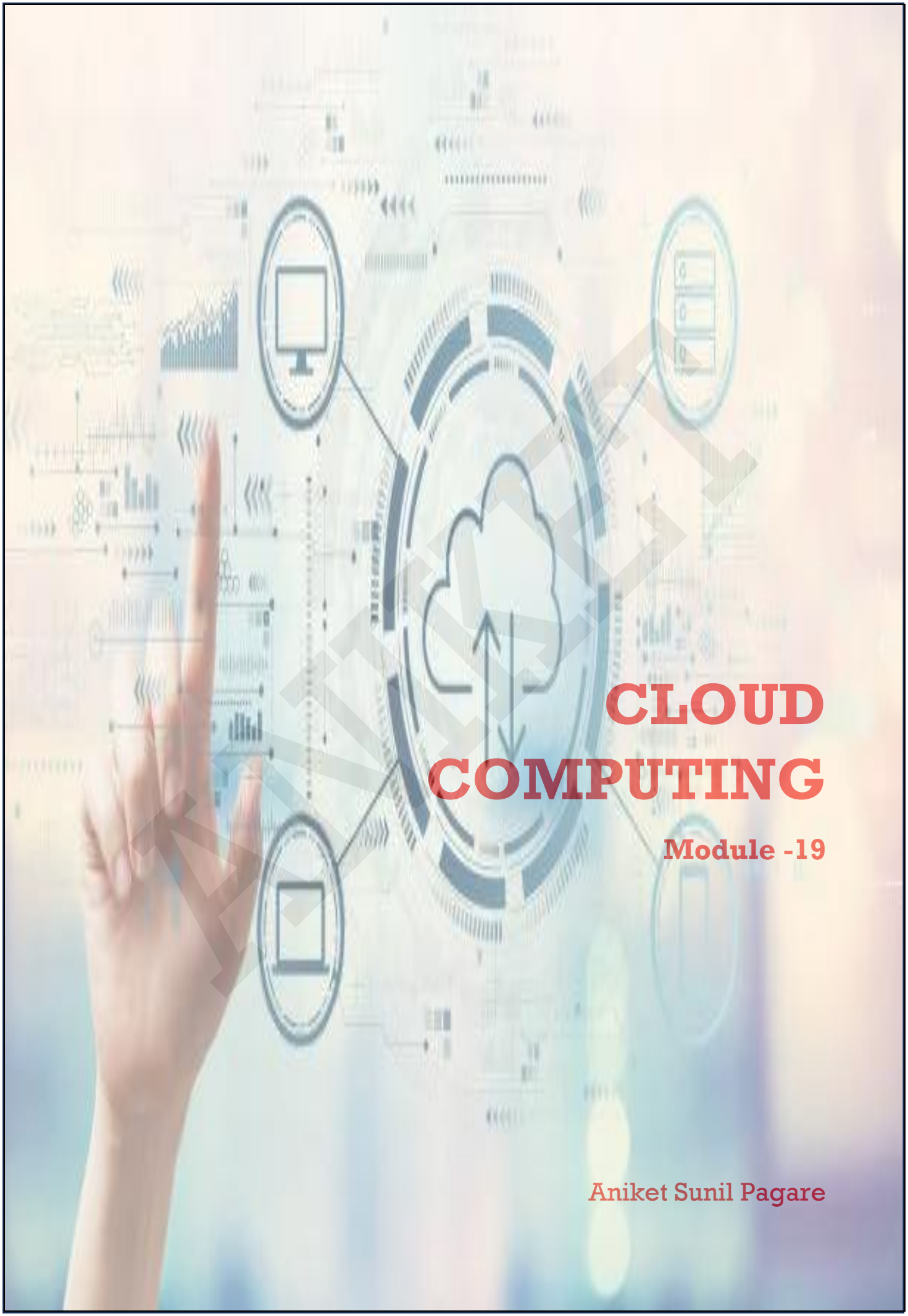


CLOUD COMPUTING

Module -19

Aniket Sunil Pagare

📘 Table of Contents

- 5.2 Key Characteristics of Serverless Computing

---

6. Cloud Security

- 6.1 OWASP Top 10 Cloud Security Risks
- 6.2 Common Cloud Computing Threats

---

7. Cloud Bucket Enumeration Using LazyS3

- 7.1 Definition
- 7.2 Key Features
- 7.3 How to Download LazyS3
- 7.4 Perform Bucket Enumeration with LazyS3

---

8. Cloud Reconnaissance and Security Testing with S3Scanner

- 8.1 Definition
- 8.2 Key Features
- 8.3 Perform Activity Using S3Scanner

---

9. Vulnerability Scanning of Container Images Using Trivy

- 9.1 Definition
- 9.2 What Trivy Can Scan
- 9.3 Perform Activity Using Trivy

Cloud Computing

Cloud computing is the delivery of **computing services** over the **internet ("the cloud")** to offer faster innovation, flexible resources, and economies of scale.

🧠 Simple Definition:

Cloud computing means storing, accessing, and processing data and programs over the internet instead of your computer's hard drive.

📦 Examples of Cloud Services:

- Watching videos on **YouTube** (streamed from cloud servers)
- Saving files to **Google Drive**
- Hosting a website on **Amazon Web Services (AWS)**

🔑 Key Components of Cloud Computing:

Component	Description
Compute	Virtual machines, containers (run apps)
Storage	Save data online (e.g., files, backups)
Networking	Connect cloud services globally
Security	Control access, encrypt data, firewall protections

📊 Benefits of Cloud Computing:

Benefit	Description
Cost-effective	Pay only for what you use
Scalable	Easily increase or decrease resources

Benefit	Description
Accessible	Use services from anywhere, anytime
Reliable	Built-in redundancy and backup
Managed Infrastructure	No need to maintain hardware

🧱 Real-World Analogy:

Think of **cloud computing** like **renting a house**:

- You don't build it (like buying servers)
- You pay monthly for use
- The landlord (cloud provider) handles maintenance

Types of Cloud Computing

Cloud computing is categorized based on **deployment models** and **service models**.

🏗️ A. Deployment Models (Where the cloud is deployed)

Type	Description	Example Use Case
Public Cloud	Services offered over the internet and shared among users	Hosting a website on AWS or Azure
Private Cloud	Cloud infrastructure for a single organization (on-premise or vendor-hosted)	Internal ERP systems of a bank
Hybrid Cloud	Combination of public and private cloud environments	Store sensitive data in private cloud, rest in public cloud

Type	Description	Example Use Case
Community Cloud	Shared among multiple organizations with common goals	Hospitals sharing a secure health cloud

---

🧰 B. Service Models (What type of service is offered)

Model	Full Form	What It Offers	Examples
IaaS	Infrastructure as a Service	Virtual machines, storage, networks	AWS EC2, Azure VMs
PaaS	Platform as a Service	Platform to build and deploy apps	Google App Engine, Heroku
SaaS	Software as a Service	Ready-to-use software over internet	Gmail, Dropbox, Zoom
FaaS	Function as a Service	Serverless execution of code functions	AWS Lambda, Azure Functions

---

📦 What is a Container in Cloud Computing?

A **container** is a lightweight, portable, and standalone executable package that includes everything needed to run an application:

- Code
- Runtime
- Libraries
- Dependencies

It allows applications to run **consistently across different computing environments** (like development, testing, and production).

🧠 Simple Definition:

A **container** is like a **sealed lunchbox** that has your food (app) along with the necessary ingredients (dependencies) so it tastes the same no matter where you eat it (run it).

🛠️ Key Features of Containers:

Feature	Description
Lightweight	Uses less memory and storage than VMs
Portable	Runs the same on any system (Windows, Linux, cloud)
Fast startup	Boots in seconds
Isolated	Runs in its own environment without affecting others
Scalable	Can be replicated quickly to handle more traffic

🔄 Containers vs Virtual Machines (VMs)

Feature	Container	Virtual Machine (VM)
Boot Time	Seconds	Minutes
Size	MBs	GBs
OS	Shares host OS	Has its own OS
Performance	Faster	Slower (resource-heavy)
Use Case	Microservices, DevOps	Full OS isolation, legacy applications

---

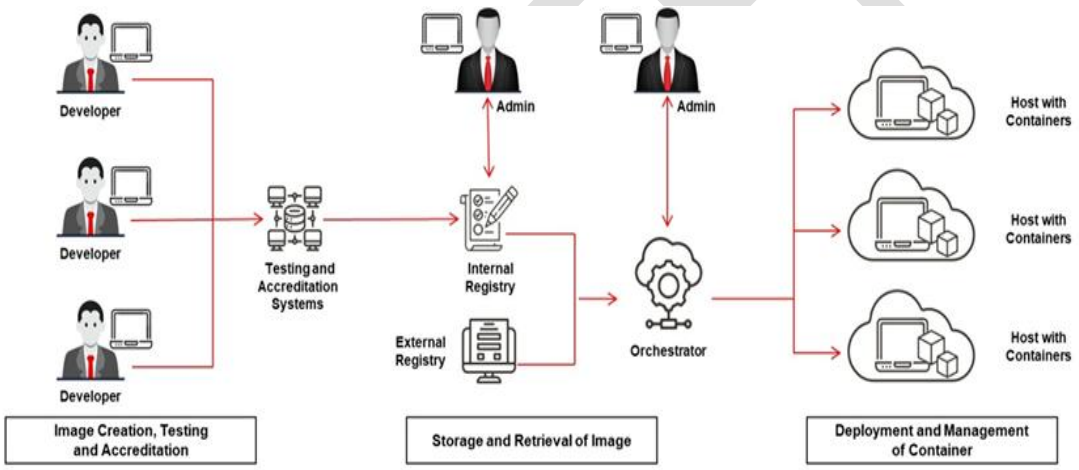
🐳 Popular Container Tools

- **Docker** 🐳 – Most widely used tool for container creation and management

- **Podman** – Docker-compatible but daemonless
- **containerd** – Core container runtime
- **CRI-O** – Kubernetes-native container runtime

☁️ Containers in the Cloud

- Containers are used in **cloud-native** apps
- Easily deploy apps using **Kubernetes**, **Amazon ECS**, **Azure AKS**, **Google GKE**
- Ideal for **CI/CD pipelines**, **microservices**, and **serverless architecture**



Architecture of container technology

🤿 What is Docker?

Docker is an open-source **platform used to build, package, run, and manage containers**.

It allows developers to bundle an application with all its **dependencies** into a **container**, so it runs **consistently across any environment** (your laptop, test server, or cloud).

🧠 Simple Definition:

Docker is like a **shipping container system for software**. It lets you package your app and ship it anywhere, and it will run exactly the same.

🔧 Why Docker is Popular

Feature	Description
Portability	Run the same container on any OS or cloud platform
Consistency	Eliminates "it works on my machine" issues
Efficiency	Uses fewer resources than traditional virtual machines
Fast Deployment	Containers start in seconds
Version Control	Dockerfiles can be versioned like code for consistent builds

---

🧱 Key Docker Components

Component	Description
Docker Engine	Core runtime that builds and runs containers
Docker Image	Blueprint or snapshot of a container (read-only template)
Docker Container	Running instance of an image (the actual app)
Dockerfile	Script that defines how to build a Docker image
Docker Hub	Public cloud registry to share and download Docker images

---

🐬 What is Docker Engine?

Docker Engine is the **core software component** of Docker that enables you to **build, run, and manage containers** on a system.
It acts as the **client-server architecture** where containers are created and managed efficiently.

🧠 Simple Definition:

Docker Engine is the **engine** under the hood that powers all Docker commands — like building images, running containers, and pulling from Docker Hub.

🧱 Docker Engine Components

Component	Description
Docker Daemon (dockerd)	Background service that builds, runs, and manages Docker containers
Docker Client (docker)	CLI tool you use to interact with Docker (e.g., docker run)
REST API	Allows the Docker client and third-party tools to communicate with the daemon

---

🔄 Architecture Diagram

Your Terminal

|

| docker build/run/pull commands

↓

Docker Client ---> Docker Daemon (dockerd) ---> Containers

↑

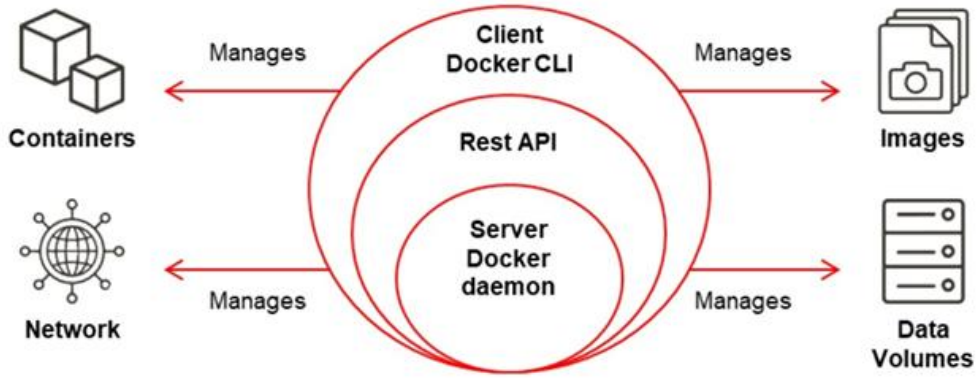
|

How Docker Engine Works

1. You run a command like:

```
docker run nginx
```

2. The **Docker client** sends this command to the **Docker daemon**.
3. The **Docker daemon**:
    - Pulls the nginx image (if not available locally)
    - Creates a **container** from it
    - Runs the container



Docker engine

What is Kubernetes?

Kubernetes (K8s) is an **open-source container orchestration platform** that automates the **deployment, scaling, and management** of containerized applications.

It was originally developed by **Google** and is now maintained by the **Cloud Native Computing Foundation (CNCF)**.

🧠 Simple Definition:

Kubernetes is like an **operating system for containers**. It helps you **manage lots of containers across multiple machines** without doing everything manually.

🚀 Why Kubernetes?

Containers like **Docker** are great — but managing hundreds or thousands of them becomes difficult.
Kubernetes solves that by:

Problem Without K8s	**Kubernetes Solves It By**
Manual container deployment	Automated deployment and rollout
Complex scaling	Auto-scaling based on demand
Failure recovery	Self-healing containers
Load balancing	Built-in service discovery and balancing

🧱 Key Components of Kubernetes

Component	**Description**
Pod	Smallest unit in K8s, holds one or more containers
Node	A worker machine (physical or virtual) that runs pods
Cluster	Group of nodes managed by Kubernetes
Deployment	Blueprint for managing replicas and rollout of pods
Service	Exposes a group of pods as a network service (internal or external)
Ingress	Manages external HTTP/HTTPS access to services

Component	Description
Namespace	Logical partitioning within the cluster
Kubelet	Agent that runs on each node and communicates with the control plane
kubectl	CLI tool to interact with Kubernetes clusters

📦 Kubernetes vs Docker

Feature	Docker	Kubernetes
Purpose	Build and run containers	Manage containerized applications
Scope	Single container runtime	Multi-container, multi-node orchestration
Scaling	Manual	Automatic
Networking	Basic	Advanced with Services and Ingress

🚨 Note: Docker is a container **runtime**, while Kubernetes is a **container orchestrator**.

☁️⚙️ What is Serverless Computing?

Serverless computing is a **cloud computing model** where the **cloud provider automatically manages** the infrastructure, so you can focus **only on writing code**.
You don't need to manage servers, patch OS, scale resources, or worry about uptime.

🧠 Simple Definition:

Serverless means **you write your code**, and the **cloud runs it automatically** — no need to manage any servers yourself.

🛠️ How It Works:

- You write a **function** (e.g., resizeImage()).
- You upload it to a **serverless platform** (like AWS Lambda).
- It runs **only when triggered** (e.g., file uploaded to S3).
- You **pay only when it runs**, not for idle time.

🧱 Key Characteristics

Feature	Description
No server management	Cloud handles provisioning, scaling, maintenance
Event-driven	Code runs in response to events (API calls, uploads, timers)
Scalable	Automatically scales up/down with traffic
Cost-efficient	Pay only for compute time used (no idle charges)

🔧 Popular Serverless Platforms

Platform	Function Service	Vendor
AWS	Lambda	Amazon
Azure	Azure Functions	Microsoft
GCP	Cloud Functions	Google
IBM Cloud	IBM Cloud Functions	IBM
Cloudflare	Workers	Cloudflare

🔐 OWASP Top 10 Cloud Security Risks

These are the most common and critical **cloud-specific security risks**, published by the **OWASP Cloud-Native Application Security Top 10** (2023 update), tailored for cloud environments (IaaS, PaaS, SaaS).

☁️ OWASP Top 10 Cloud Security Risks (CNAS-2023)

Rank	Risk Code	Title	Summary
1	**CNA-01**	**Cloud Provider Misconfiguration**	Misconfigured storage, compute, IAM, or networking leads to data leaks or compromise.
2	**CNA-02**	**Inadequate Change Management**	Untracked or unauthorized changes to cloud environments can introduce vulnerabilities.
3	**CNA-03**	**Over-Privileged Access and Identity Risks**	Excessive permissions or poor IAM practices can lead to account takeover or privilege escalation.
4	**CNA-04**	**Insecure APIs**	Poorly secured cloud APIs can be exploited by attackers to access cloud resources.
5	**CNA-05**	**Lack of Visibility and Monitoring**	Missing logs or alerts make it hard to detect and respond to attacks in cloud environments.
6	**CNA-06**	**Insecure Workload Configuration**	Weak configuration of containers, VMs, or serverless functions may expose vulnerabilities.

Rank	Risk Code	Title	Summary
7	**CNA-07**	**Lack of Segmentation and Isolation**	Shared resources without proper isolation allow lateral movement by attackers.
8	**CNA-08**	**Poor Secret Management**	Hardcoded or exposed secrets (API keys, credentials) lead to cloud service compromise.
9	**CNA-09**	**Supply Chain Vulnerabilities**	Vulnerable third-party packages or cloud dependencies can be entry points for attackers.
10	**CNA-10**	**Denial of Service (DoS)**	Improper resource limits can allow abuse, exhausting bandwidth or compute in the cloud.

⚠️ Cloud Computing Threats

Cloud environments introduce new types of **security risks and threats**. These threats can affect data confidentiality, integrity, and availability — especially in **IaaS, PaaS, and SaaS** models.

🔐 Top Cloud Computing Threats (Based on CSA, OWASP, and industry trends)

Threat No.	Name of Threat	Description
1️⃣	**Data Breaches**	Unauthorized access to sensitive data due to misconfigurations, poor access control, or shared environments.

Threat No.	Name of Threat	Description
2	**Account Hijacking**	Stolen credentials lead to attacker gaining control over cloud resources (e.g., via phishing or brute force).
3	**Insecure APIs**	Cloud APIs with weak authentication, input validation, or encryption are prone to abuse.
4	**Misconfigured Cloud Storage**	Public S3 buckets, open databases, or improperly configured firewalls lead to data exposure.
5	**Insider Threats**	Malicious or careless employees accessing or destroying data/resources from within the organization.
6	**Inadequate Identity and Access Management (IAM)**	Over-privileged users, lack of MFA, or poor role-based access.
7	**Denial of Service (DoS/DDoS)**	Cloud apps or services are overwhelmed by massive traffic, causing downtime.
8	**Poor Data Deletion Practices**	Sensitive data not fully deleted from cloud storage, leading to future exposure.
9	**Vendor Lock-In**	Difficulty migrating from one cloud to another due to proprietary platforms or configurations.
10	**Lack of Cloud Security Visibility**	No logging or monitoring, making it hard to detect or respond to security incidents.

Threat No.	Name of Threat	Description
11	**Supply Chain Attacks**	Vulnerabilities in third-party libraries, cloud services, or CI/CD tools that integrate with the cloud.
12	**Shared Technology Vulnerabilities**	Multi-tenant cloud environments can be affected if hypervisor or shared resource isolation fails.

How to Mitigate Cloud Threats

Category	Mitigation Techniques
Data Security	Encrypt data (at rest & in transit), use DLP, classify data
IAM	Apply least privilege, enable MFA, audit permissions
Network	Use VPC, subnets, security groups, and WAF
Monitoring	Enable logging (e.g., AWS CloudTrail, Azure Monitor), SIEM
Compliance	Follow standards like ISO 27001, GDPR, HIPAA
API Security	Use rate limiting, OAuth2, API Gateway, input validation

---

Cloud Bucket Enumeration Using LazyS3

LazyS3 is an **open-source cloud reconnaissance tool** used to **enumerate Amazon S3 (Simple Storage Service) buckets** related to a specific domain or keyword.
It helps security researchers and penetration testers identify **misconfigured or publicly accessible S3 buckets** that may contain sensitive data.

🧠 Simple Definition:

LazyS3 is a Python tool that automatically tries different bucket name combinations to find **open or exposed S3 buckets** on Amazon AWS related to a target domain.

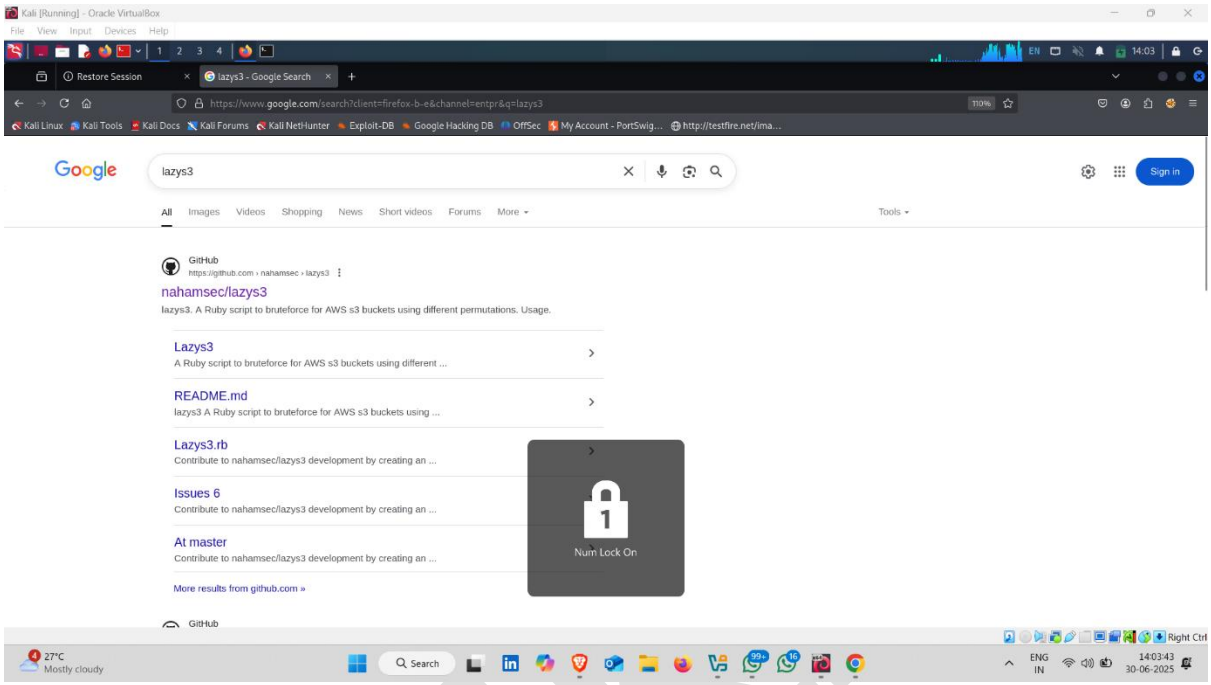
🔍 Key Features of LazyS3

Feature	Description
Automated bucket discovery	Uses wordlists and permutations to guess S3 bucket names
Domain-based search	Accepts target domains to base bucket name guesses
Checks bucket accessibility	Reports if buckets are **public**, **forbidden**, or **non-existent**
Lightweight and simple	Easy to use, CLI-based Python tool
Great for passive recon	Non-intrusive way to map cloud assets

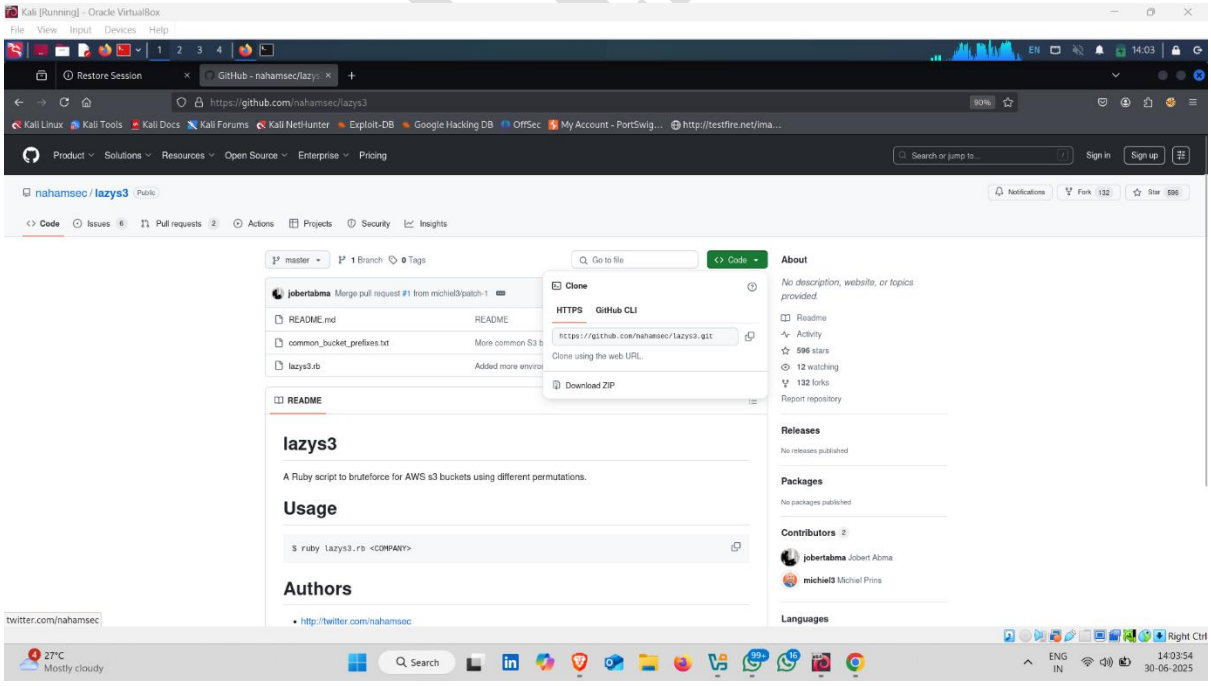
---

How to Download it -:

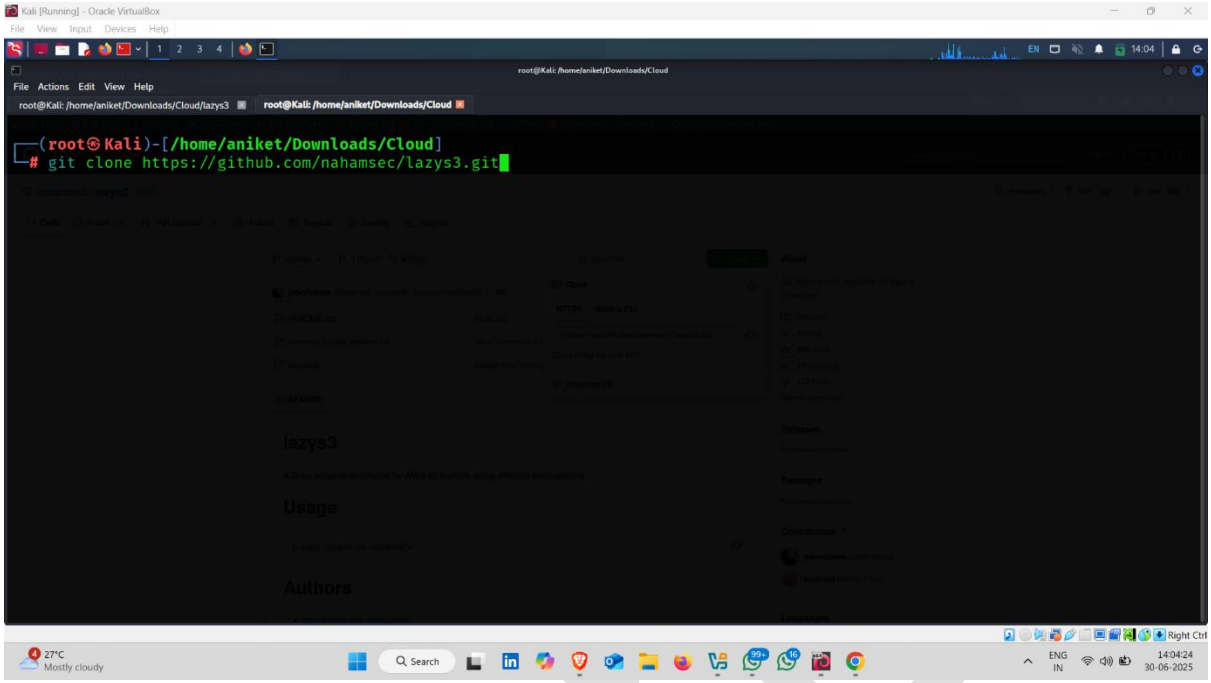
- Open kali linux browser and search lazys3 git hub
- Click on first link



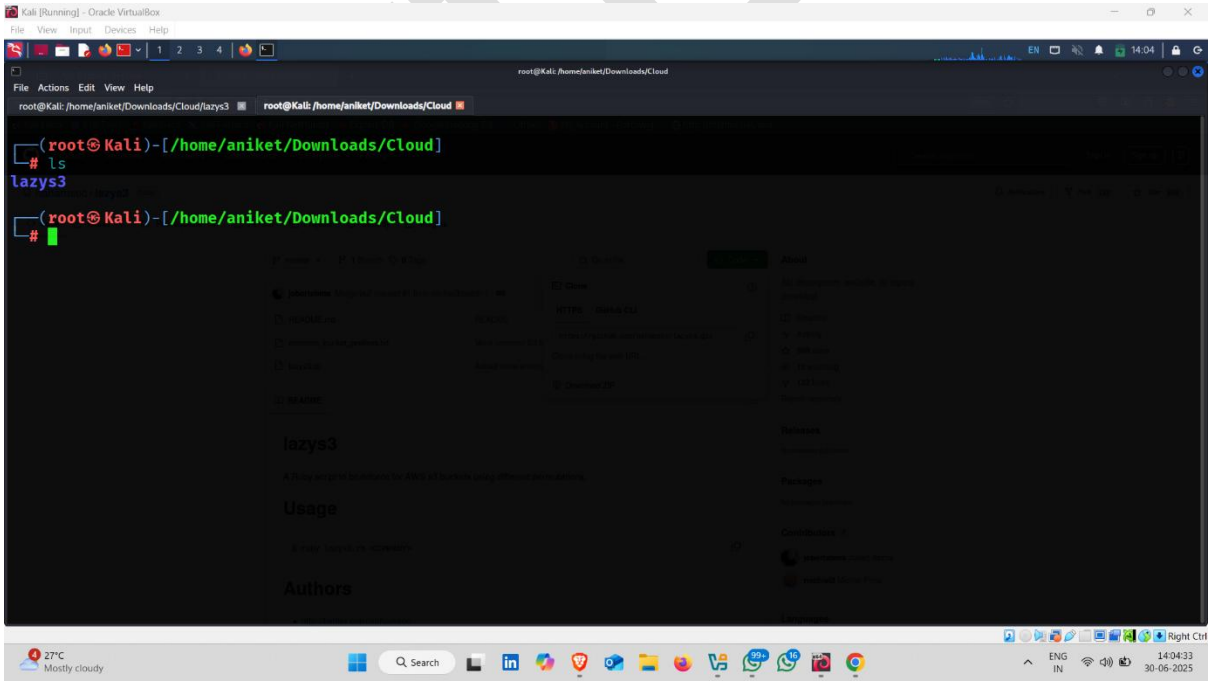
- Click on **green code** button and **copy HTTPS URL**



- Open terminal and paste HTTPS URL with git clone 👇

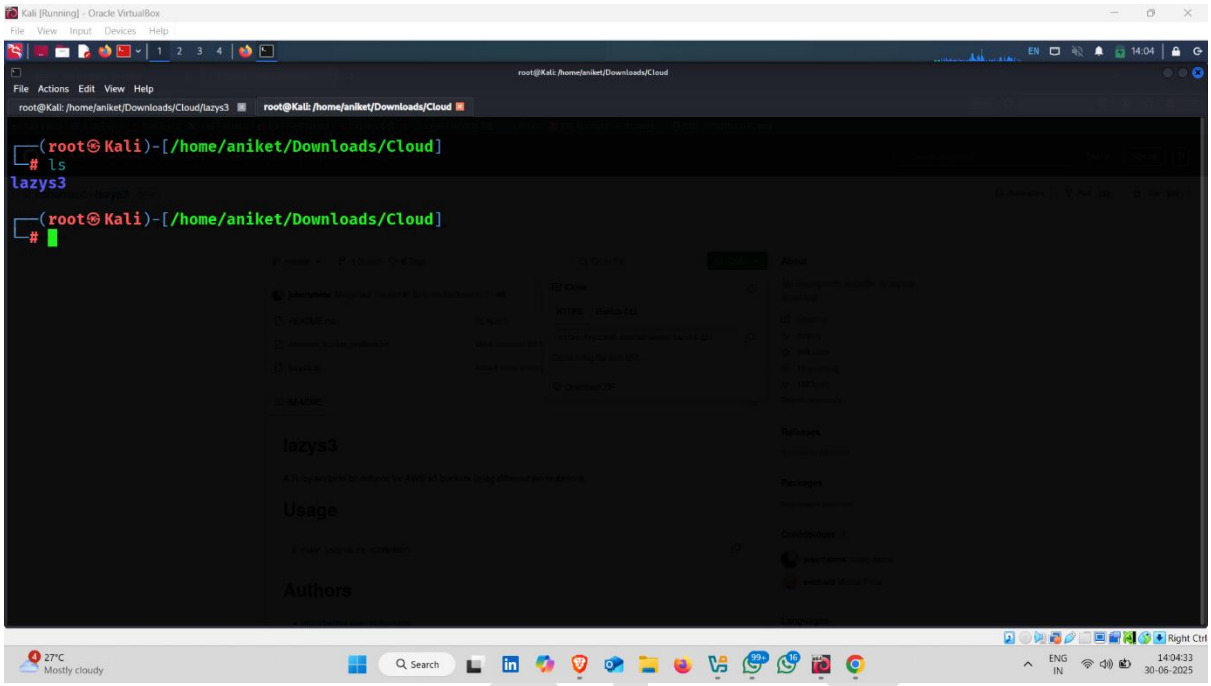


- Lazys3 downloaded 👇✅

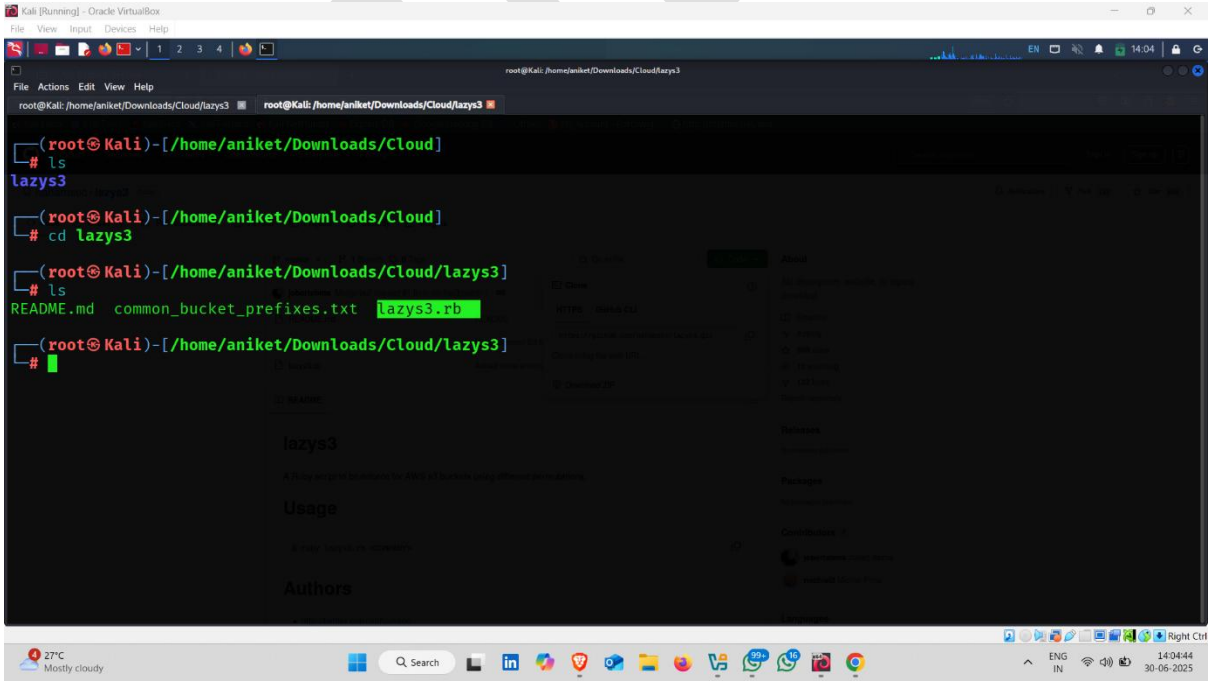


How to use it -:

- Go to the lazys3 directory



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

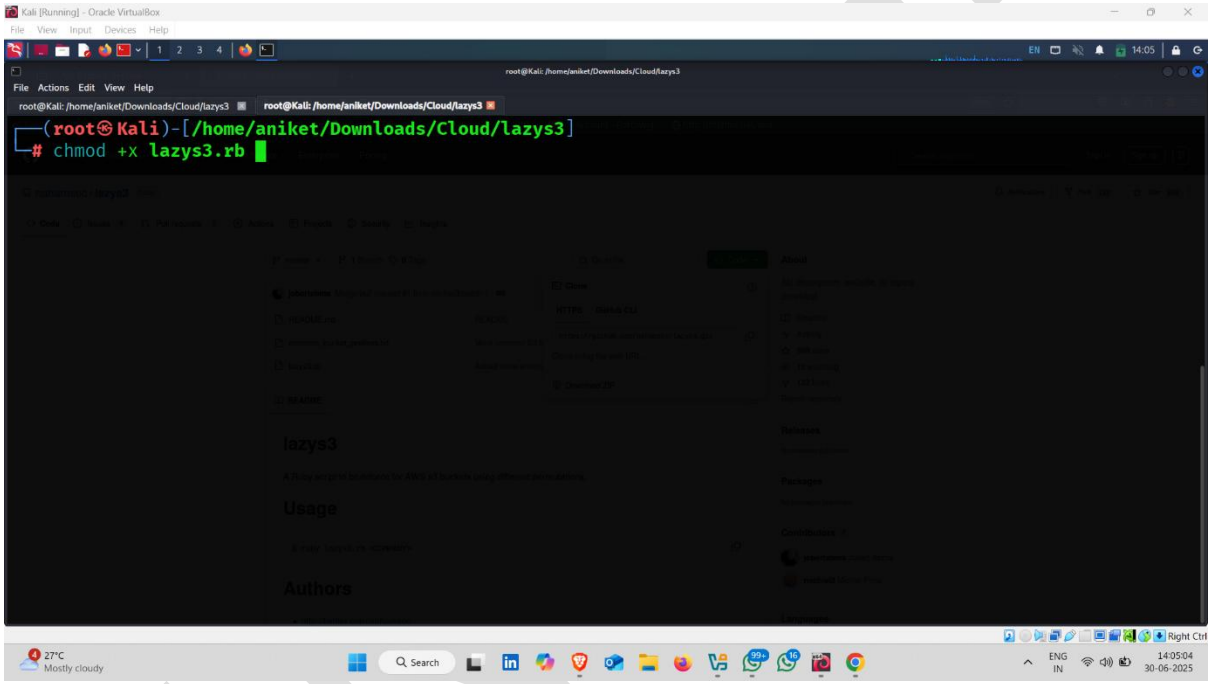


- Type following command

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).



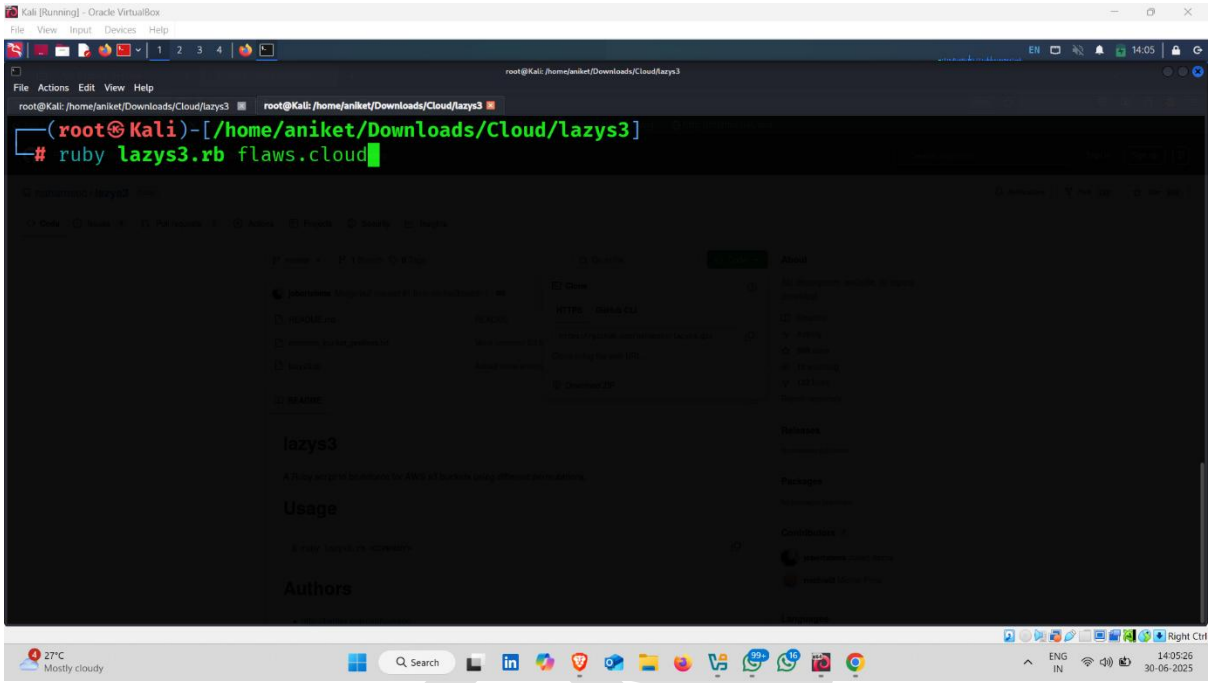
- type next command

command -: ruby lazys3.rb flaws.cloud

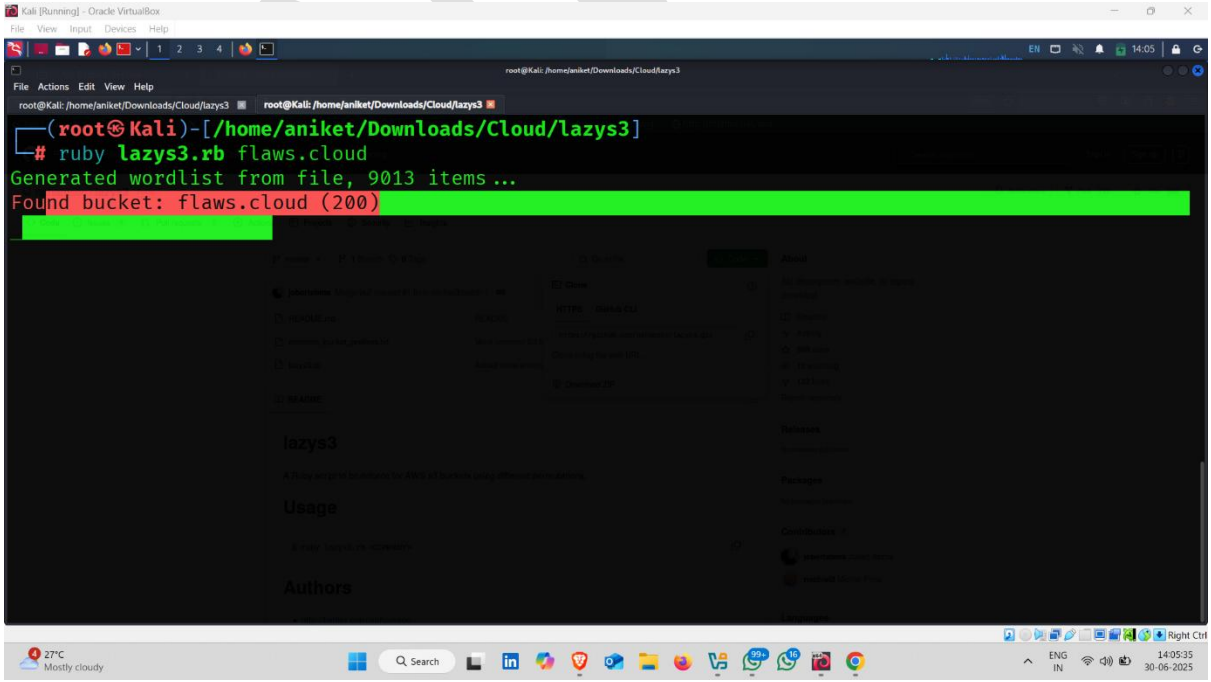
explanation -:

- 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**

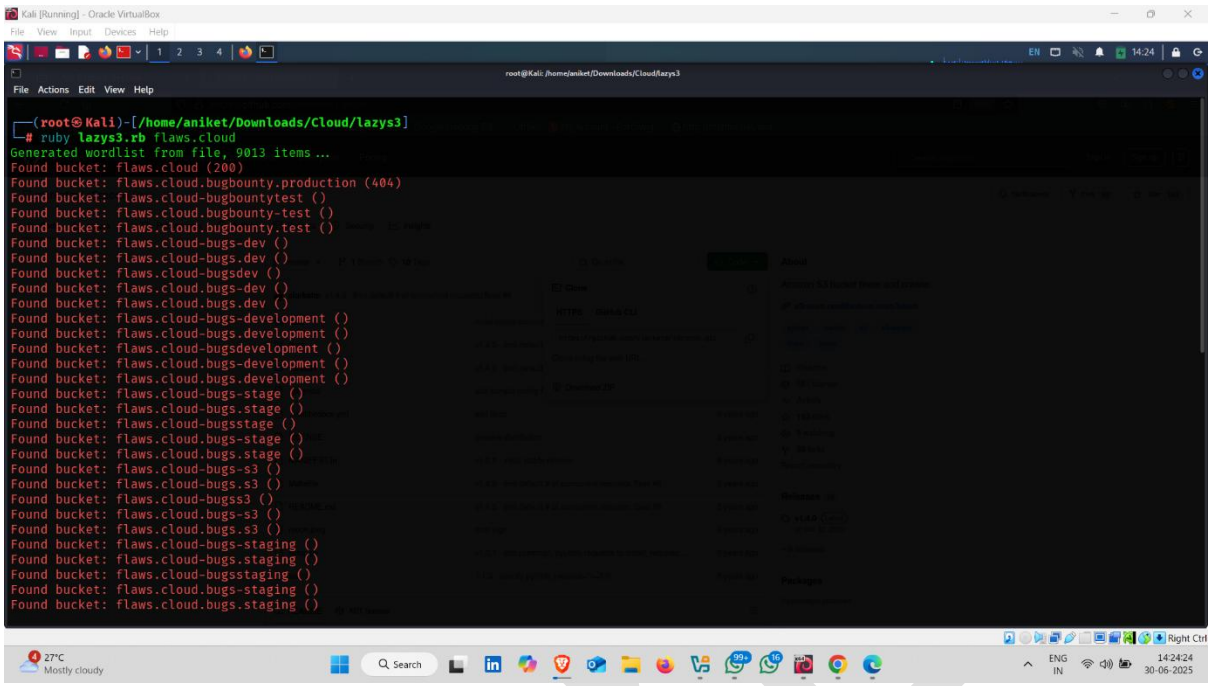
- Are **publicly accessible**
- Are **restricted** or **forbidden**



- Bucket found ✅ 👇



- **Result 👇 ✅**



Cloud Reconnaissance and Security Testing with S3Scanner

🔍 What is S3Scanner?

S3Scanner is an **open-source AWS S3 bucket enumeration tool** that helps **security professionals, penetration testers, and bug bounty hunters** identify publicly accessible or misconfigured **Amazon S3 buckets**.

🧠 Simple Definition:

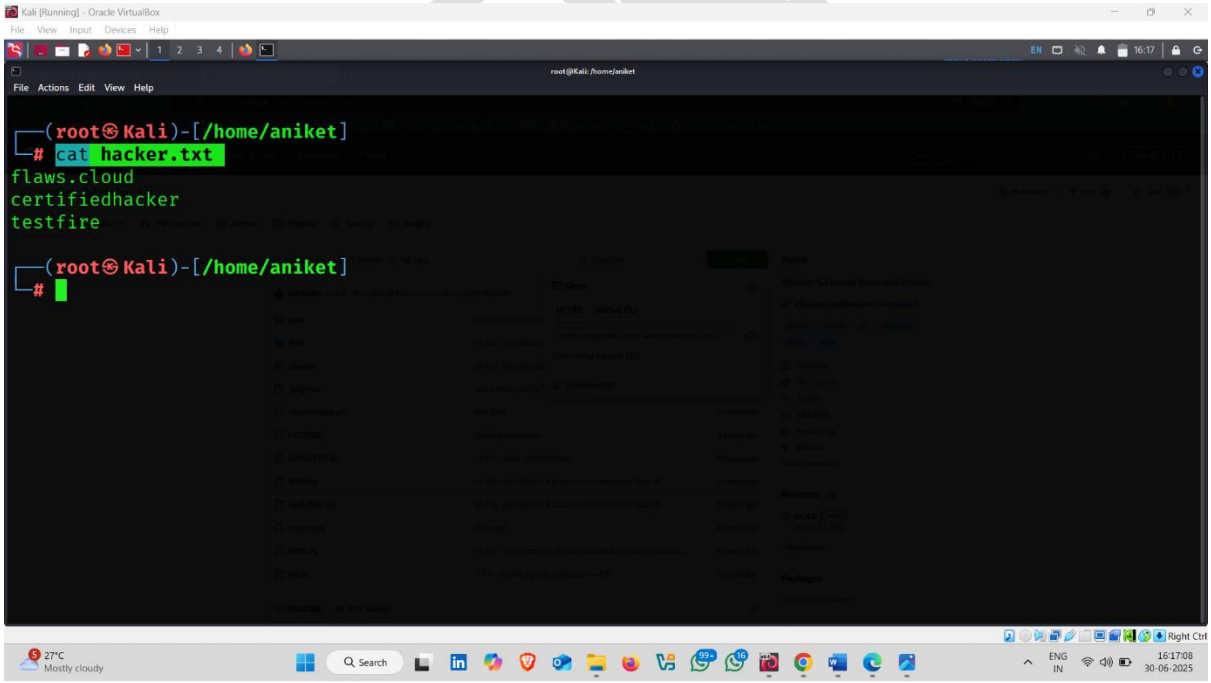
S3Scanner scans a list of potential S3 bucket names and tells you which ones exist and whether they are publicly accessible.

🛠️ Key Features

Feature	Description
Fast scanning	Checks thousands of bucket names quickly
Public access detection	Identifies buckets that are readable/listable
Optional write testing	Can check if you can write/upload to a bucket (if permitted)
Custom wordlist support	Accepts your own list of target bucket names
HTTP status codes	Shows 200 (OK), 403 (Forbidden), 404 (Not Found)

How to use it -:

- I have already one txt file that contain 3 websites



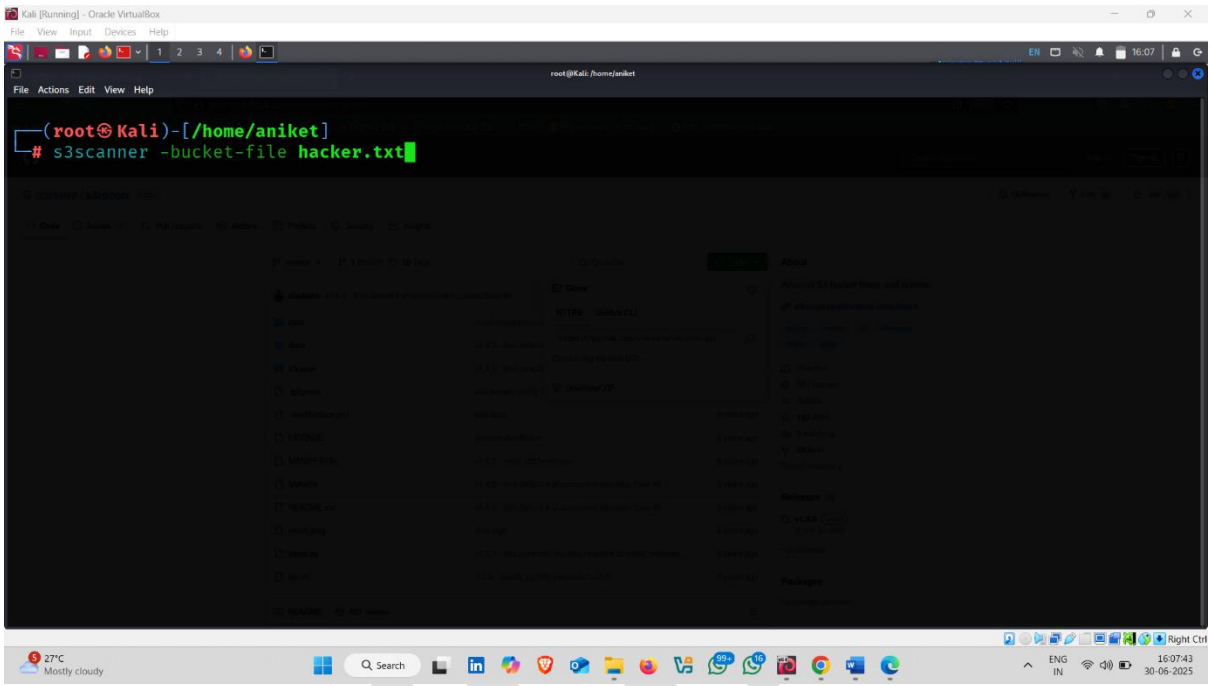
- Type following command

Command -: s3scanner -bucket-file hacker.txt

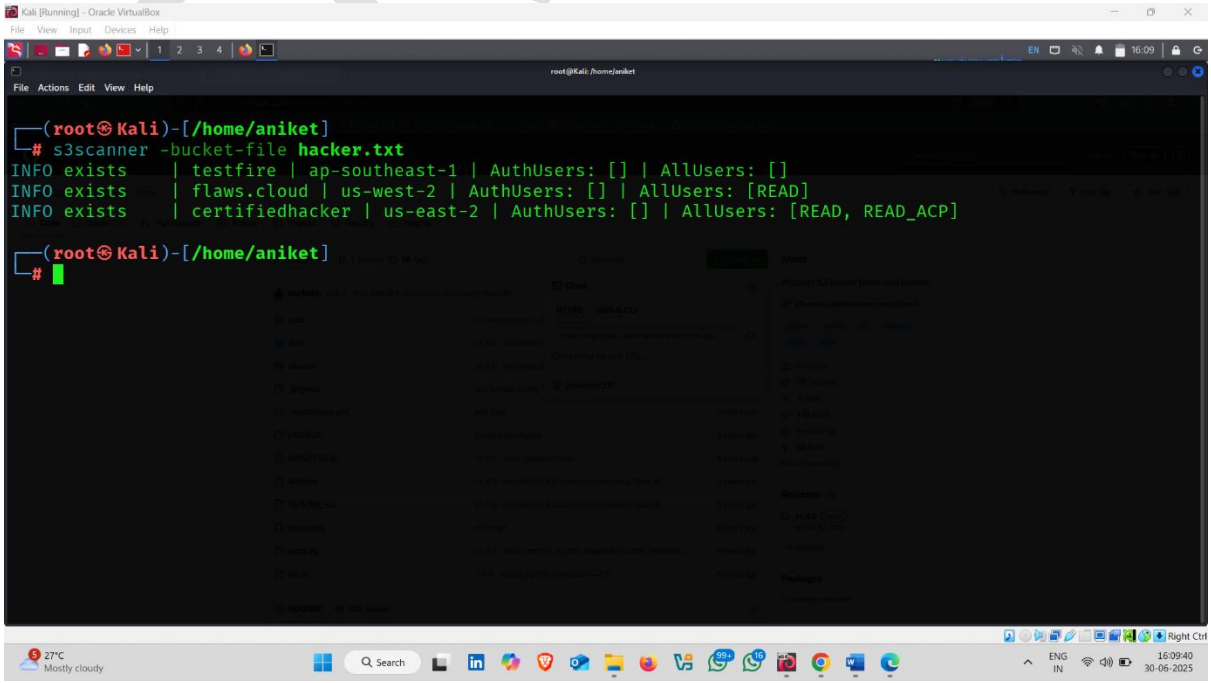
Explanation -:

This command tells **S3Scanner** to:

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



- **S3Scanner found** three existing S3 buckets: **flaws.cloud** and **certifiedhacker** are publicly readable, while **testfire** exists but is private. **certifiedhacker** also exposes its access control policy (READ_ACP).

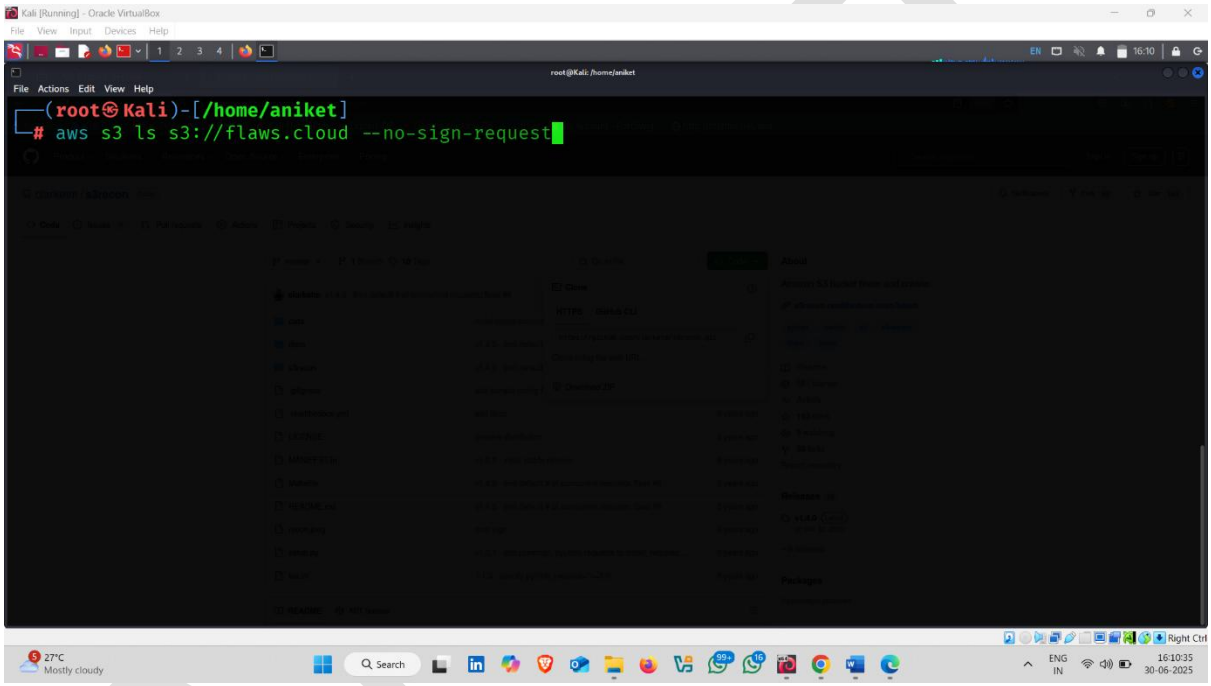


- Type next command

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

Explanation -:

This command lists the files inside the **public S3 bucket** named flaws.cloud
--no-sign-request means **no AWS login is needed** (used for open buckets)
It shows you the files and folders if the bucket is publicly accessible.
✅

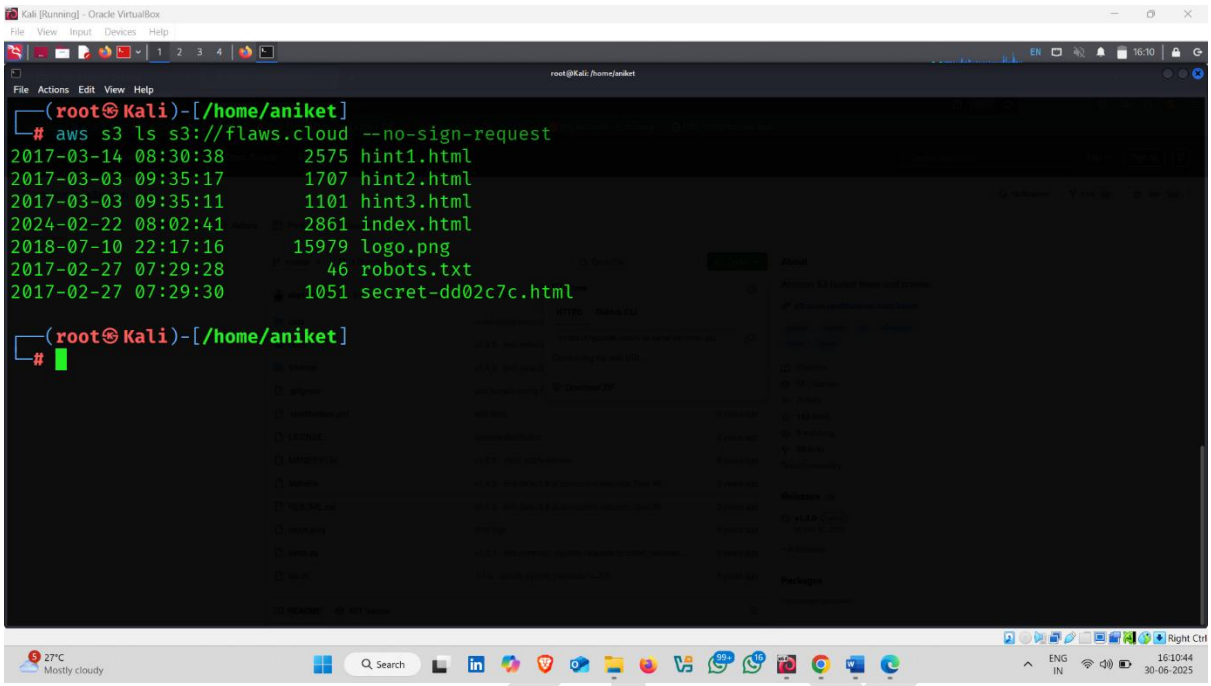


- Result ✅👇

You found 7 public files in the S3 bucket flaws.cloud.
Here's what they are:

- 📝 **hint1.html, hint2.html, hint3.html: Clue files**
- 🏠 **index.html: Main page**
- 🖼️ **logo.png: Image file**
- ⚙️ **robots.txt: Tells search engines what to ignore**
- 🔐 **secret-dd02c7c.html: May contain a hidden flag or secret**

👉 **You can open any file in your browser like this:**
http://flaws.cloud/secret-dd02c7c.html



Vulnerability Scanning of Container Images Using Trivy

Trivy (pronounced *"triv-ee"*) is a **free and open-source vulnerability scanner** used to find **security issues** in:

- 🐳 **Docker containers**
- 📦 **Operating systems & packages**
- 📝 **Infrastructure as Code (IaC)** (e.g., Terraform, CloudFormation)
- ☁️ **Cloud services (AWS)**
- 📄 **SBOMs (Software Bill of Materials)**

🧠 Simple Definition:

Trivy is a tool that quickly scans your container images, code, or cloud for **vulnerabilities and misconfigurations** before you deploy them.

✅ What Trivy Can Scan

🔍 Target	📌 Description
Container Images	Scans for known CVEs (vulnerabilities) in Docker images
File Systems	Checks local files or root file systems for issues
Git Repos	Analyzes project dependencies from your source code
Kubernetes Clusters	Scans for misconfigured or insecure settings
IaC Files	Detects issues in Terraform, Dockerfiles, etc.
AWS Accounts	Finds insecure cloud configurations
SBOM	Validates third-party packages via SPDX/CycloneDX

How to use it -:

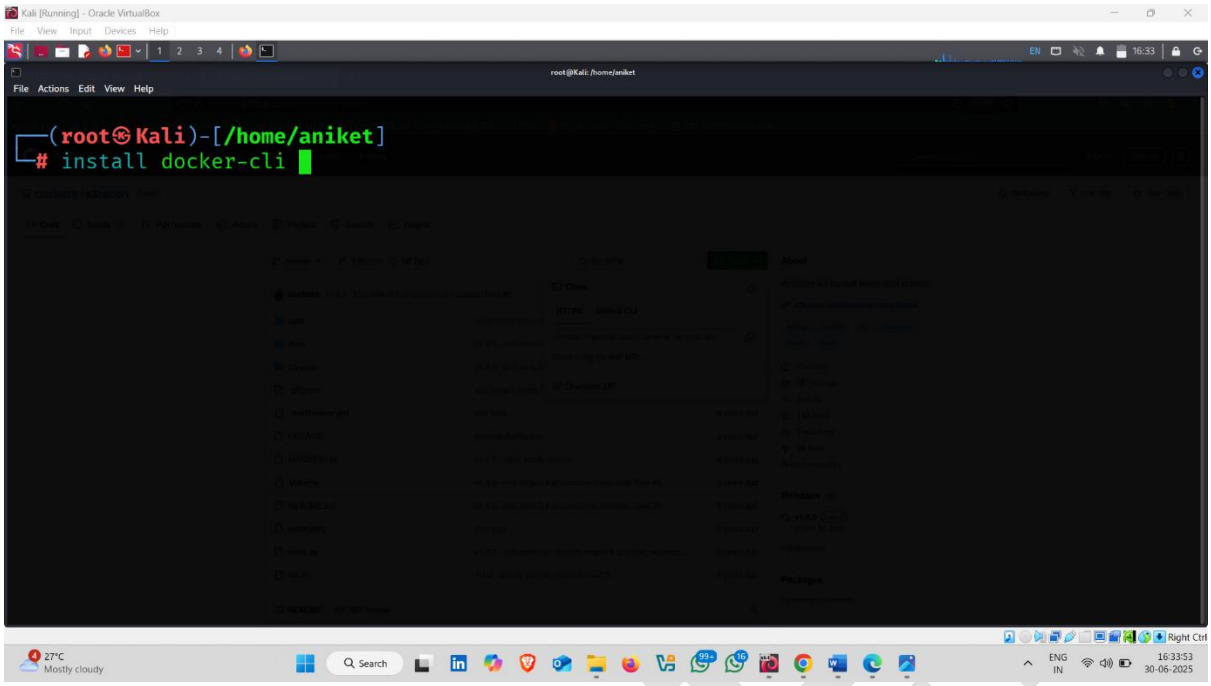
- Type following command

Command -: install docker-cli

Explanation -:

This command installs only the Docker Command Line Interface (CLI) — which means:

You can run docker commands (like docker pull, docker run, etc.) on the system.

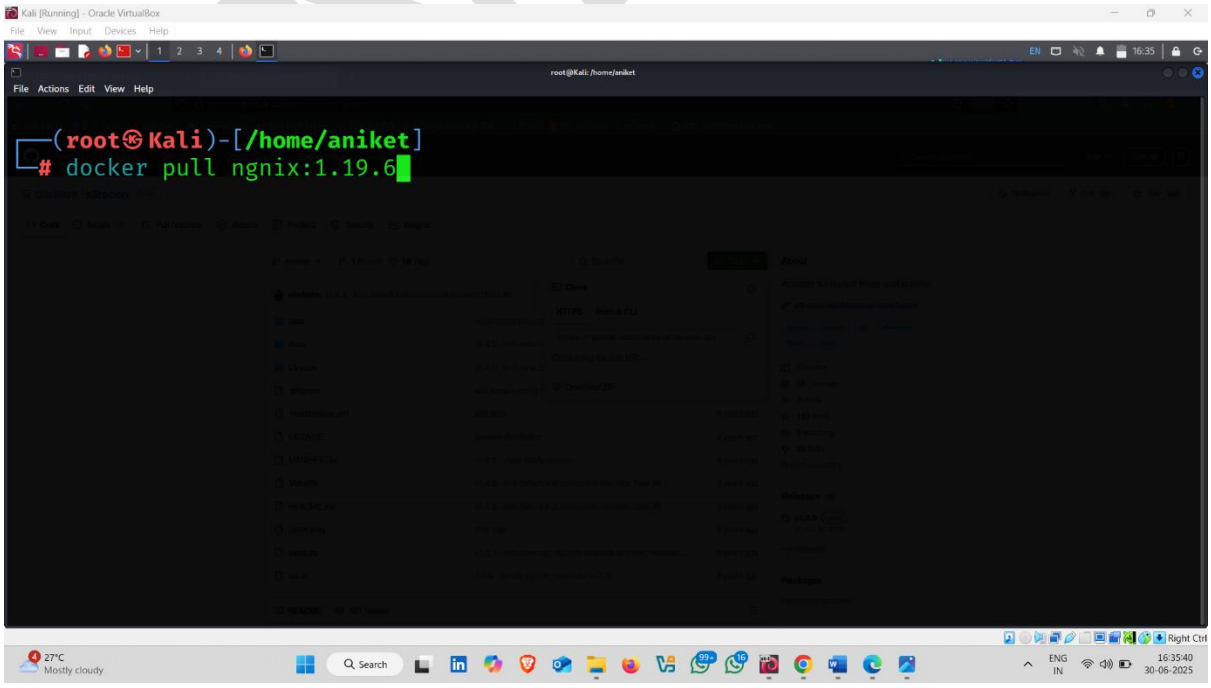


- Type next command

Command -: docker pull ngnix:1.19.6

Explanation -:

This command tells Docker to **download the nginx web server image** version **1.19.6** from **Docker Hub** (official container registry).

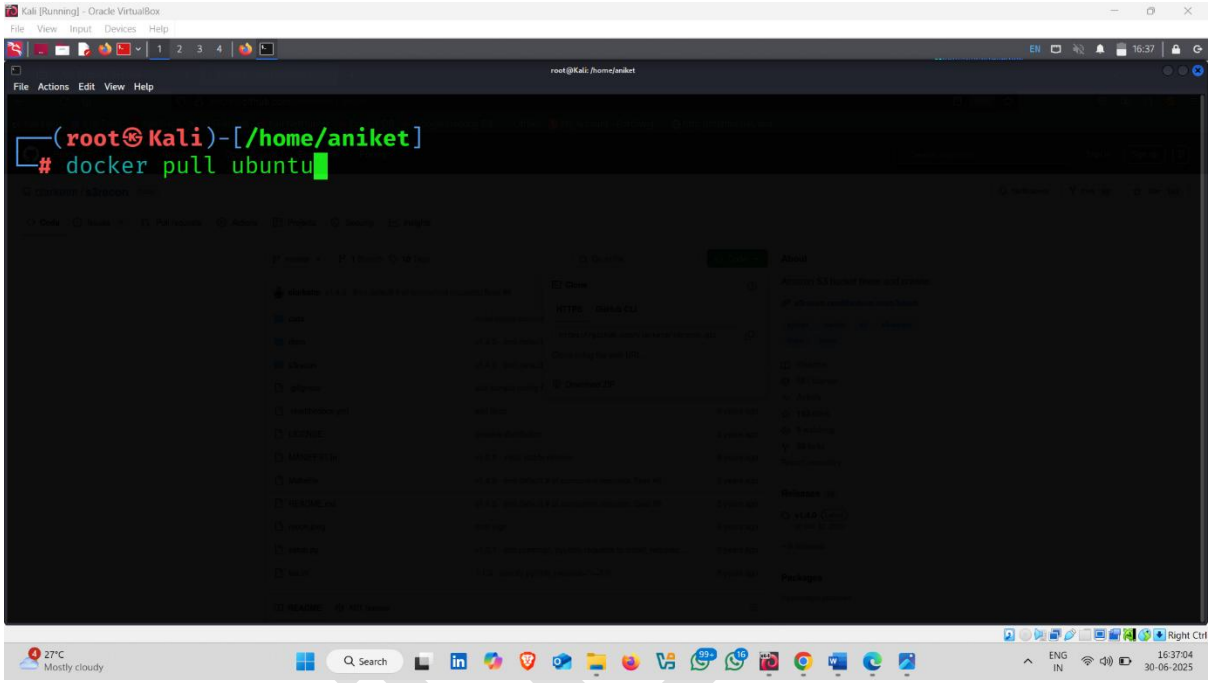


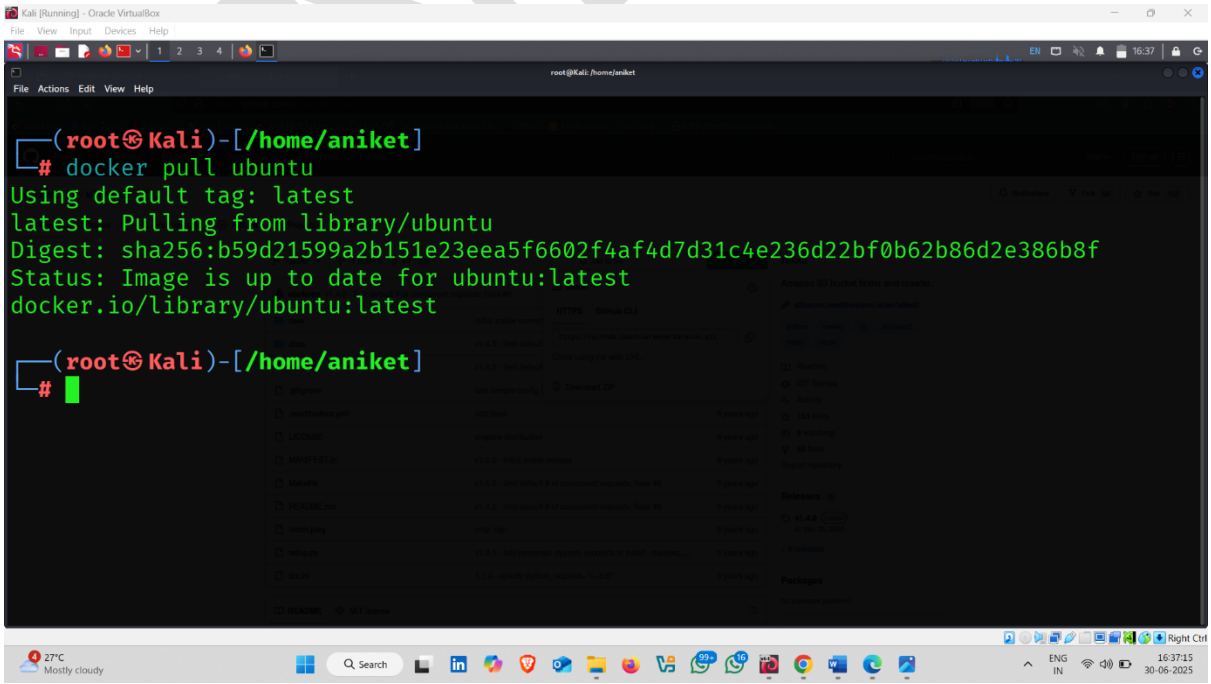
- Type next command

Command -: docker pull ubuntu

Explanation -:

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



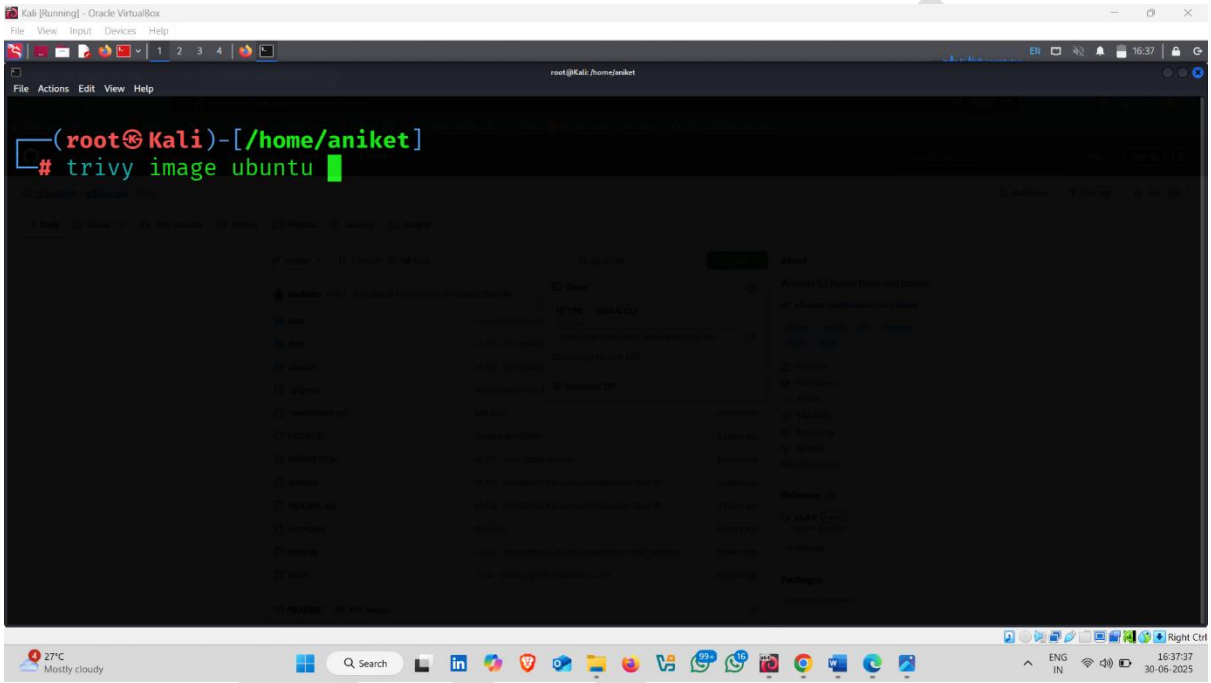


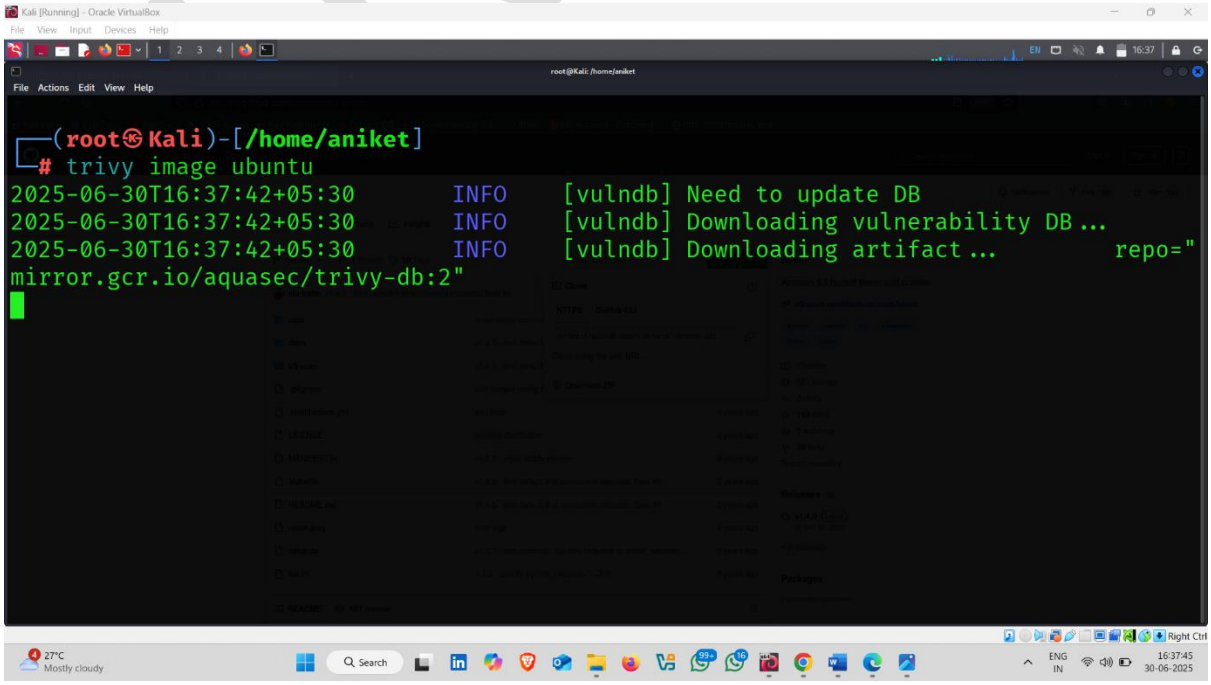
- Next command

Command -: trivy image ubuntu

Explanation -: This command uses **Trivy** to **scan the Ubuntu Docker image** for **vulnerabilities** such as:

- Known CVEs (Common Vulnerabilities and Exposures)
- Outdated or insecure software packages

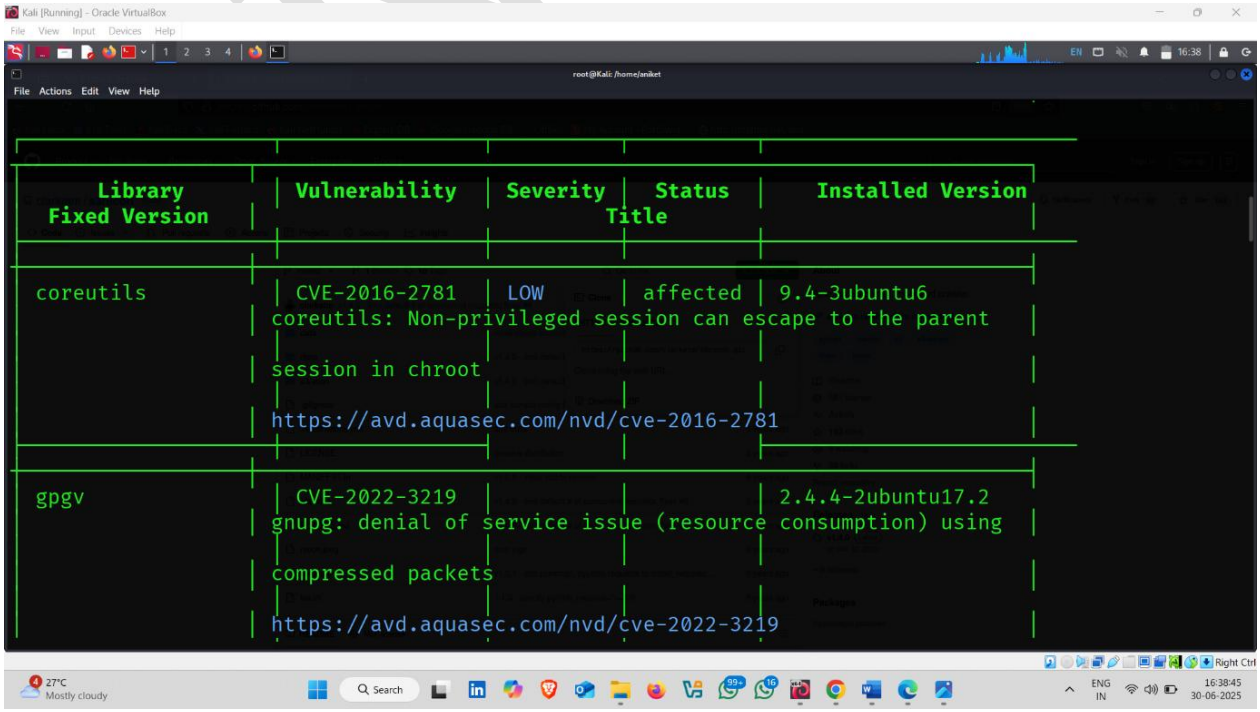


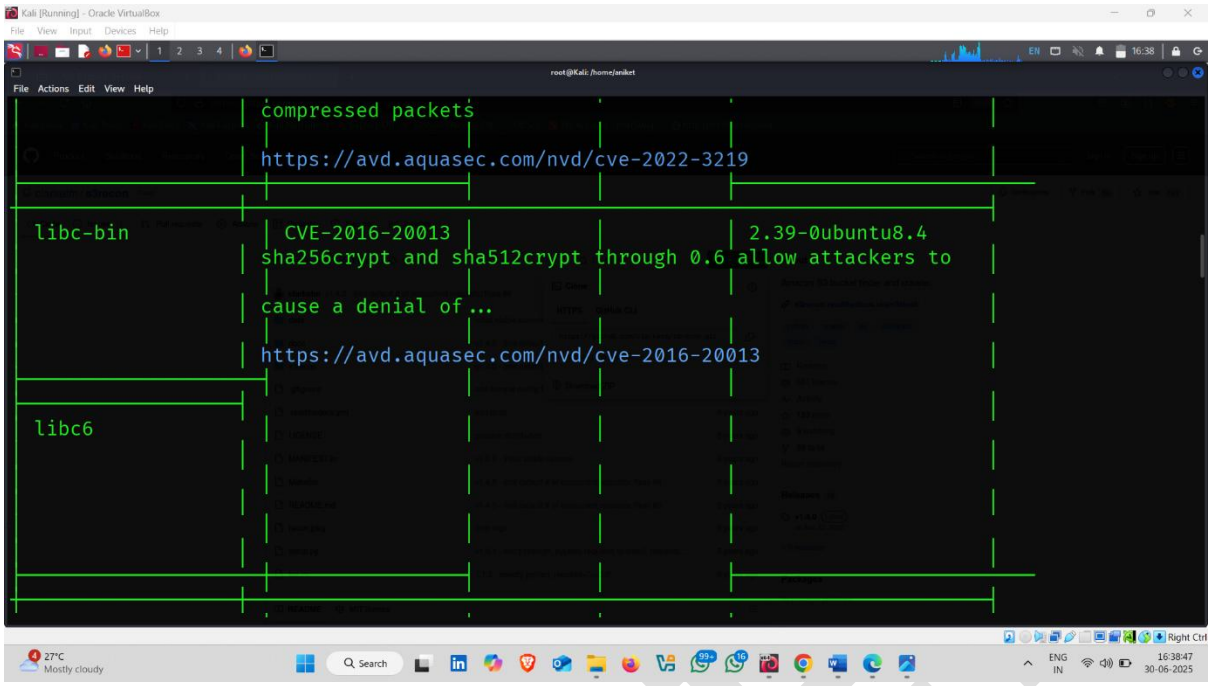


- **Result – vulnerabilities found** 👇 ✅

📄 **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**.




Kali [Running] - Oracle VirtualBox
File View Input Devices Help
root@Kali: /home/aniket
File Actions Edit View Help
compressed packets
https://avd.aquasec.com/nvd/cve-2022-3219
libc-bin
CVE-2016-20013
2.39-0ubuntu8.4
sha256crypt and sha512crypt through 0.6 allow attackers to
cause a denial of ...
https://avd.aquasec.com/nvd/cve-2016-20013
libc6
27°C
Mostly cloudy
Search
ENG
IN
16:38:47
30-06-2025
Right Ctrl