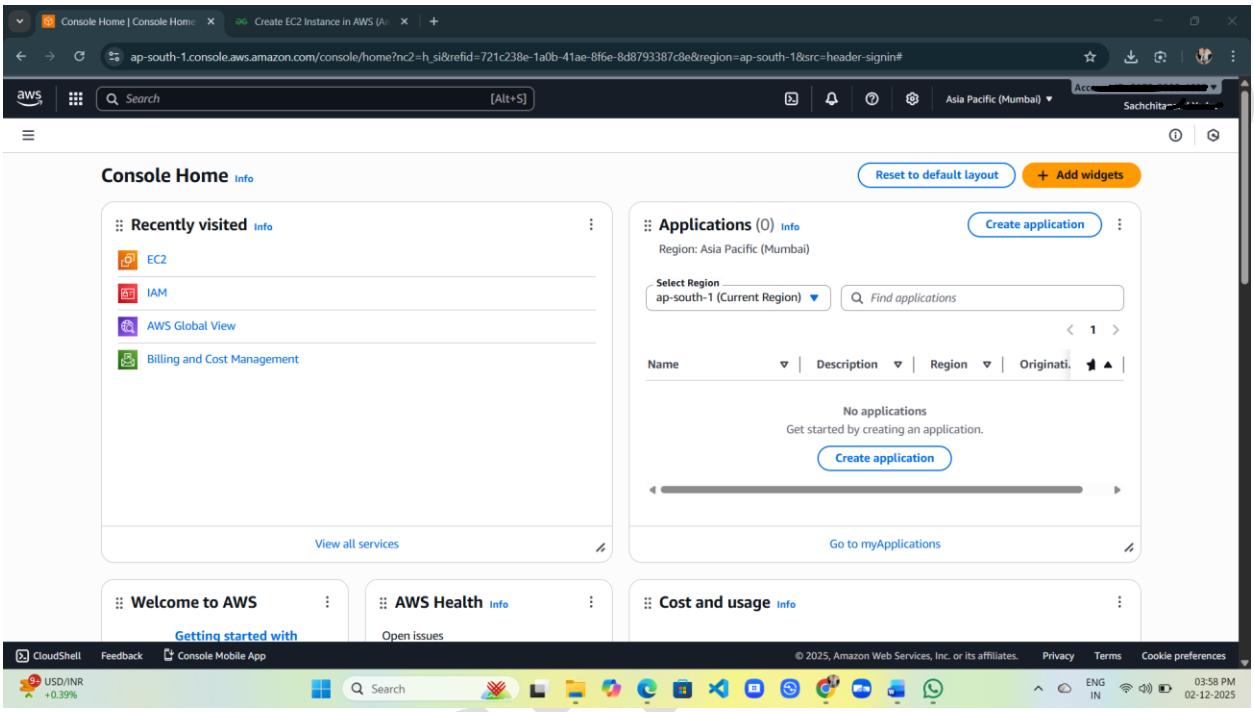
- **Log in to your AWS Management Console.**



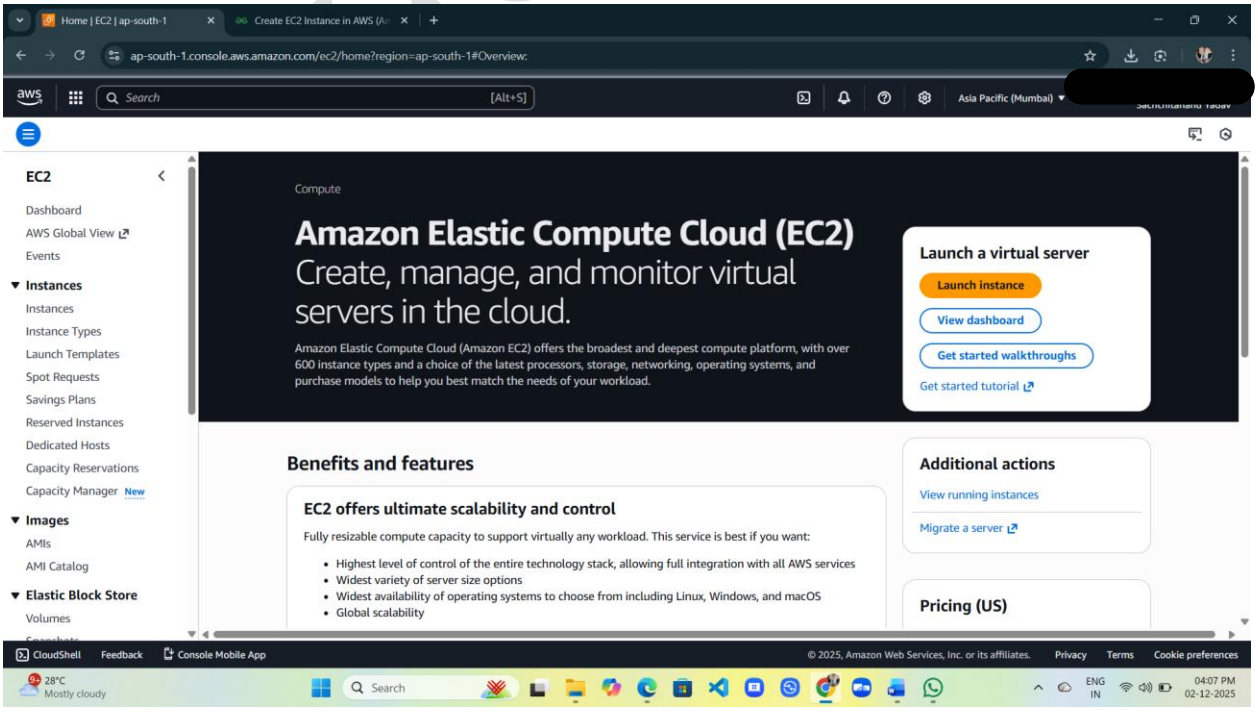
- **Login to AWS Console**:Go to: https://console.aws.amazon.com/ec2/
- **Select Root User**



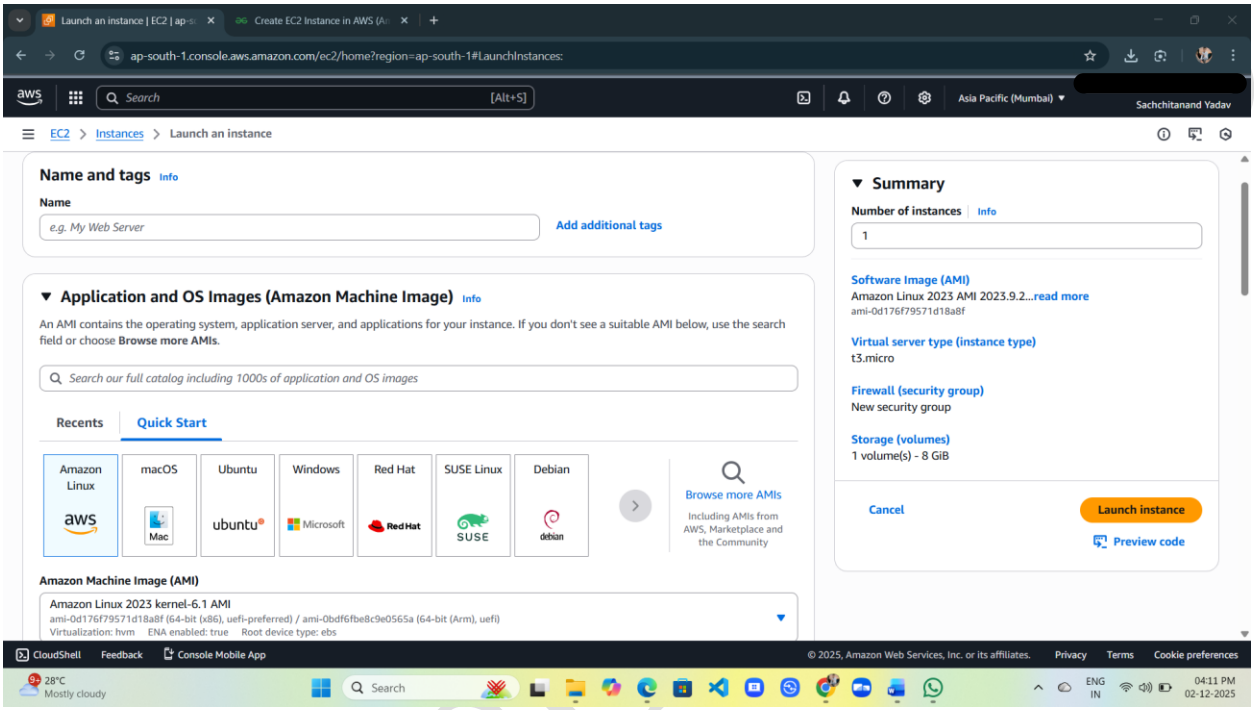
- **From the Services menu, choose EC2 under the Compute section.**



- **Click Launch Instance.**

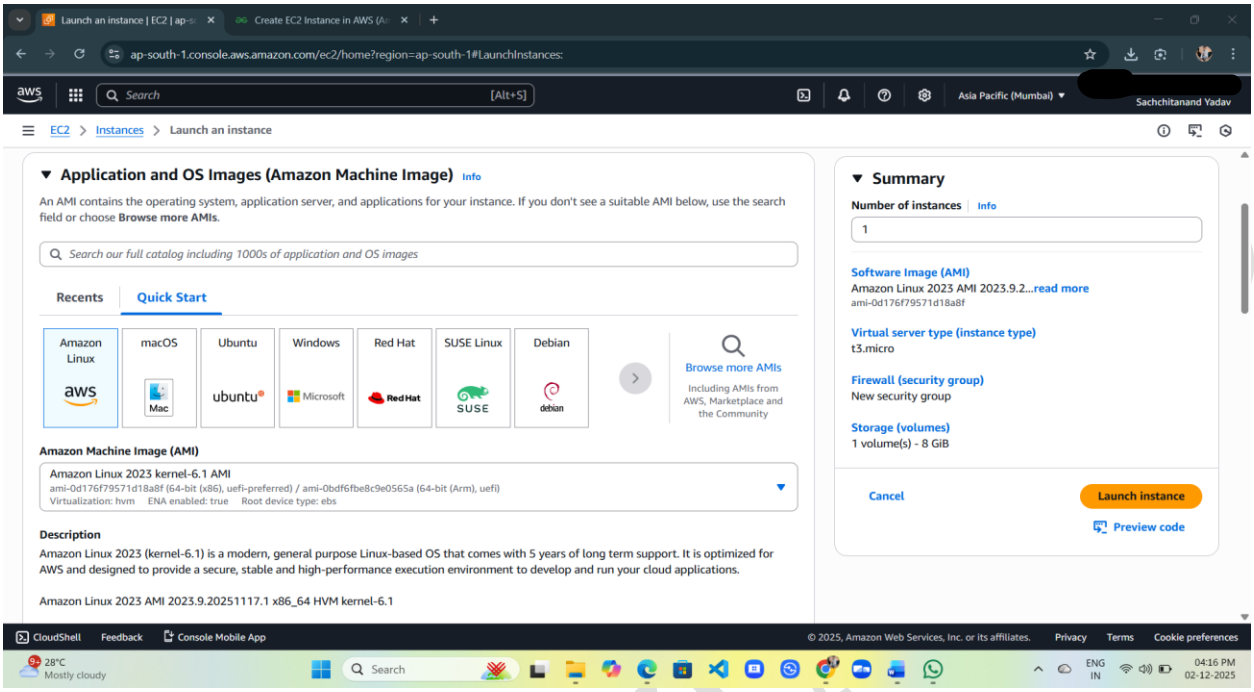


- **On the "Launch an Instance" page, enter a name for your instance (e.g., my-first-ec2-server).**

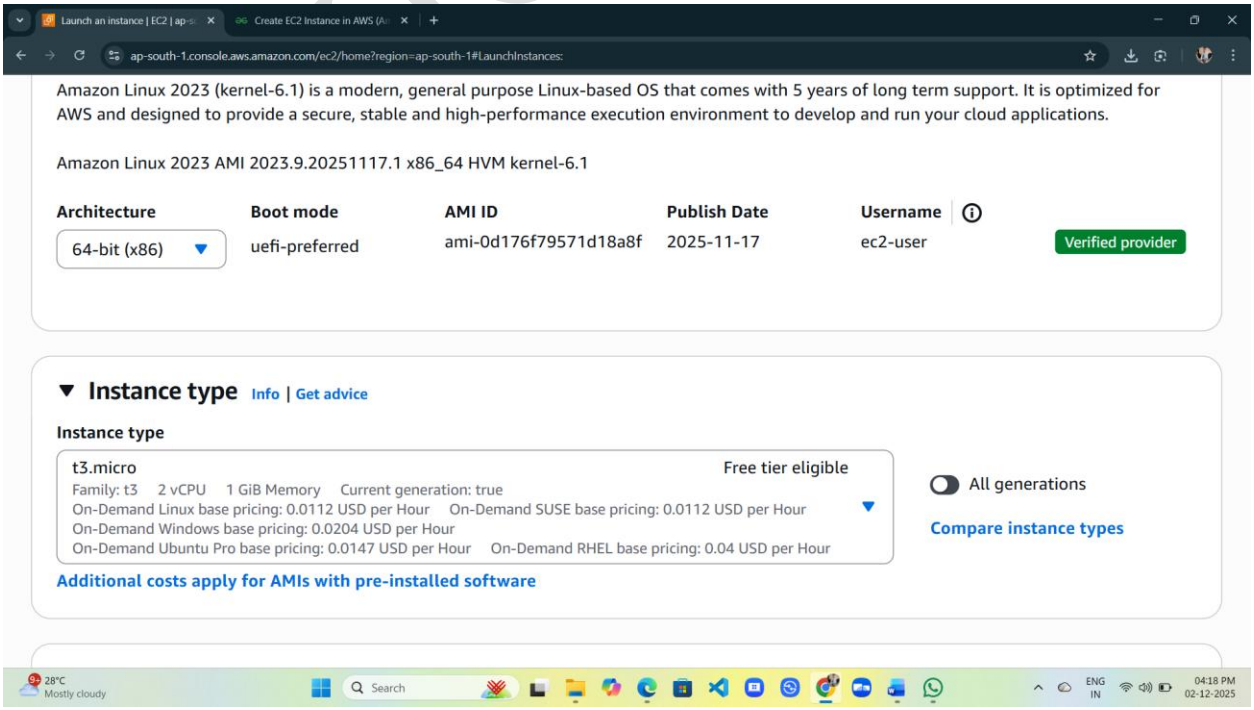


- **Choose Amazon Machine Image**

  - Select an Amazon Machine Image (AMI), which is the OS for your server.
  - For beginners, choose Amazon Linux 2, Ubuntu, or Windows, depending on your needs.
  - AMIs come preconfigured with OS and some software like templates.



- **Select Instance Type**
  - Select the instance type (defines CPU and memory).
  - For Free Tier, choose t2.micro — 1 vCPU and 1 GB RAM.
  - Avoid selecting higher types like t2.small, t3.medium, etc., unless needed, as they may incur charges.



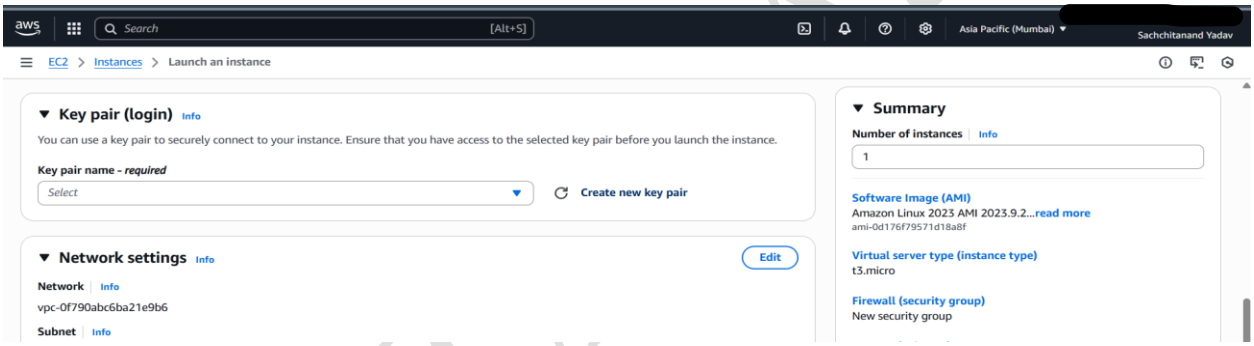
- **Configure Key Pair**

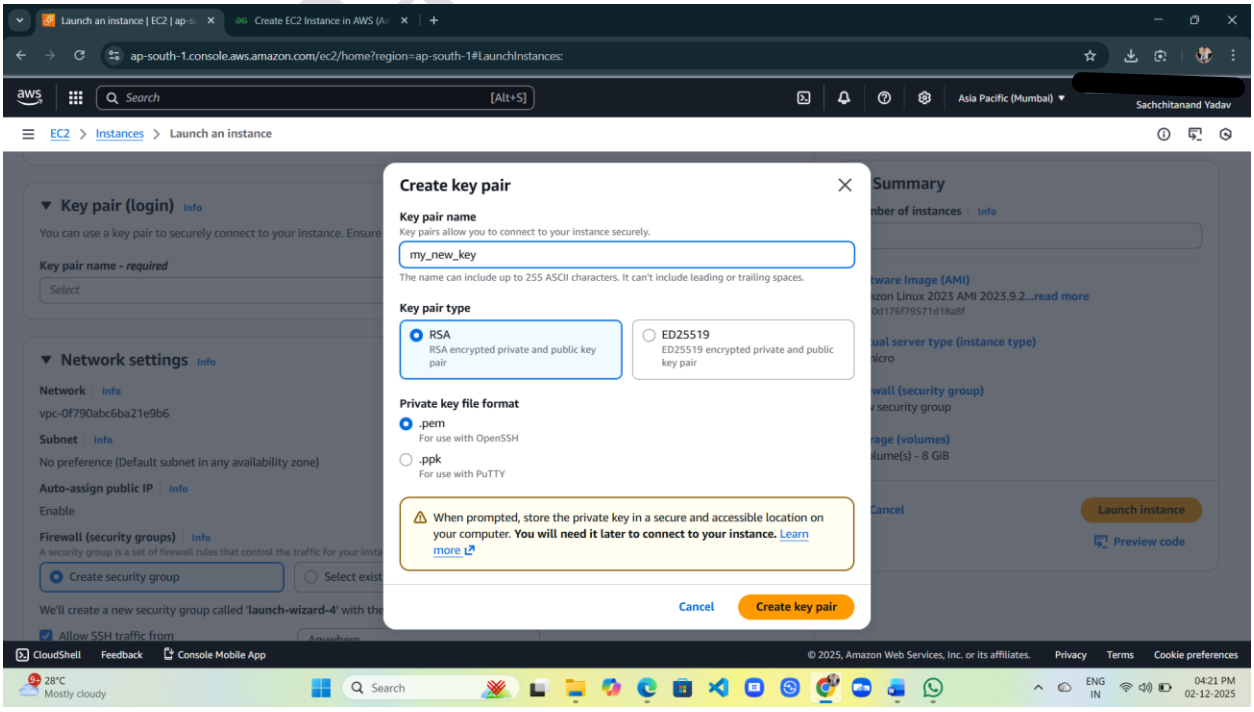
1. EC2 instances use SSH key pairs for secure access.

2. Click Create new key pair:

- Enter a name.
- Choose file format: .pem for Linux/macOS or .ppk for Windows (for PuTTY).
- Download the key file and save it securely (you won't be able to download it again).

3. Select the created key pair from the dropdown.





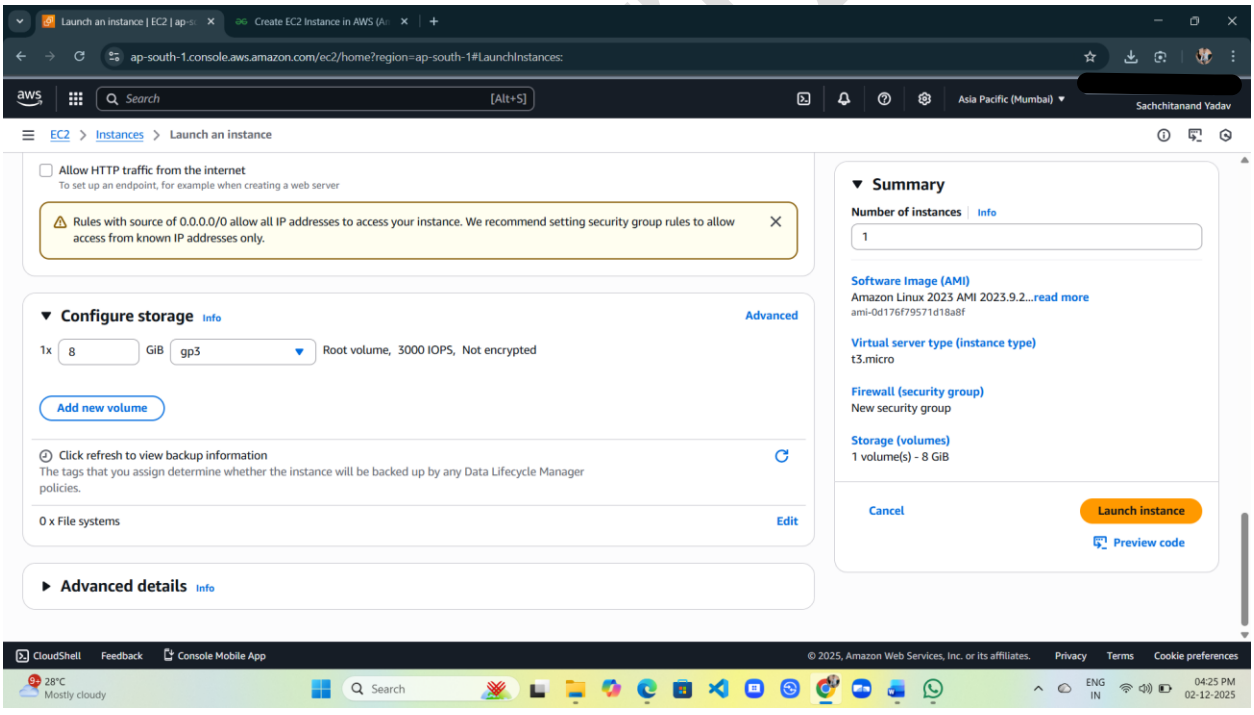
- **Network and Storage Configuration**

1. Network Settings: Use the default VPC and subnet unless you have specific networking needs.

2. Firewall (Security Group): Allow SSH (port 22) for Linux or RDP (port 3389) for Windows.

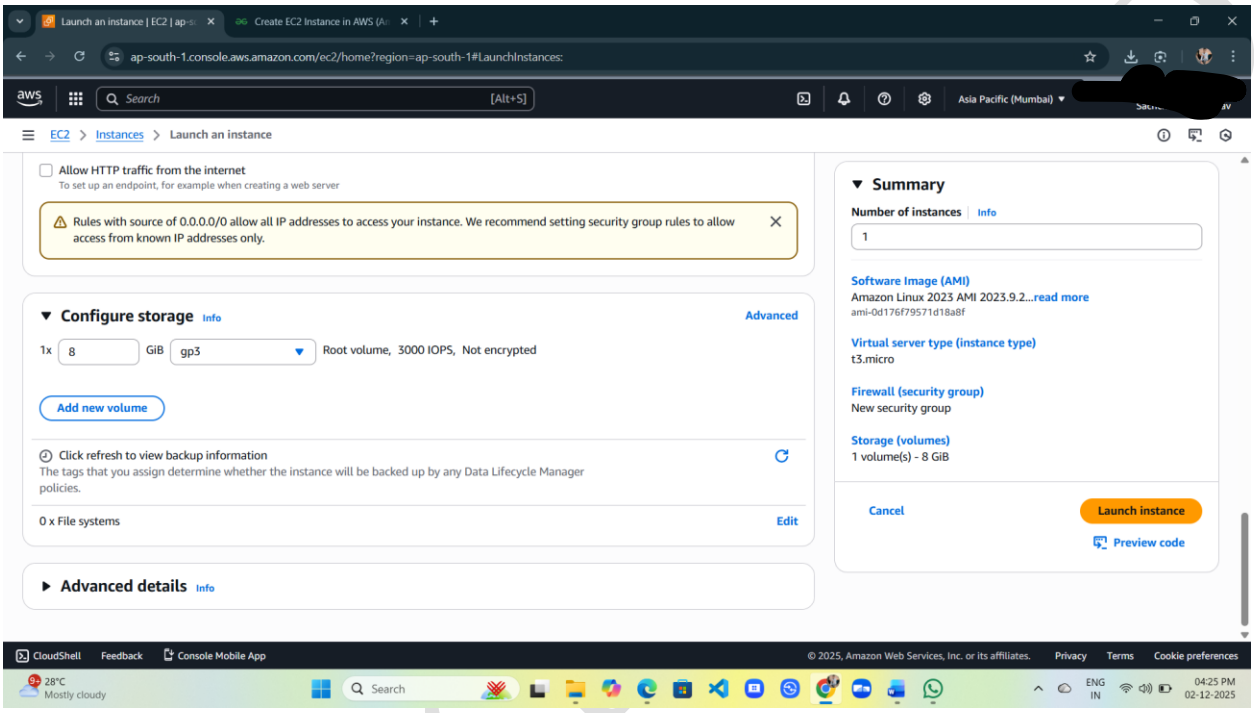
3. Storage Settings:

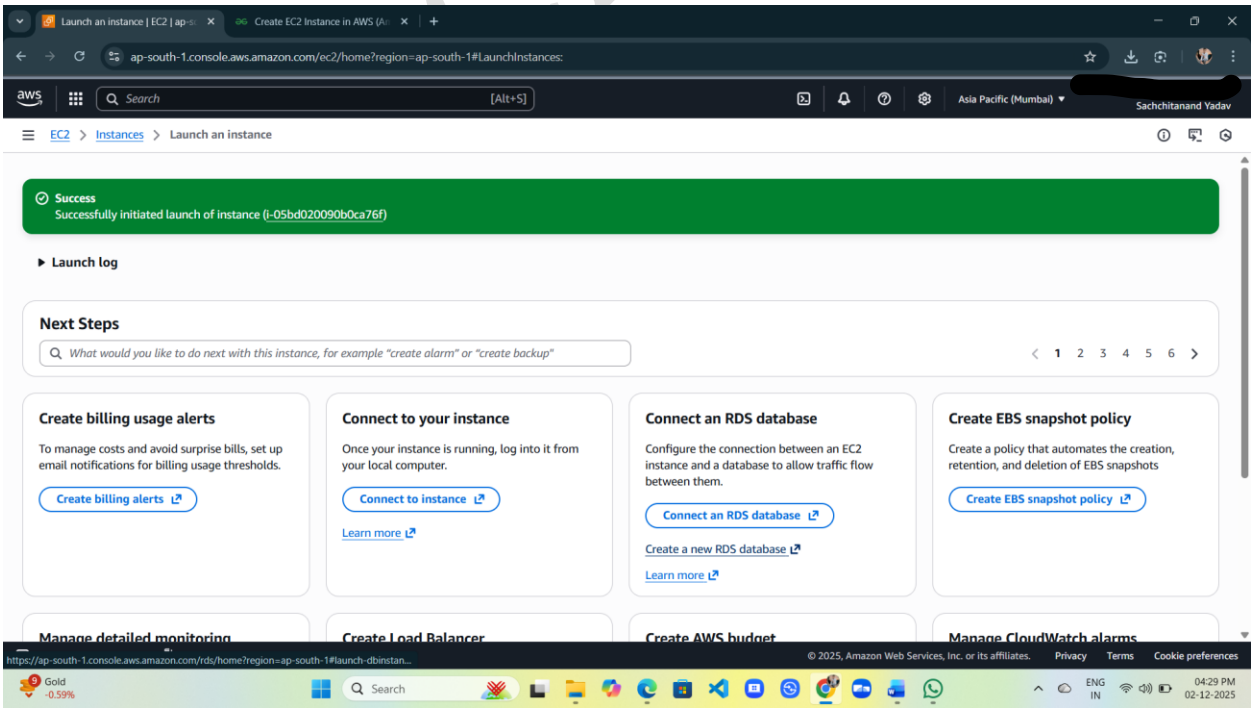
- Free Tier allows up to 30 GB of General Purpose SSD (gp2).
- Keep default (8 GB) or increase as needed.



- **Review and Launch**
  - Review all configurations to ensure they are Free Tier eligible.
  - Click Launch Instance.

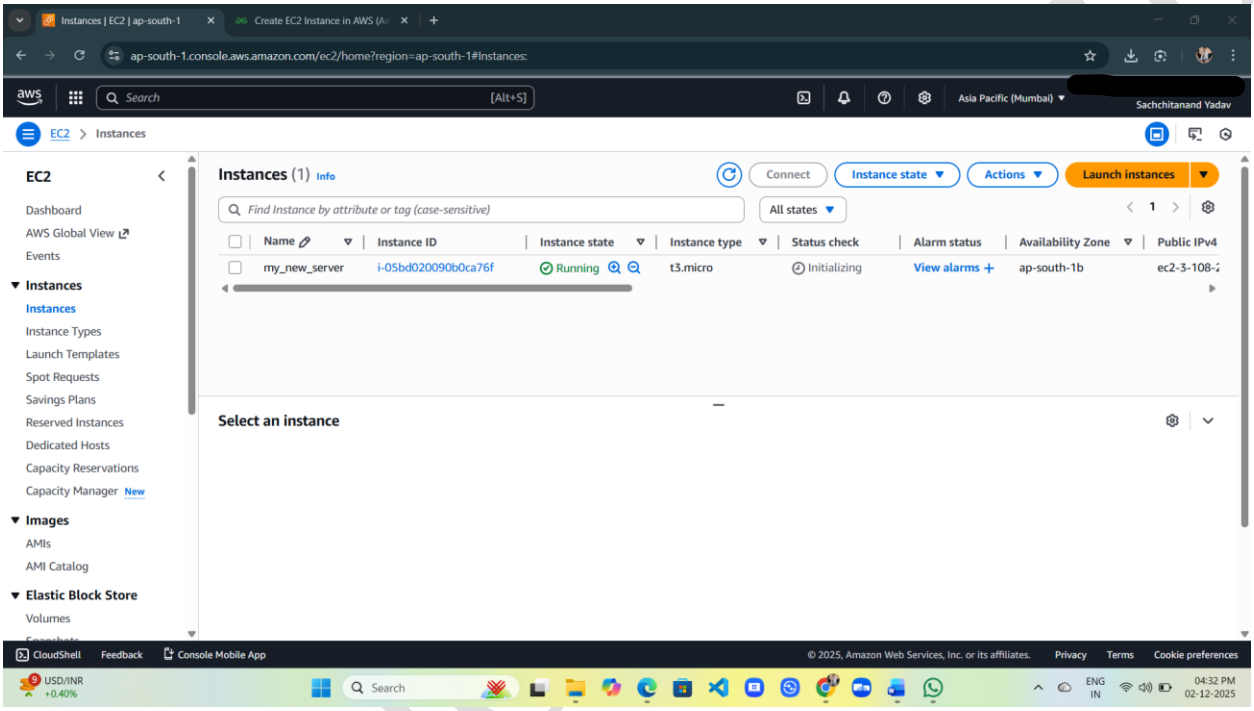
- You will see a confirmation page. Click View Instances to see your new server being initialized.





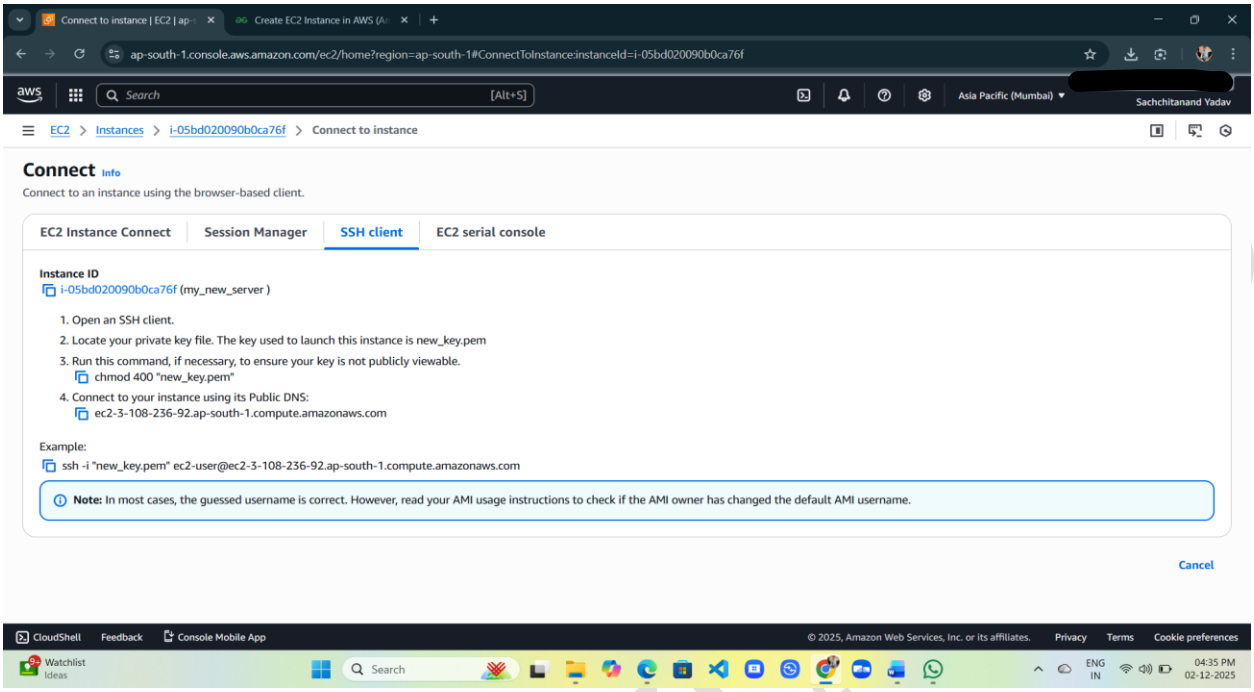
- **Locate Connection Details**

  - Select the server to which you want to connect and click on the connect button at the top of that instance as shown in the image below.



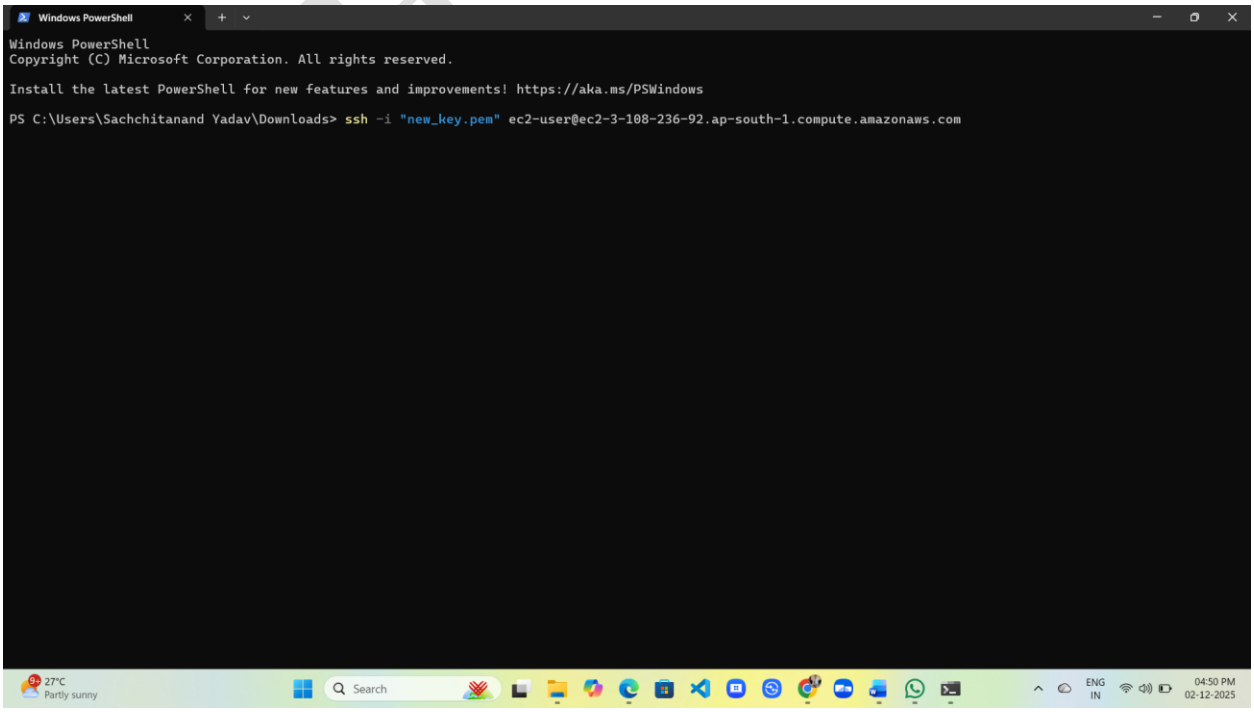
- **Copy the SSH Command**

  - Copy the SSH key which is right following the example it will acct as a key-pair to connect to EC2-Instance.

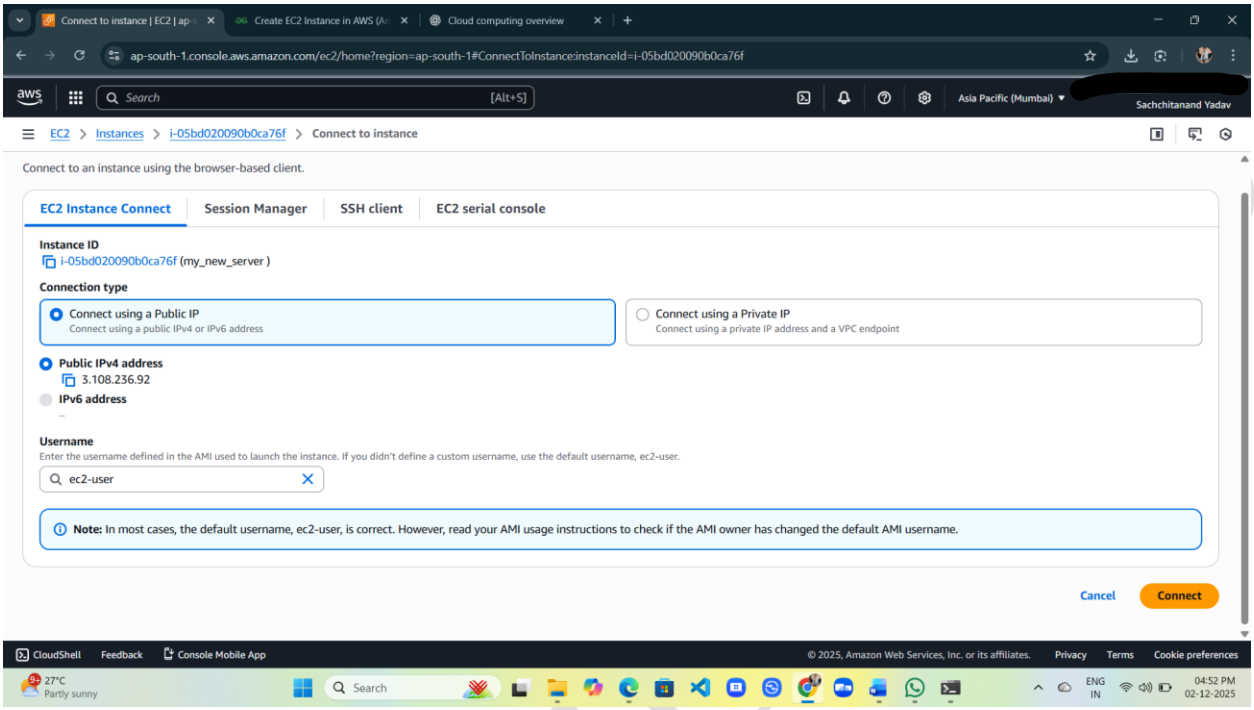


- **Step 3: Use Terminal**

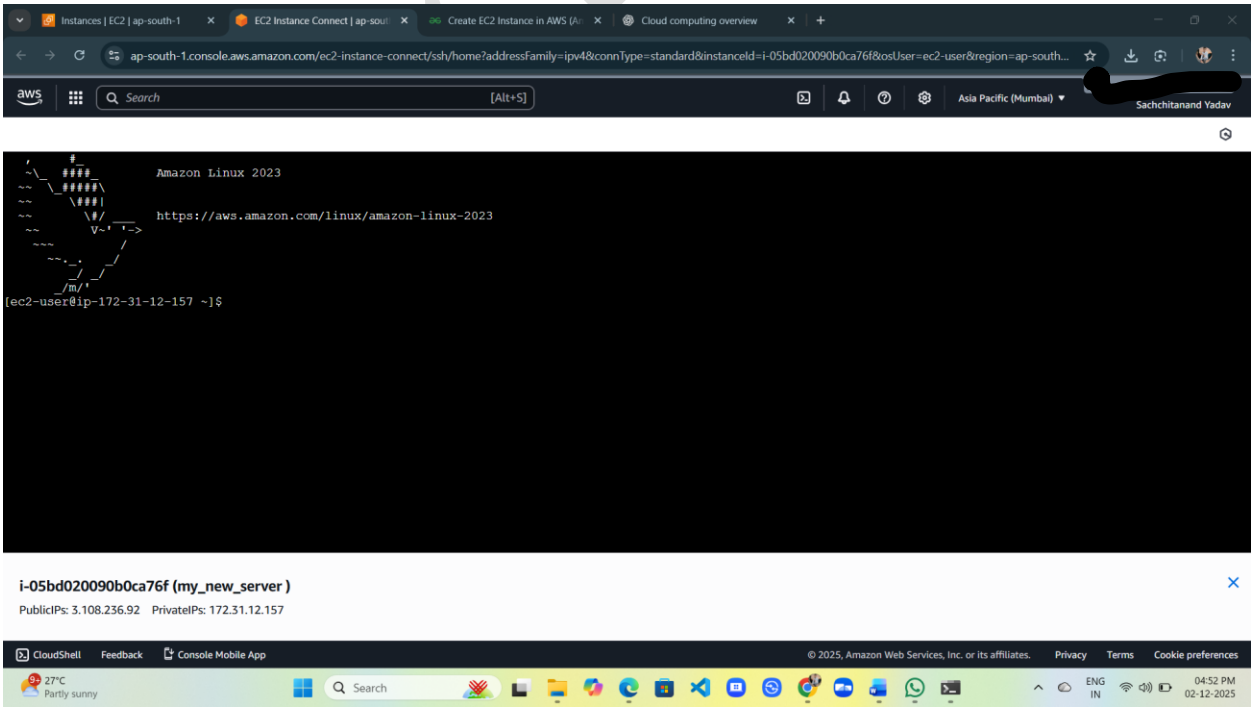
  - Open the terminal and go to the folder where your .pem file is located and paste the key that you have copied in AWS and paste it in the terminal.

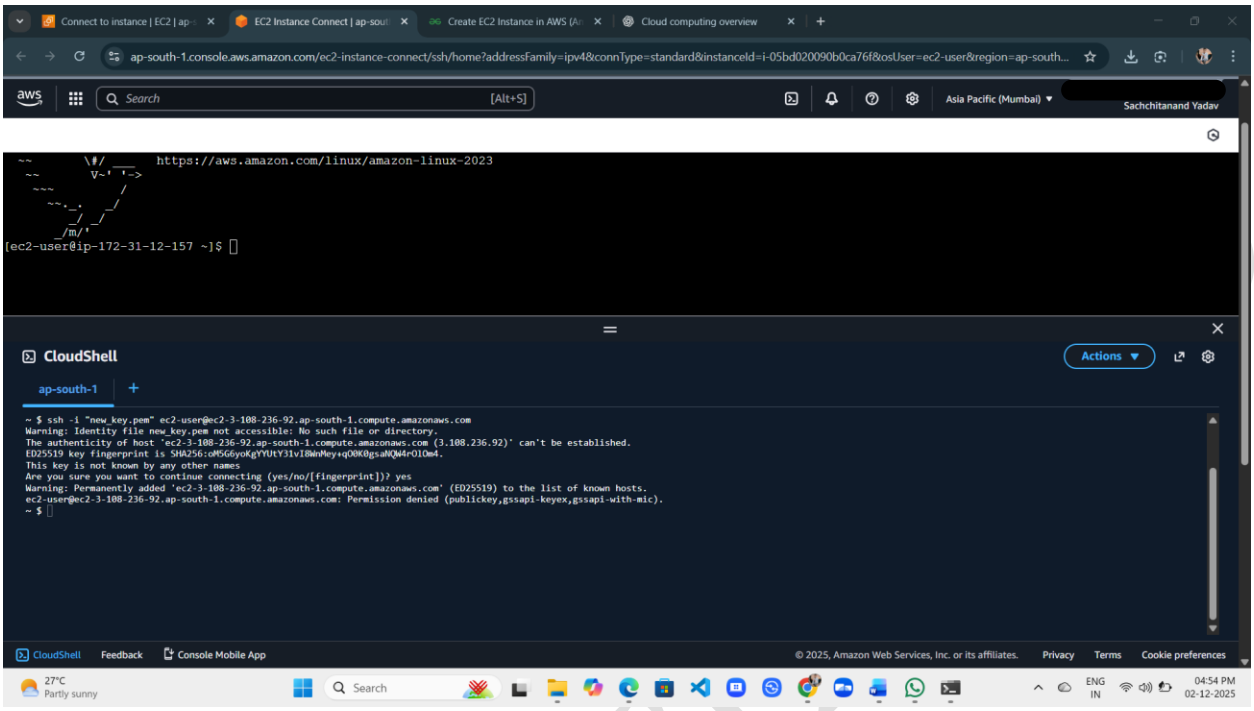


- **Click Connect**



- **Connected Successfully**




Connect to instance | EC2 | ap-
EC2 Instance Connect | ap-sout
Create EC2 Instance in AWS (Ar
Cloud computing overview
ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-05bd020090b0ca76f&osUser=ec2-user&region=ap-south...
Search
[Alt+S]
Asia Pacific (Mumbai)
Sachchitanand Yadav
https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-12-157 ~]$
CloudShell
Actions
ap-south-1
~ $ ssh -i "new_key.pem" ec2-user@ec2-3-108-236-92.ap-south-1.compute.amazonaws.com
Warning: Identity file new_key.pem not accessible: No such file or directory.
The authenticity of host 'ec2-3-108-236-92.ap-south-1.compute.amazonaws.com (3.108.236.92)' can't be established.
ED25519 key fingerprint is SHA256:oM5G6yoKgYYUtY31vI8mNeyvqO0K0gsaNQW4rO1Om4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-108-236-92.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
ec2-user@ec2-3-108-236-92.ap-south-1.compute.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
~ $
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
27°C
Partly sunny
Search
ENG
IN
04:54 PM
02-12-2025

IAM User Policy Creation

IAM User Policy – Definition

An **IAM User Policy** is a permission document in AWS that specifies what actions an individual IAM user is allowed or not allowed to perform. It defines access to AWS services and resources and ensures that each user operates with only the permissions required for their role.

Purpose of IAM User Policies

IAM user policies define what a user can or cannot do inside AWS, helping maintain secure and controlled access to cloud resources.

Types of IAM Policies

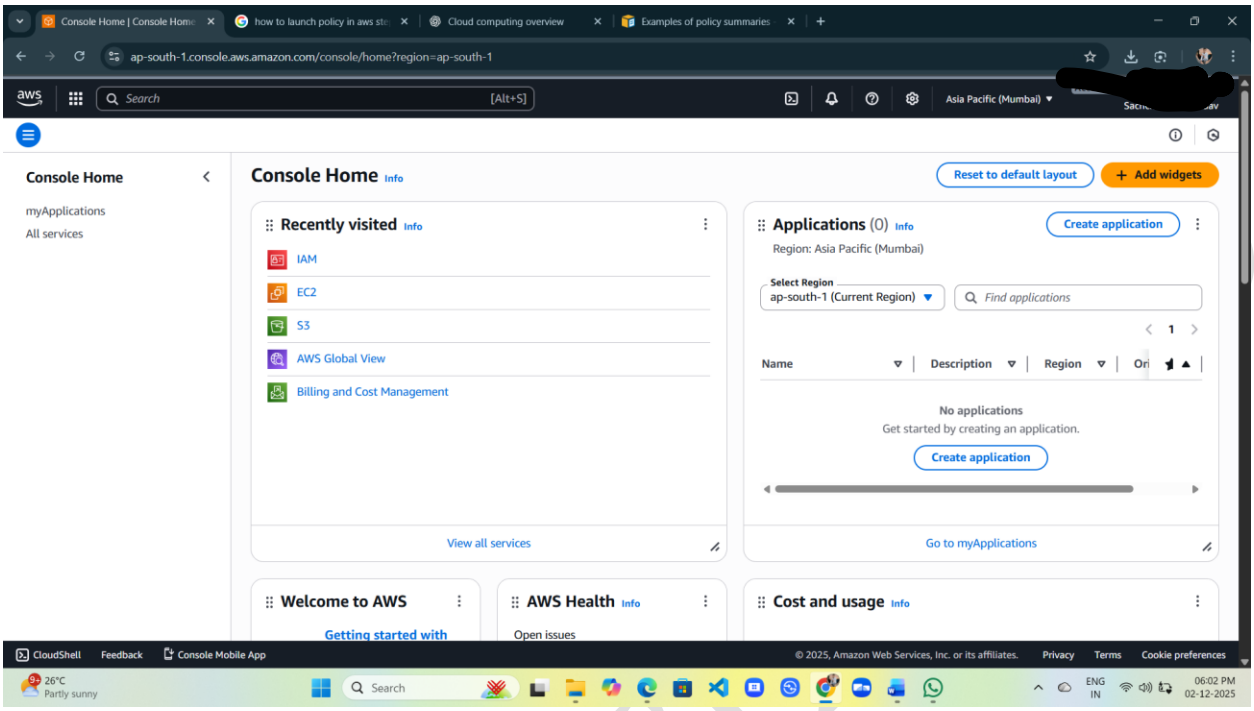
Inline Policies

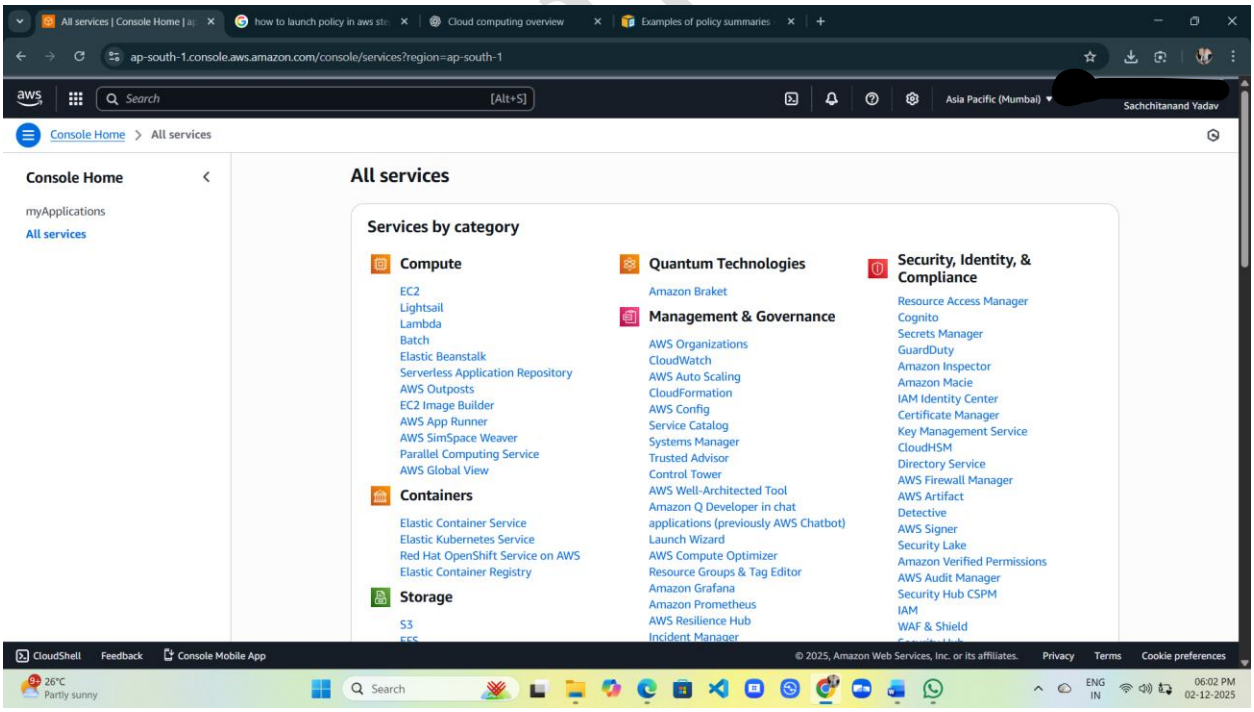
Inline policies are directly attached to one specific user, group, or role. They are custom and cannot be reused.

Managed Policies

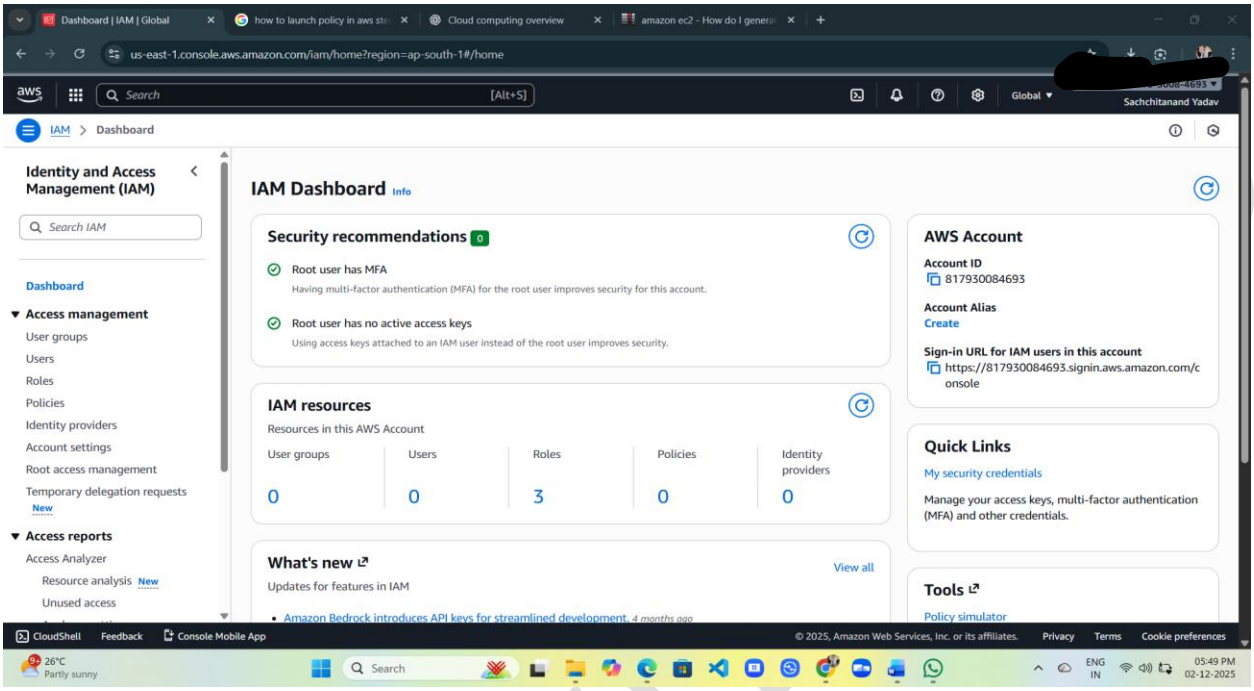
Managed policies are pre-built, reusable permission sets that can be attached to multiple users, making permission management easier and more consistent.

- **How to use it:**
    - **Navigate to IAM Users**
    - Log into the **AWS Management Console**.

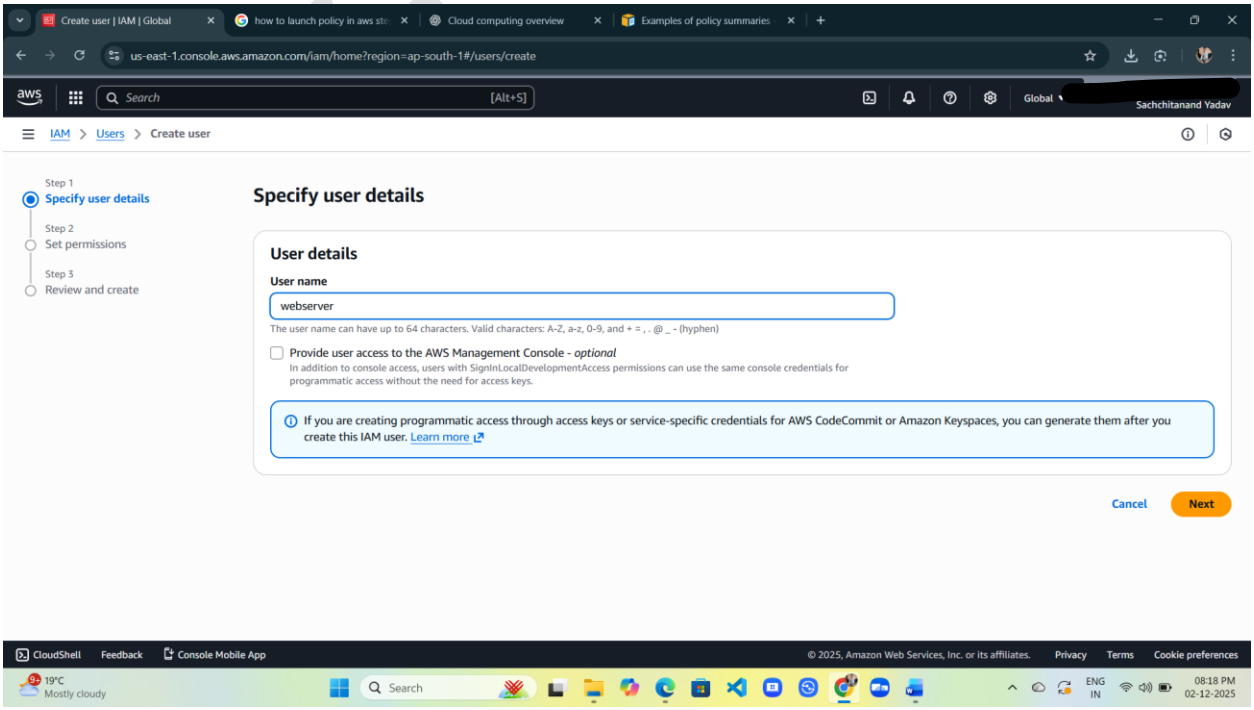

Console Home | Console Home
how to launch policy in aws ste
Cloud computing overview
Examples of policy summaries
ap-south-1.console.aws.amazon.com/console/home?region=ap-south-1
Search
[Alt+S]
Asia Pacific (Mumbai)
Console Home
myApplications
All services
Console Home Info
Reset to default layout
+ Add widgets
Recently visited Info
IAM
EC2
S3
AWS Global View
Billing and Cost Management
View all services
Applications (0) Info
Create application
Region: Asia Pacific (Mumbai)
Select Region
ap-south-1 (Current Region)
Find applications
Name
Description
Region
No applications
Get started by creating an application.
Create application
Go to myApplications
Welcome to AWS
Getting started with
AWS Health Info
Open issues
Cost and usage Info
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
26°C
Partly sunny
ENG
IN
06:02 PM
02-12-2025


All services | Console Home
how to launch policy in aws ste
Cloud computing overview
Examples of policy summaries
ap-south-1.console.aws.amazon.com/console/services?region=ap-south-1
Search
[Alt+S]
Asia Pacific (Mumbai)
Sachchitanand Yadav
Console Home > All services
Console Home
myApplications
All services
All services
Services by category
Compute
EC2
Lightsail
Lambda
Batch
Elastic Beanstalk
Serverless Application Repository
AWS Outposts
EC2 Image Builder
AWS App Runner
AWS SimSpace Weaver
Parallel Computing Service
AWS Global View
Containers
Elastic Container Service
Elastic Kubernetes Service
Red Hat OpenShift Service on AWS
Elastic Container Registry
Storage
S3
Quantum Technologies
Amazon Braket
Management & Governance
AWS Organizations
CloudWatch
AWS Auto Scaling
CloudFormation
AWS Config
Service Catalog
Systems Manager
Trusted Advisor
Control Tower
AWS Well-Architected Tool
Amazon Q Developer in chat applications (previously AWS Chatbot)
Launch Wizard
AWS Compute Optimizer
Resource Groups & Tag Editor
Amazon Grafana
Amazon Prometheus
AWS Resilience Hub
Incident Manager
Security, Identity, & Compliance
Resource Access Manager
Cognito
Secrets Manager
GuardDuty
Amazon Inspector
Amazon Macie
IAM Identity Center
Certificate Manager
Key Management Service
CloudHSM
Directory Service
AWS Firewall Manager
AWS Artifact
Detective
AWS Signer
Security Lake
Amazon Verified Permissions
AWS Audit Manager
Security Hub CSPM
IAM
WAF & Shield
CloudShell
Feedback
Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
26°C
Partly sunny
ENG
IN
06:02 PM
02-12-2025

- Navigate to the **IAM Dashboard**.



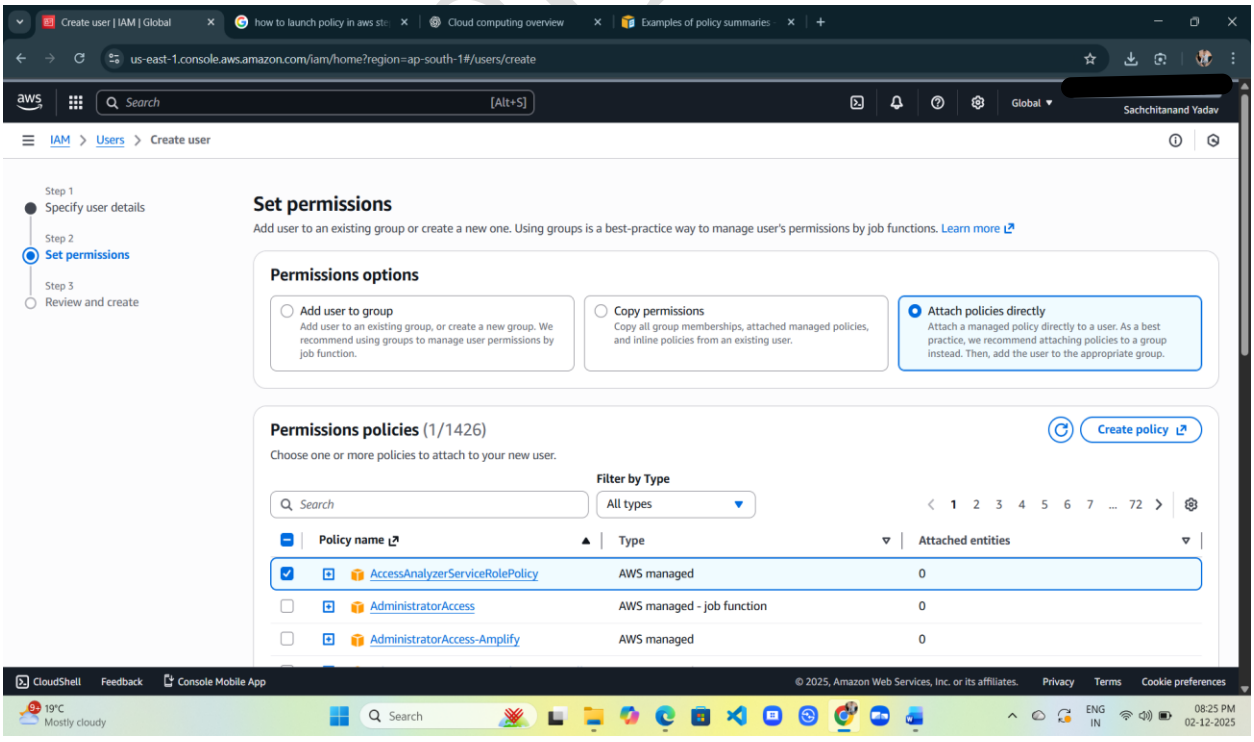
- In the navigation pane, click **"Users."**
- Select the **IAM User** to whom you want to grant permissions (e.g., SecurityAnalyst).



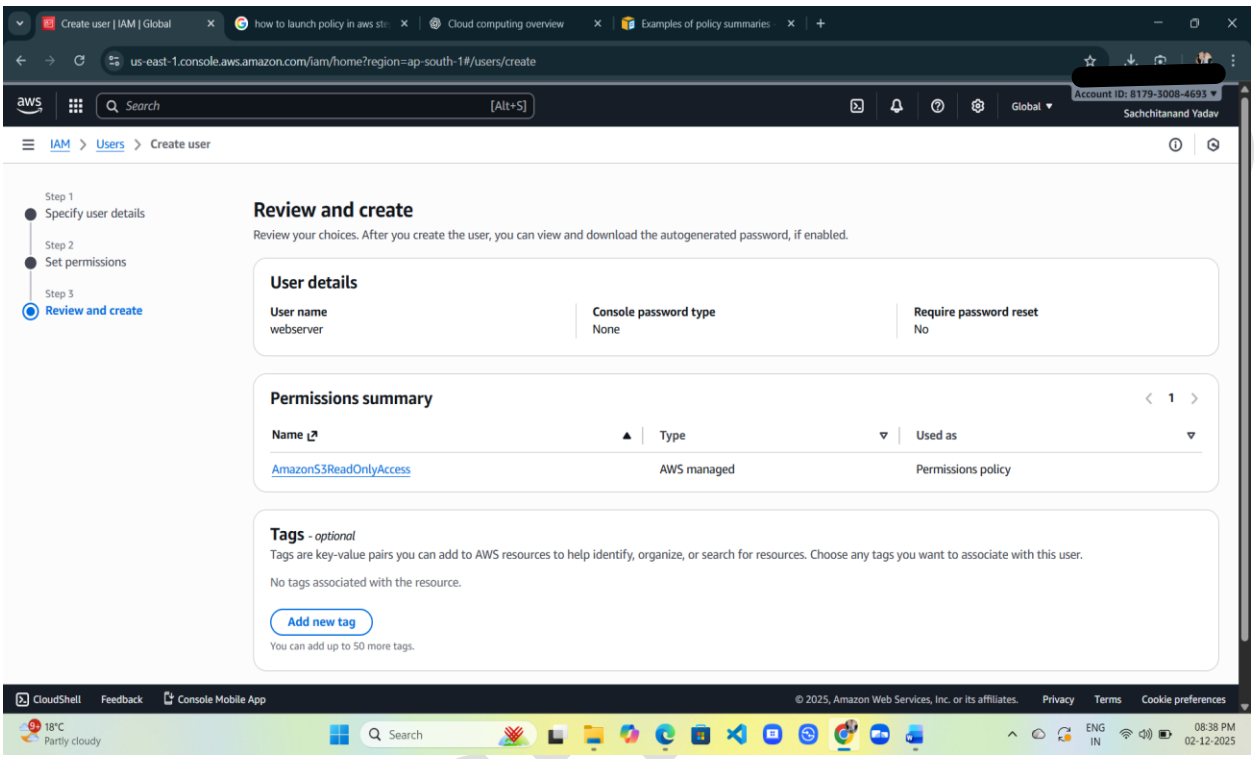
- On the User's summary page, click the **"Permissions"** tab.
- Click the **"Add permissions"** button (or **"Attach policies directly"** if using the older console layout).

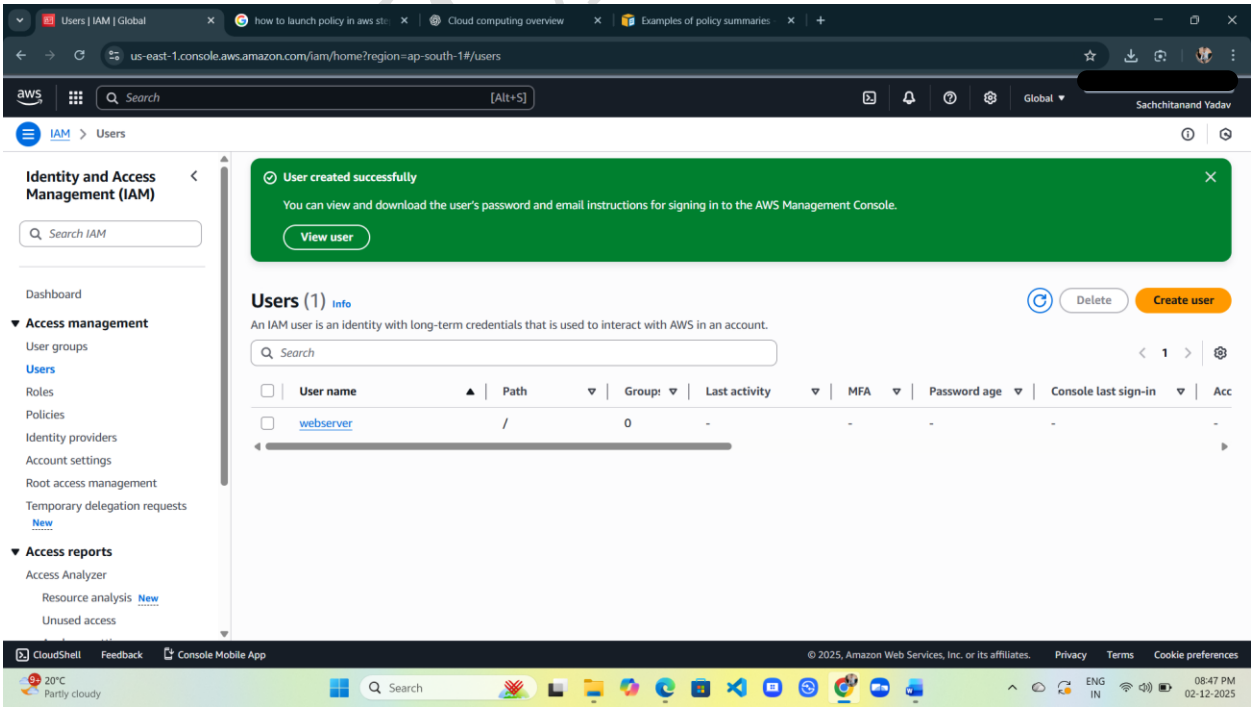
Select and Attach the Policy

- Select the option **"Attach policies directly."**
- In the search bar, filter for the specific policy you need.
    - *Sub-point (Managed Policy):* Select a pre-built AWS Managed Policy (e.g., AmazonS3ReadOnlyAccess).
    - *Sub-point (Custom Policy):* Select a Customer Managed Policy you created previously.
- **Check the box** next to the policy name(s) you wish to attach.Click **"Next: Review"** (if applicable) and then click **"Add permissions"** or **"Attach policy"** to finalize.



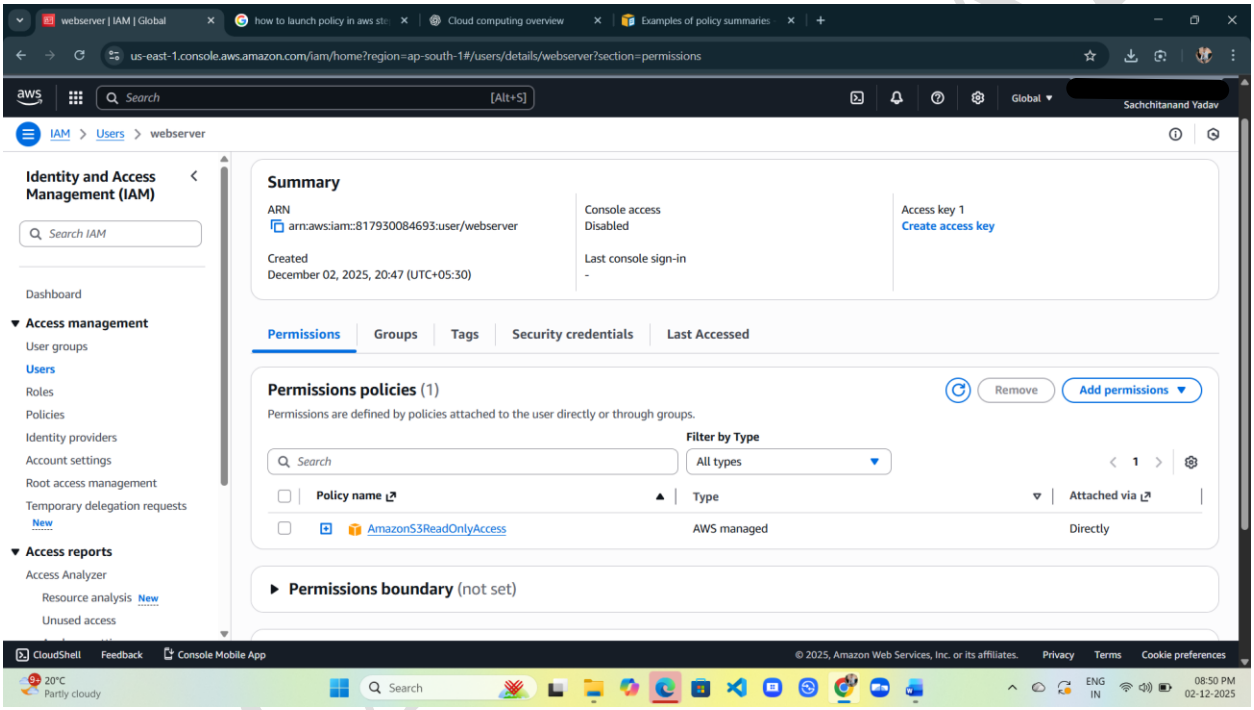
- Click **"Next: Review"** (if applicable) and then click **"Add permissions"** or **"Attach policy"** to finalize.





Verification

- Return to the User's **"Permissions"** tab.
- Verify that the newly attached policy appears in the list under **"Permissions policies"** and is active.
- Test the user's access to an AWS service to ensure the policy grants the intended permissions.



Cloud Bucket Enumeration Using LazyS3

LazyS3

LazyS3 is an open-source cloud reconnaissance tool designed to identify Amazon S3 (Simple Storage Service) buckets associated with a specific domain or keyword. It is commonly used during cloud security assessments to detect misconfigured or publicly accessible S3 buckets that may expose sensitive information.

Definition

LazyS3 is a Python-based reconnaissance tool that systematically generates and tests multiple S3 bucket name combinations related to a target domain in order to discover exposed or improperly secured AWS S3 buckets.

Purpose and Use

LazyS3 assists security researchers and penetration testers in mapping an organization's cloud storage assets. By identifying open, restricted, or incorrectly configured S3 buckets, the tool helps highlight potential data leakage risks in cloud environments.

Key Features of LazyS3

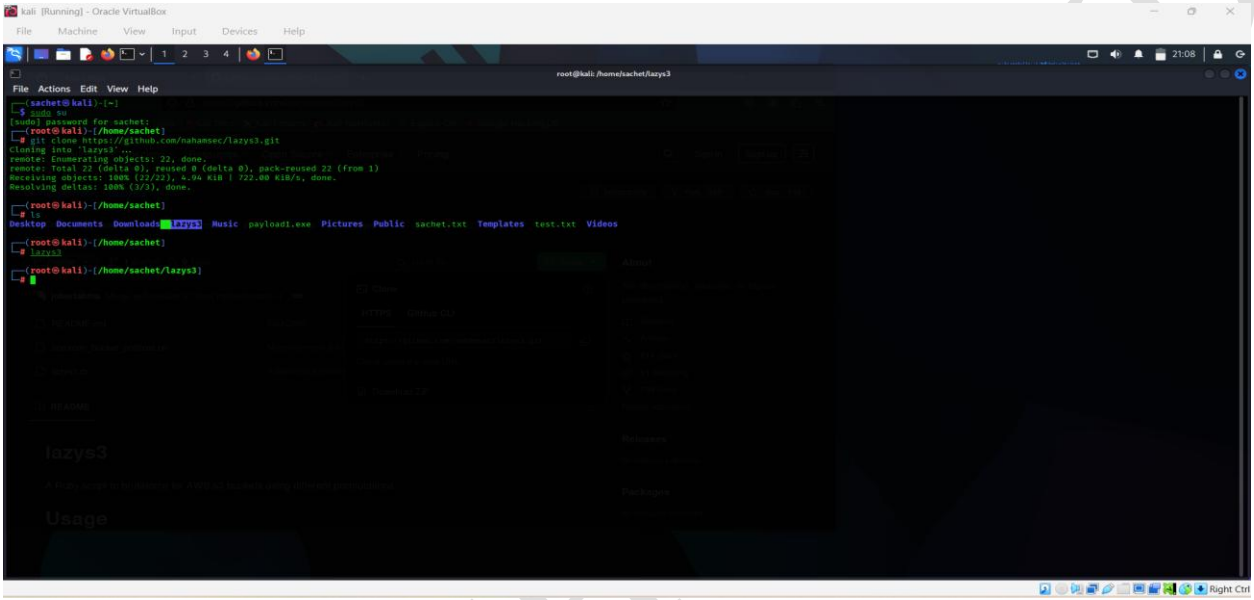
Feature	Description
Automated Bucket Enumeration	Uses wordlists and naming permutations to identify possible S3 bucket names
Domain-Based Targeting	Generates bucket names based on a given domain or keyword
Accessibility Validation	Determines whether buckets are public, restricted, or unavailable
Lightweight Tool	Simple command-line interface with minimal dependencies
Passive Reconnaissance	Performs non-intrusive discovery without modifying resources

Security Relevance

Misconfigured S3 buckets are a frequent cause of cloud data breaches. LazyS3 helps detect such weaknesses early, enabling organizations to secure exposed storage resources before they are exploited by malicious actors.

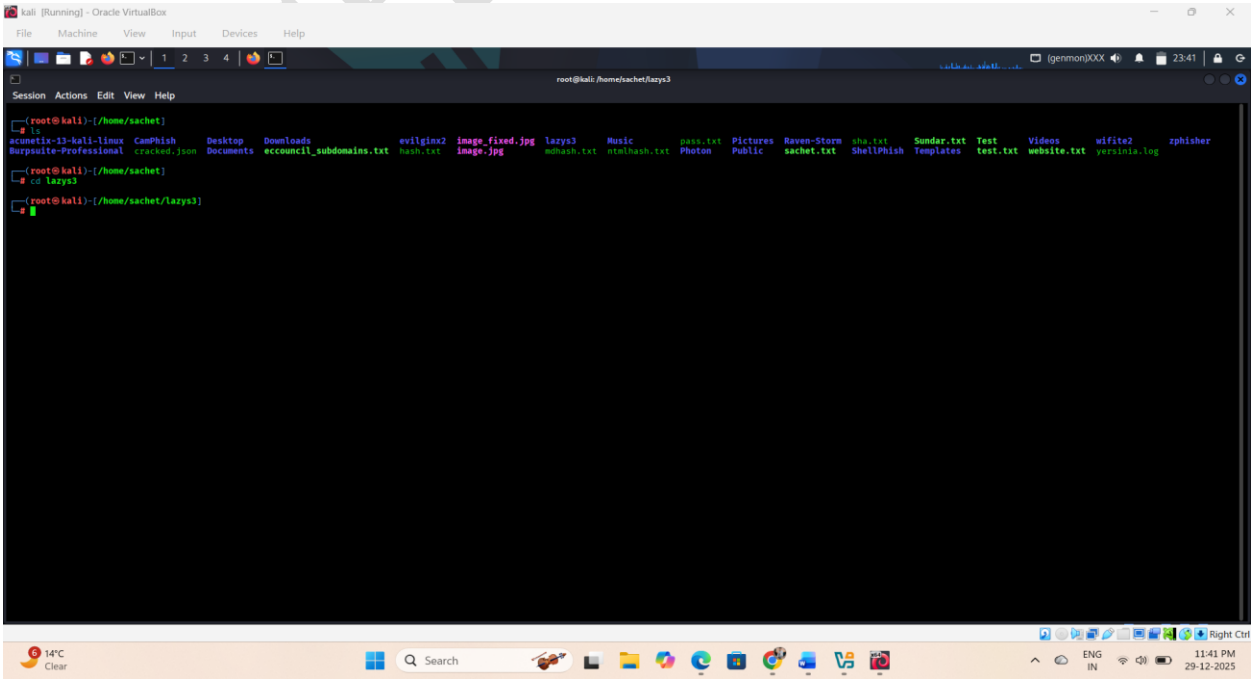
How to Download it:

- pen Kali Linux browser and search *lazys3 GitHub*
- Click on the first link
- Lazys3 downloaded

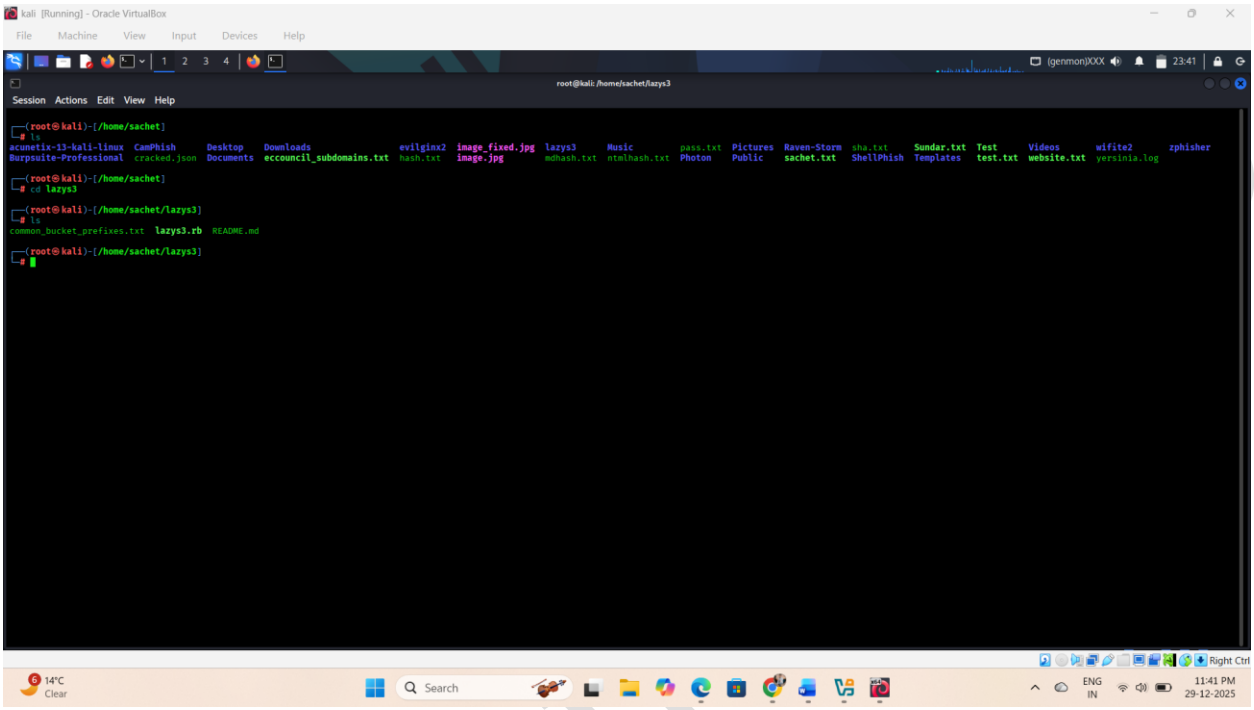


How to use -

- Go to the lazys3 directory



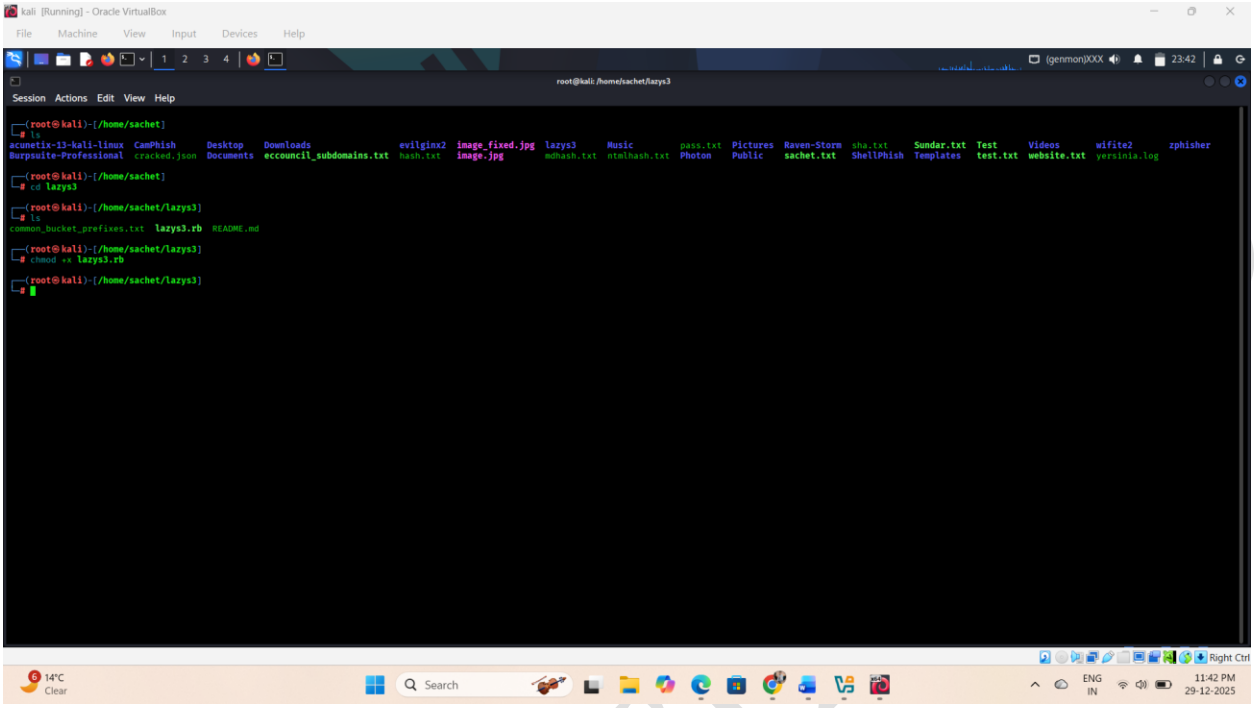
- **Type ls –** to see files of lazys3



Command - chmod +x lazys3.rb

Explanation –

- chmod: Stands for "change mode", a command used to modify file permissions.
- +x: Adds execute permission to the file (for the user, group, or others).
- lazys3.rb: This is the file name, in this case, a Ruby script (.rb).



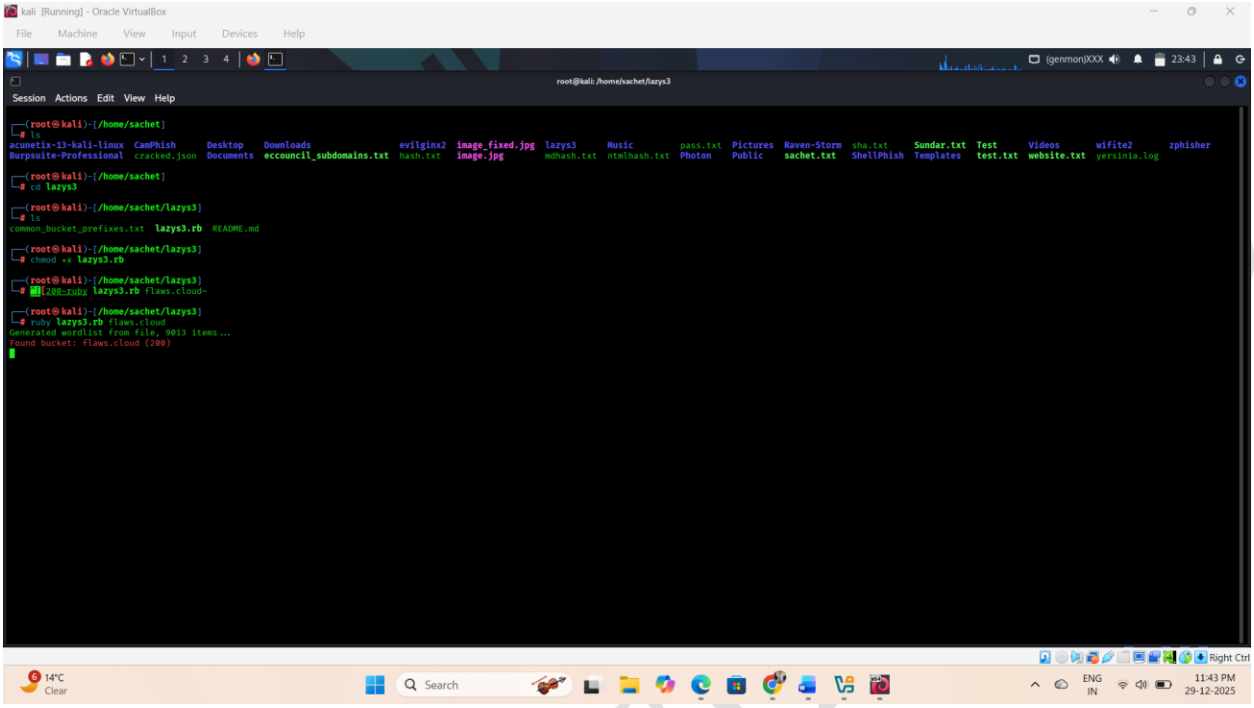
command - ruby lazys3.rb flaws.cloud

explanation -

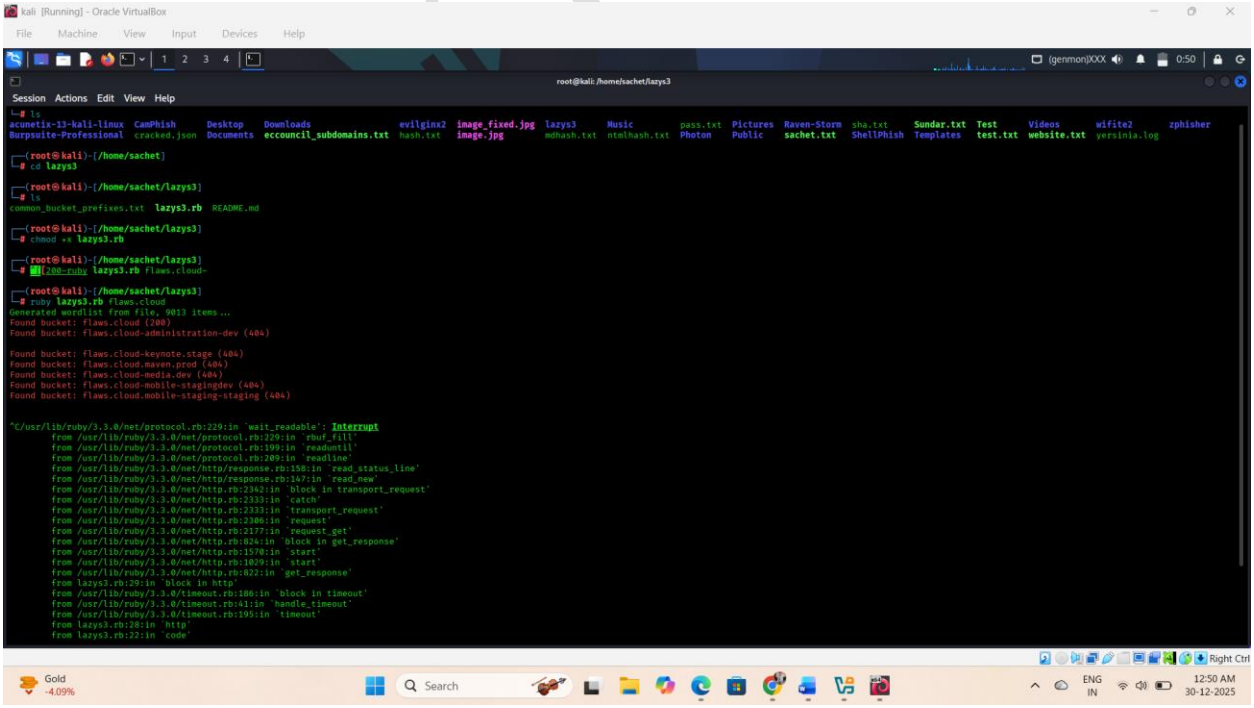
- It enumerates possible S3 bucket names related to flaws.cloud.
- It uses permutations, prefixes, and suffixes to generate potential S3 bucket names (e.g., flaws, flaws-dev, flaws-prod, etc.).
- Then, it checks if these buckets:

  - **Exist**
  - Are publicly accessible
  - Are restricted or forbidden

- Bucket found


kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
root@kali: /home/sachet/lazys3
Session Actions Edit View Help
(root㉿kali)-[/home/sachet]
ls
acunetix-13-kali-linux CamPhish Desktop Downloads evilginx2 image_fixed.jpg lazys3 Music pass.txt Pictures Raven-Storm sha.txt Sundar.txt Test Videos wifite2 zphisher
Burpsuite-Professional cracked.json Documents eccouncil_subdomains.txt hash.txt image.jpg md5hash.txt ntmlhash.txt Photon Public sachet.txt ShellPhish Templates test.txt website.txt yersinia.log
(root㉿kali)-[/home/sachet]
cd lazys3
(root㉿kali)-[/home/sachet/lazys3]
ls
common_bucket_prefixes.txt lazys3.rb README.md
(root㉿kali)-[/home/sachet/lazys3]
chmod +x lazys3.rb
(root㉿kali)-[/home/sachet/lazys3]
[200-ruby lazys3.rb flaws.cloud~
(root㉿kali)-[/home/sachet/lazys3]
ruby lazys3.rb flaws.cloud
Generated wordlist from file, 9013 items ...
Found bucket: flaws.cloud (200)
14°C Clear
Search
ENG IN
11:43 PM 29-12-2025
Right Ctrl

- **Result**


kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
root@kali: /home/sachet/lazys3
Session Actions Edit View Help
Found bucket: flaws.cloud (200)
Found bucket: flaws.cloud-administration-dev (404)
Found bucket: flaws.cloud-keynote.stage (404)
Found bucket: flaws.cloud.maven.prod (404)
Found bucket: flaws.cloud-media.dev (404)
Found bucket: flaws.cloud-mobile-stagingdev (404)
Found bucket: flaws.cloud.mobile-staging-staging (404)
^C/usr/lib/ruby/3.3.0/net/protocol.rb:229:in `wait_readable': Interrupt
from /usr/lib/ruby/3.3.0/net/protocol.rb:229:in `rbuf_fill'
from /usr/lib/ruby/3.3.0/net/protocol.rb:199:in `readuntil'
from /usr/lib/ruby/3.3.0/net/protocol.rb:209:in `readline'
from /usr/lib/ruby/3.3.0/net/http/response.rb:158:in `read_status_line'
from /usr/lib/ruby/3.3.0/net/http/response.rb:147:in `read_new'
from /usr/lib/ruby/3.3.0/net/http.rb:2342:in `block in transport_request'
from /usr/lib/ruby/3.3.0/net/http.rb:2333:in `catch'
from /usr/lib/ruby/3.3.0/net/http.rb:2333:in `transport_request'
from /usr/lib/ruby/3.3.0/net/http.rb:2306:in `request'
from /usr/lib/ruby/3.3.0/net/http.rb:2177:in `request_get'
from /usr/lib/ruby/3.3.0/net/http.rb:824:in `block in get_response'
from /usr/lib/ruby/3.3.0/net/http.rb:1570:in `start'
from /usr/lib/ruby/3.3.0/net/http.rb:1029:in `start'
from /usr/lib/ruby/3.3.0/net/http.rb:822:in `get_response'
from lazys3.rb:29:in `block in http'
from /usr/lib/ruby/3.3.0/timeout.rb:186:in `block in timeout'
from /usr/lib/ruby/3.3.0/timeout.rb:41:in `handle_timeout'
from /usr/lib/ruby/3.3.0/timeout.rb:195:in `timeout'
from lazys3.rb:28:in `http'
from lazys3.rb:22:in `code'
Gold -4.09%
Search
ENG IN
12:50 AM 30-12-2025
Right Ctrl

S3Scanner

S3Scanner

S3Scanner is an open-source reconnaissance and enumeration tool used to identify Amazon S3 (Simple Storage Service) buckets that are publicly accessible or improperly configured. It is widely used by security professionals, penetration testers, and bug bounty researchers during cloud security assessments.

Definition

S3Scanner is a command-line tool that scans a list of potential Amazon S3 bucket names to determine whether the buckets exist and to identify their access permissions, such as public read, list, or write access.

Purpose

The primary purpose of S3Scanner is to detect exposed S3 buckets that may lead to unauthorized data disclosure or data manipulation due to weak or incorrect access controls.

Key Features of S3Scanner

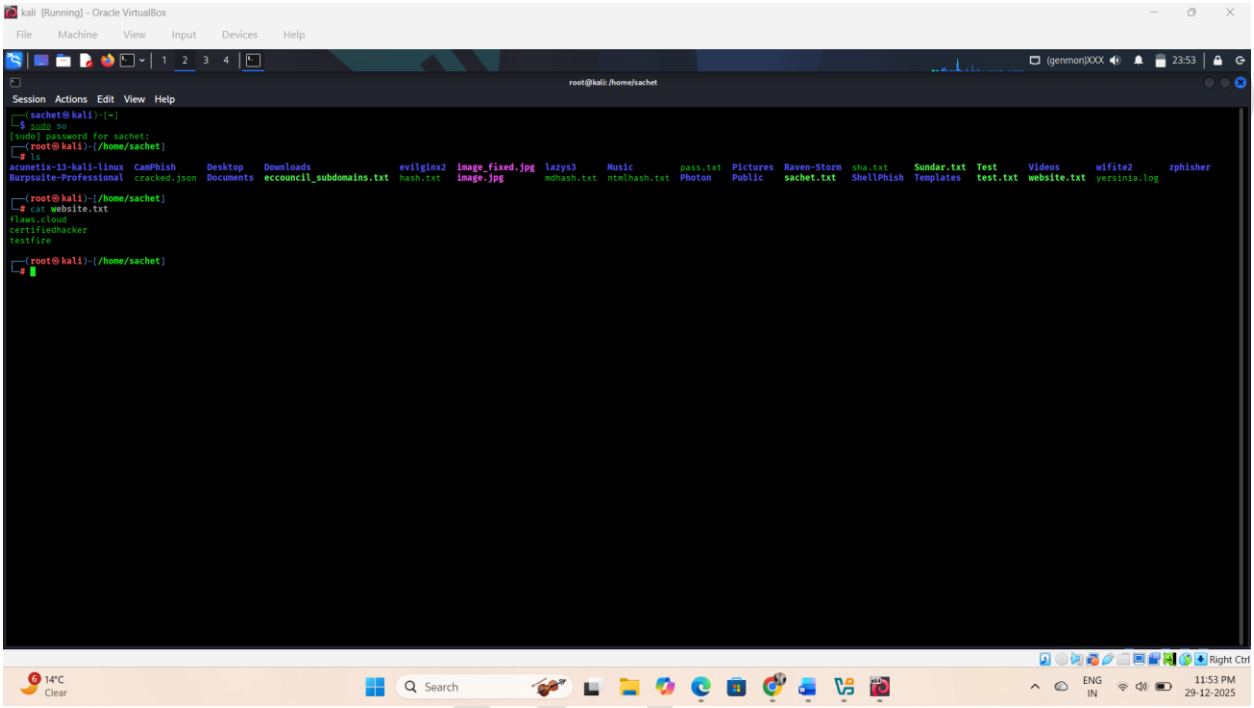
- Performs fast enumeration of large numbers of Amazon S3 bucket names
- Identifies publicly accessible S3 buckets (readable or listable)
- Detects misconfigured buckets using HTTP status codes (200, 403, 404)
- Supports custom wordlists for targeted bucket scanning
- Optionally checks write or upload permissions (when enabled)
- Works as a lightweight, command-line based tool
- Useful for passive cloud reconnaissance and security assessments

Security Significance

Publicly exposed S3 buckets are a common source of cloud data breaches. S3Scanner helps identify such misconfigurations early, enabling organizations to restrict access and secure cloud storage resources effectively.

How to use -

- I have already one txt file that contain 3 websites



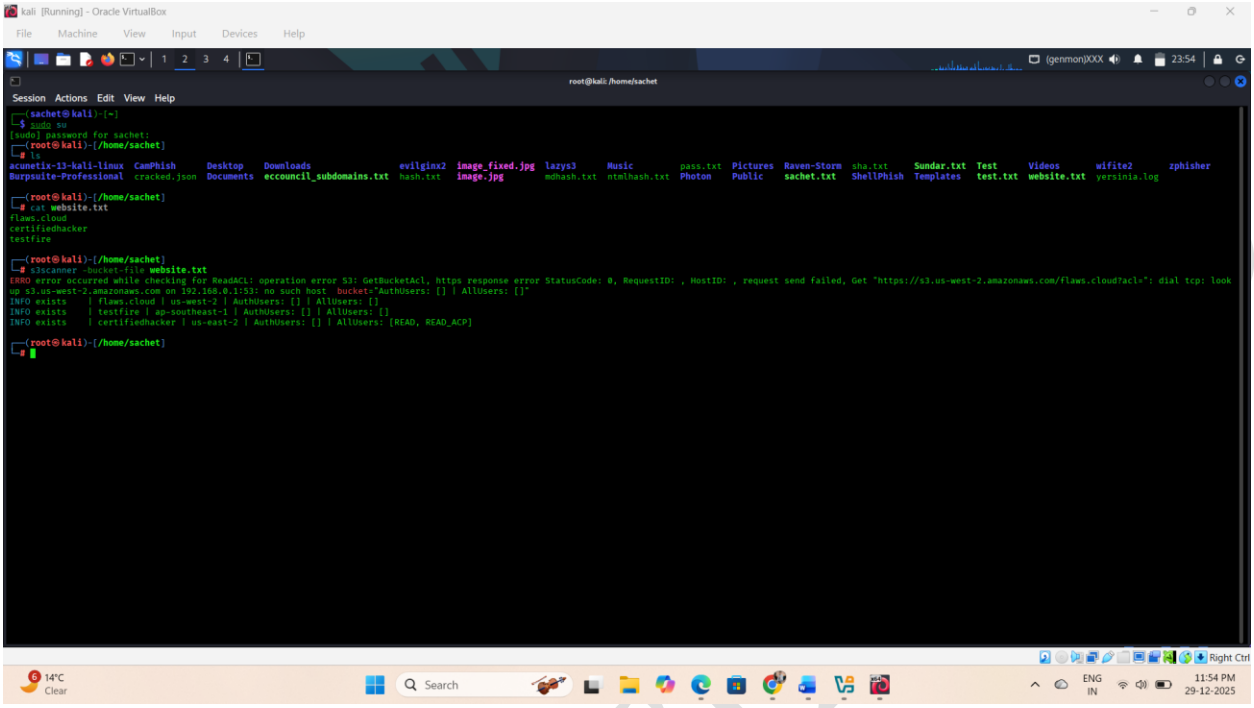
Command - s3scanner -bucket-file website.txt

Explanation -

This command tells S3Scanner to:

> **Read a list of S3 bucket names from a file called website.txt and check which buckets exist and are public.**

S3Scanner identified three existing Amazon S3 buckets. The buckets **flaws.cloud** and **certifiedhacker** were found to be publicly readable, while **testfire** exists but is configured as private. Additionally, the **certifiedhacker** bucket was observed to expose its access control policy (READ_ACP).



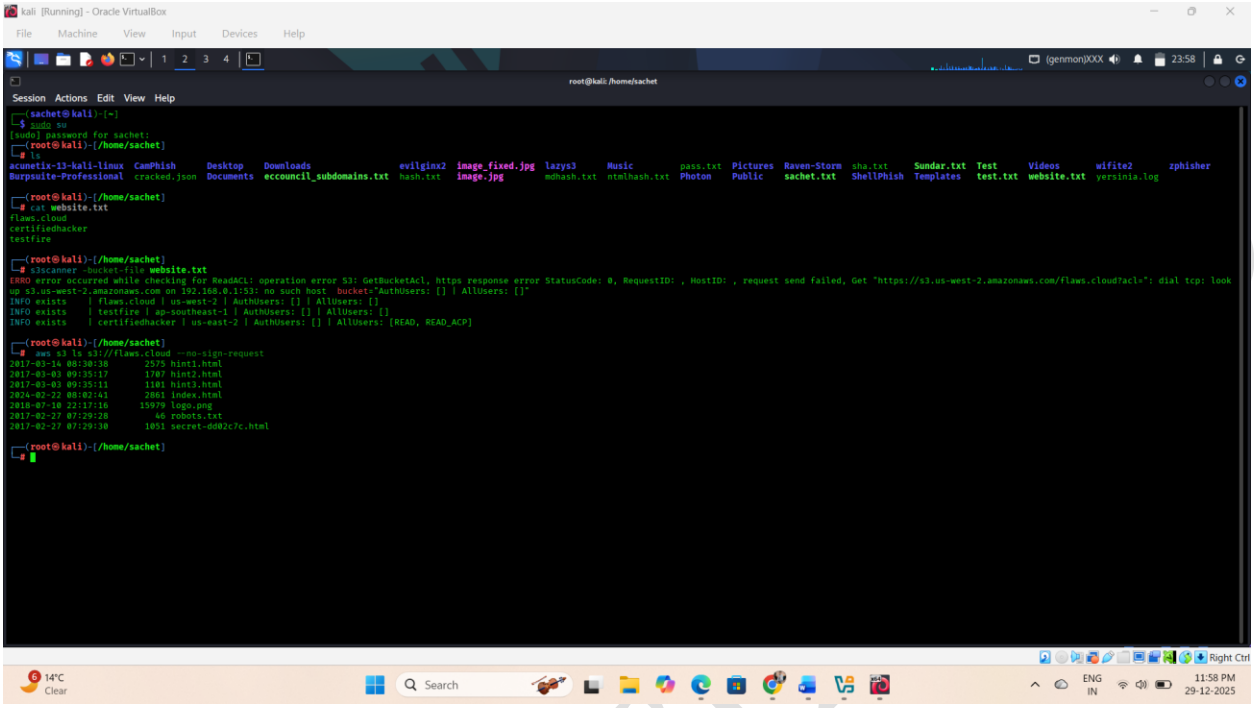
Command - aws s3 ls s3://flaws.cloud --no-sign-request

Explanation-

This command is used to list the contents of the publicly accessible Amazon S3 bucket named **flaws.cloud**. The `--no-sign-request` option indicates that no AWS credentials are required, confirming that the bucket allows unauthenticated access. If the bucket is publicly accessible, the command displays the available files and directories stored within it.

- **Result –**

You found 7 public files in the S3 bucket flaws.cloud.


kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
root@kali: /home/sachet
Session Actions Edit View Help
(sachet㉿kali)-[~]
$ sudo su
[sudo] password for sachet:
(root㉿kali)-[/home/sachet]
ls
acunetix-13-kali-linux CamPhish Desktop Downloads evilginx2 image_fixed.jpg lazys3 Music pass.txt Pictures Raven-Storm sha.txt Sundar.txt Test Videos wifite2 zphisher
Burpsuite-Professional cracked.json Documents eccouncil_subdomains.txt hash.txt image.jpg mdhash.txt ntmlhash.txt Photon Public sachet.txt ShellPhish Templates test.txt website.txt yersinia.log
(root㉿kali)-[/home/sachet]
cat website.txt
flaws.cloud
certifiedhacker
testfire
(root㉿kali)-[/home/sachet]
s3scanner -bucket-file website.txt
ERRO error occurred while checking for ReadACL: operation error S3: GetBucketAcl, https response error StatusCode: 0, RequestID: , HostID: , request send failed, Get "https://s3.us-west-2.amazonaws.com/flaws.cloud?acl=": dial tcp: look up s3.us-west-2.amazonaws.com on 192.168.0.1:53: no such host bucket="AuthUsers: [] | AllUsers: []"
INFO exists | flaws.cloud | us-west-2 | AuthUsers: [] | AllUsers: []
INFO exists | testfire | ap-southeast-1 | AuthUsers: [] | AllUsers: []
INFO exists | certifiedhacker | us-east-2 | AuthUsers: [] | AllUsers: [READ, READ_ACP]
(root㉿kali)-[/home/sachet]
aws s3 ls s3://flaws.cloud --no-sign-request
2017-03-14 08:30:38 2575 hint1.html
2017-03-03 09:35:17 1707 hint2.html
2017-03-03 09:35:11 1101 hint3.html
2024-02-22 08:02:41 2861 index.html
2018-07-10 22:17:16 15979 logo.png
2017-02-27 07:29:28 46 robots.txt
2017-02-27 07:29:30 1051 secret-dd02c7c.html
(root㉿kali)-[/home/sachet]
#
14°C Clear
Search
ENG IN
11:58 PM 29-12-2025
Right Ctrl

Vulnerability Assessment on Docker Images using Trivy

Vulnerability assessment of Docker images is a critical security practice used to identify known weaknesses before containerized applications are deployed into production. Since Docker images often include operating systems, libraries, and third-party packages, any unpatched vulnerability within them can be exploited by attackers.

Trivy is widely used for this purpose due to its speed, accuracy, and ease of integration into development and security workflows.

Objective

The objective of performing vulnerability assessment on Docker images using Trivy is to detect known security vulnerabilities (CVEs), misconfigurations, and outdated packages present in container images, thereby reducing the attack surface of containerized applications.

How Trivy Performs Docker Image Scanning

Trivy scans Docker images by analyzing the operating system packages and application dependencies included within the image. It compares these components against continuously updated vulnerability databases to identify known security issues.

The scanning process includes:

- Detecting the base operating system used in the image
- Identifying installed packages and libraries
- Matching packages with known CVEs
- Reporting severity levels such as LOW, MEDIUM, HIGH, and CRITICAL

Steps Involved in Vulnerability Assessment

1. A Docker image is selected for assessment, either from a local system or a public registry.
2. Trivy analyzes the image layers and extracts package information.
3. The extracted data is compared with vulnerability databases.
4. Identified vulnerabilities are categorized based on severity.
5. A detailed vulnerability report is generated for review and remediation.

Key Findings from Trivy Scan

Trivy provides detailed information for each detected vulnerability, including:

- Vulnerability ID (CVE)
- Affected package name and version
- Severity level
- Short description of the vulnerability
- Recommended remediation (upgrade or patch)

Importance of Docker Image Vulnerability Assessment

Regular vulnerability assessment of Docker images helps in:

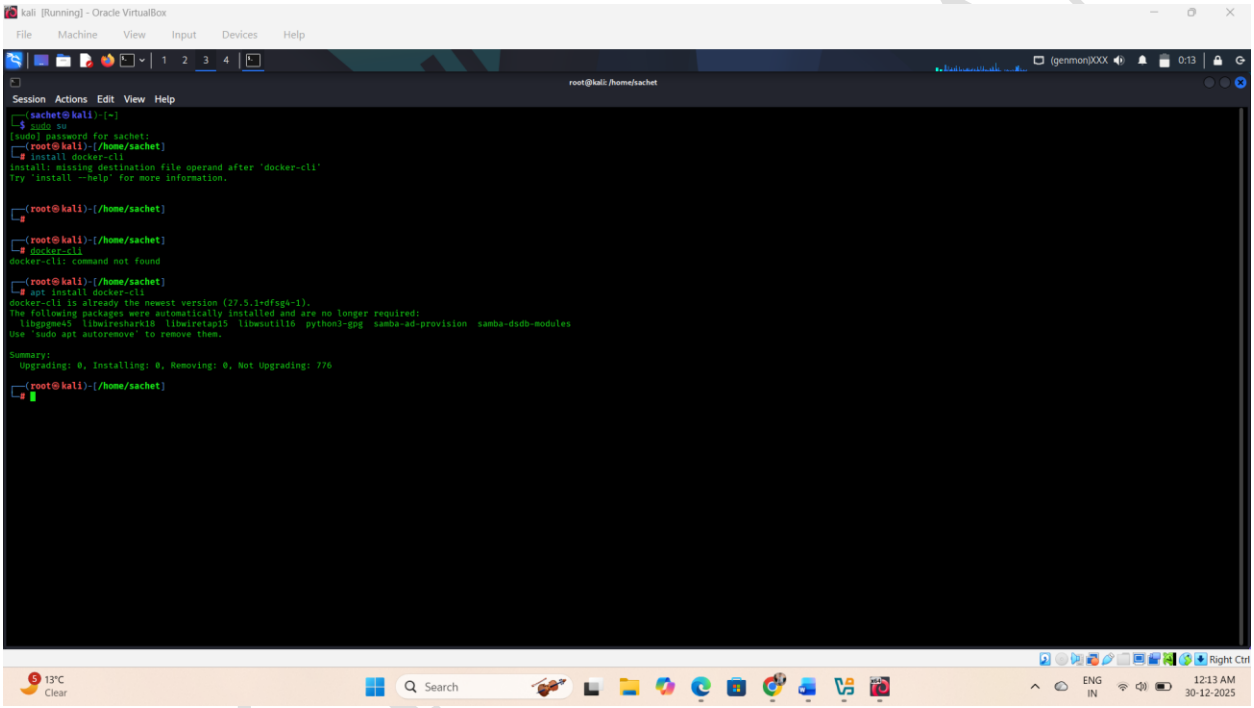
- Preventing deployment of vulnerable containers
- Reducing the risk of container compromise
- Improving overall cloud and application security
- Meeting security compliance requirements
- Supporting secure DevSecOps practices

How to use -

Command - install docker-cli

Explanation-

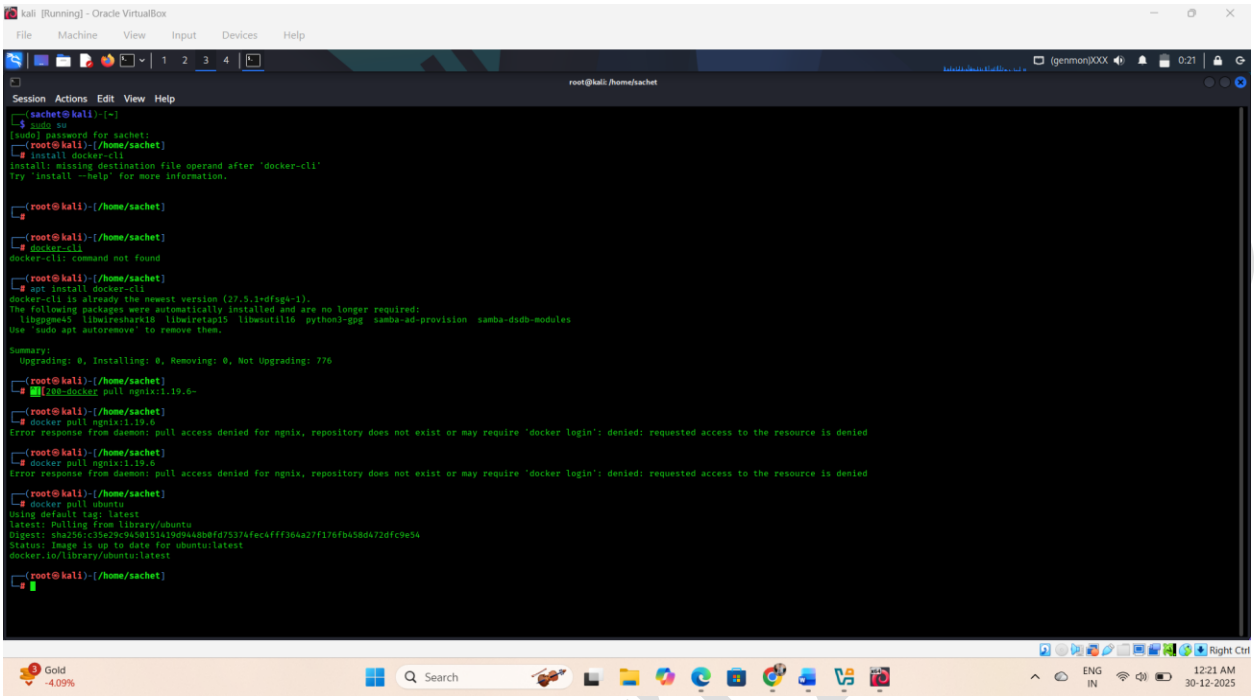
This command installs only the Docker Command Line Interface (CLI). It allows the user to execute Docker commands such as `docker pull`, `docker run`, and other Docker-related operations on the system without installing the full Docker engine.



Command - docker pull ngnix:1.19.6

Explanation -

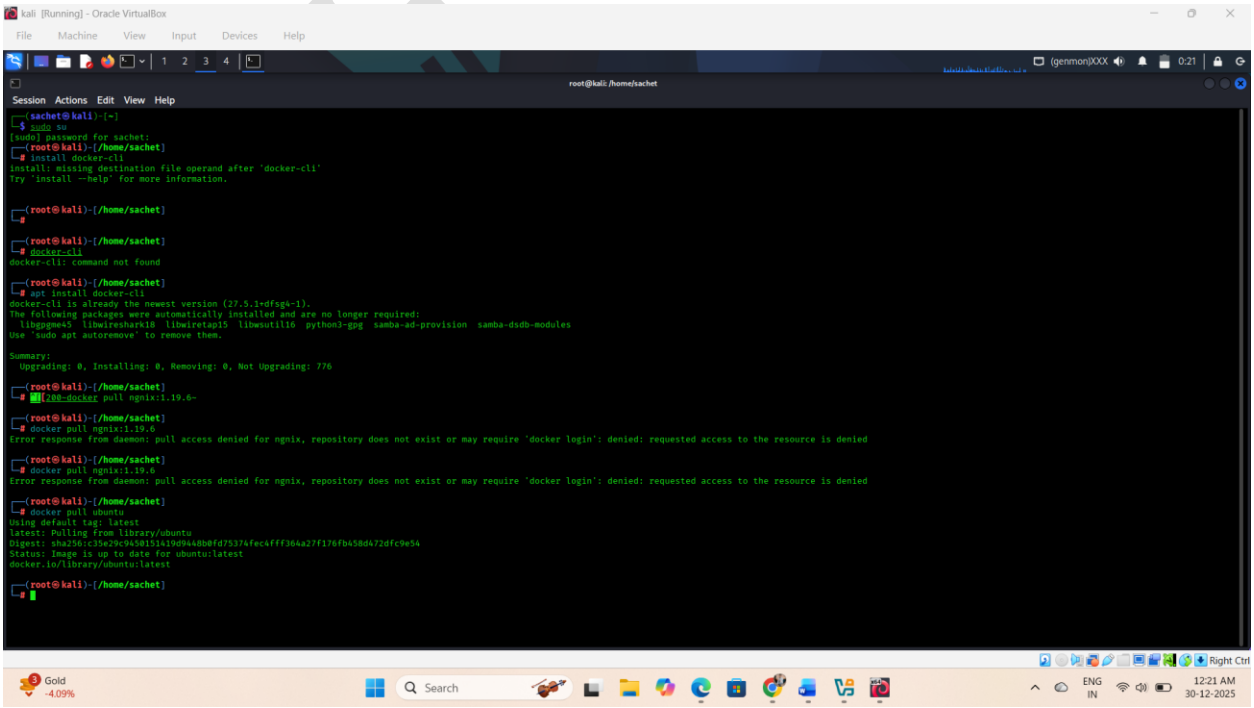
This command instructs Docker to download the Nginx web server image (version 1.19.6) from Docker Hub, which is the official public container image registry.



Command - docker pull ubuntu

Explanation -

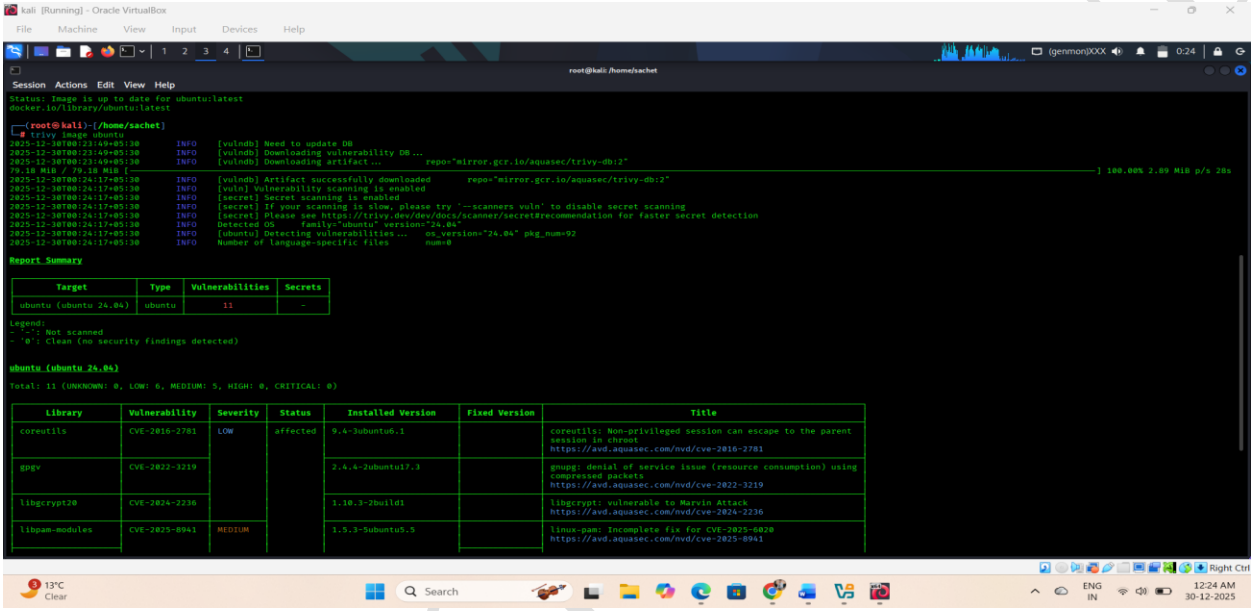
This command downloads the official Ubuntu Linux image from Docker Hub so you can run Ubuntu inside a container.



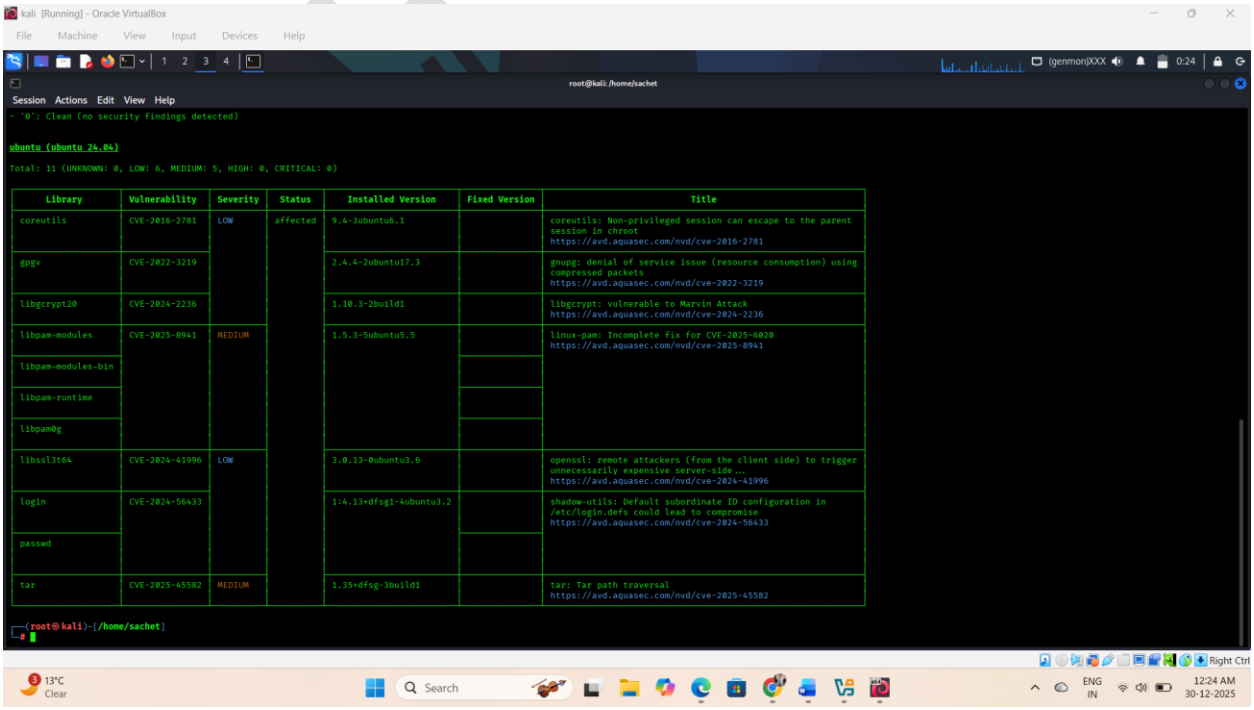
Command -: trivy image ubuntu

Explanation-

This command uses **Trivy** to scan the **Ubuntu Docker image** for security vulnerabilities, including known **Common Vulnerabilities and Exposures (CVEs)** and outdated or insecure software packages present within the image.



- **Result – vulnerabilities found**



Key Vulnerabilities Found:

1. **coreutils – [CVE-2016-2781] (LOW)**
    - Affects chroot environments; may allow a user to escape to the parent session.
2. **gpgv – [CVE-2022-3219]**
    - Denial of service possible using specially crafted compressed packets.
3. **libc-bin – [CVE-2016-20013]**
    - sha256/sha512 password hashing could be slow, allowing DoS (denial of service).
4. **libgcrypt20 – [CVE-2024-2236]**
    - Vulnerable to the Marvin Attack, a type of cryptographic side-channel attack.
5. **libpam-modules – [CVE-2024-10041 & CVE-2024-10963] (MEDIUM)**
    - One allows **reading hashed passwords,** the other could **bypass access control.**

Conclusion

Vulnerability assessment on Docker images using Trivy is an effective and efficient approach to strengthening container security. By identifying vulnerabilities early in the development lifecycle, organizations can remediate security issues before deployment, thereby minimizing potential exploitation in production environments.

Cloud Computing Countermeasures

Cloud threats don't need panic; they need discipline. Most attacks succeed not because the cloud is weak, but because humans get lazy. These countermeasures are how you stay sharp.

---

1. Identity and Access Management (IAM)

If everyone has the keys, nobody is safe.

Countermeasures:

- Apply **least privilege access** (only what's necessary)
- Enable **Multi-Factor Authentication (MFA)**
- Regularly **review and revoke unused accounts**
- Use **role-based access control (RBAC)**

👉 Old rule still holds: trust is earned, not assumed.

---

2. Data Protection & Encryption

Data is the crown jewel. Treat it like one.

Countermeasures:

- Encrypt data **at rest and in transit**
- Use **customer-managed encryption keys**
- Apply **data classification policies**
- Implement **secure data deletion**

If your data isn't encrypted, it's basically gossip.

---

3. Secure Configuration Management

Misconfiguration is the #1 cloud villain. Always has been.

Countermeasures:

- Follow **CIS cloud benchmarks**
- Disable **public access by default**

- Use **Infrastructure as Code (IaC)** with reviews
- Run **automated configuration audits**

Cloud doesn't fail—defaults do.

---

4. Network Security Controls

The cloud is open by nature. That's not an invitation.

Countermeasures:

- Use **Virtual Private Clouds (VPCs)**
- Apply **security groups and network ACLs**
- Deploy **Web Application Firewalls (WAF)**
- Segment workloads using **network isolation**

Moats still matter, even in the sky.

---

5. API and Application Security

APIs are doors. Lock them properly.

Countermeasures:

- Enforce **strong authentication (OAuth2, JWT)**
- Apply **rate limiting and throttling**
- Validate input to prevent injection attacks
- Use **API gateways**

If your API is wide open, attackers won't knock.

---

6. Monitoring, Logging & Visibility

If you can't see it, you can't stop it.

Countermeasures:

- Enable **centralized logging**
- Monitor with **SIEM tools**
- Configure **real-time alerts**

- Track **user and API activity**

Silence in logs is not peace—it's danger.

---

7. Workload & Container Security

Containers are fast, not fearless.

Countermeasures:

- Scan container images for vulnerabilities
- Use **trusted image registries**
- Apply **runtime protection**
- Enforce **pod security policies**

Tiny containers, massive impact if ignored.

---

8. Patch & Vulnerability Management

Old bugs never die—they just wait.

Countermeasures:

- Regularly patch OS and applications
- Automate vulnerability scanning
- Remove unused services and ports
- Track CVEs in third-party libraries

Legacy habits save modern systems.

---

9. Backup, Recovery & Availability

Hope is not a recovery strategy.

Countermeasures:

- Schedule **automated backups**
- Store backups in **separate regions**
- Test **disaster recovery plans**
- Use **DDoS protection services**

Downtime teaches lessons the hard way.

---

10. Compliance, Governance & Policies

Rules exist for a reason. Ignore them, pay later.

Countermeasures:

- Follow **ISO 27001, GDPR, HIPAA**
- Implement **cloud governance frameworks**
- Maintain audit trails
- Conduct regular security assessments

Module Summary

This module presented a comprehensive overview of cloud computing and its role in modern digital infrastructure. It covered fundamental cloud computing concepts, including various cloud service models and deployment types. The module also explored container technology and serverless computing environments, highlighting their importance in cloud-native application development.

In addition, the module examined major cloud computing threats and attack vectors, along with real-world cloud hacking techniques. These included attacks targeting Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), and containerized environments. The discussion emphasized how misconfigurations, weak access controls, and insecure workloads are commonly exploited by threat actors.

Furthermore, the module detailed multiple countermeasures and defensive strategies used to protect cloud environments from attacks. These included security best practices, access control mechanisms, monitoring techniques, and cloud security tools designed to strengthen overall cloud posture.

The module concluded with an overview of cloud security tools and protection techniques, setting the foundation for advanced security concepts.

In the next module, the focus will shift to cryptography, examining how attackers, ethical hackers, and penetration testers use cryptographic techniques to secure and protect data in cloud environments.

THANK YOU