CLOUD LAB

What is Docker?

Docker is a platform that allows you to develop, ship, and run applications in isolated environments called **containers**. Containers package applications and their dependencies together, ensuring that they run consistently across different environments. It eliminates the "works on my machine" problem.

How Does Docker Work?

Docker uses **containerization** to isolate applications. It works by:

1. Using **Docker Engine**, which runs on the host system to manage containers.
2. Utilizing **Docker Images**, which are pre-packaged applications with all dependencies.
3. Running **Docker Containers**, which are instances of Docker images.
4. Using **Docker Hub**, a repository of pre-built images that you can pull and run.

Steps to Use Docker on Kali Linux

1. Install Docker on Kali Linux

Run the following commands to install Docker:

```
sudo apt update
sudo apt install docker.io -y
```

Enable and start the Docker service:

```
sudo systemctl enable --now docker
```

Verify installation:

```
docker --version
```

2. Verify Docker Installation

Check if Docker is working properly:

```
docker run hello-world
```

This downloads a small test image and runs it in a container.

Running a Container in Docker

1. Pull an Image from Docker Hub

You can pull images from Docker Hub. For example, to pull an **Ubuntu** image:

```
docker pull ubuntu
```

2. Run a Container

Start a container with an interactive shell:

```
docker run -it ubuntu bash
```

This starts a new Ubuntu container and opens a bash shell inside it.

3. List Running Containers

To see running containers:

```
docker ps
```

To see all containers (including stopped ones):

```
docker ps -a
```

4. Stop and Remove Containers

To stop a running container:

```
docker stop <container_id>
```

To remove a container:

```
docker rm <container_id>
```

5. Remove an Image

To remove a downloaded image:

```
docker rmi <image_name>
```

Example: Running Nginx on Kali Linux

If you want to run a web server using **Nginx**, follow these steps:

1. Pull the Nginx image:

   ```
   docker pull nginx
   ```

2. Run the Nginx container:

   ```
   docker run -d -p 8080:80 nginx
   ```

3. Open a browser and visit:

   ```
   http://localhost:8080
   ```

   You should see the default Nginx welcome page.

4. Stop the Nginx container:

   ```
   docker stop <container_id>
   ```

Running DVWA (Damn Vulnerable Web Application) Using Docker on Kali Linux

DVWA is a deliberately vulnerable web application for security testing. You can quickly set it up using Docker.

Step 1: Pull the DVWA Docker Image

Run the following command to download the DVWA image:

```
docker pull vulnerables/web-dvwa
```

Step 2: Run DVWA Container

Now, start a new DVWA container using the command below:

```
docker run -d -p 8080:80 vulnerables/web-dvwa
```

- `d`: Runs the container in detached mode (in the background).
- `p 8080:80`: Maps port 8080 on your Kali Linux machine to port 80 inside the container.

Step 3: Access DVWA

Open your web browser and go to:

```
http://localhost:8080
```

If running on a remote server, use:

```
http://<your-ip>:8080
```

Step 4: Login to DVWA

Use the default login credentials:

- **Username**: `admin`

- **Password**: `password`

Click **Login**, and you should now be inside DVWA.

Step 5: Configure DVWA

Once logged in:

1. Go to **DVWA Security** settings.
2. Set **Security Level** to **Low** to enable testing.
3. Click **Create/Reset Database** under **Database Setup**.

Step 6: Managing the DVWA Container

Check Running Containers

```
docker ps
```

Stop the Container

```
docker stop <container_id>
```

Restart the Container

```
docker start <container_id>
```

Remove the Container

```
docker rm <container_id>
```

Remove the Image (if needed)

```
docker rmi vulnerables/web-dvwa
```

DC4 using docker

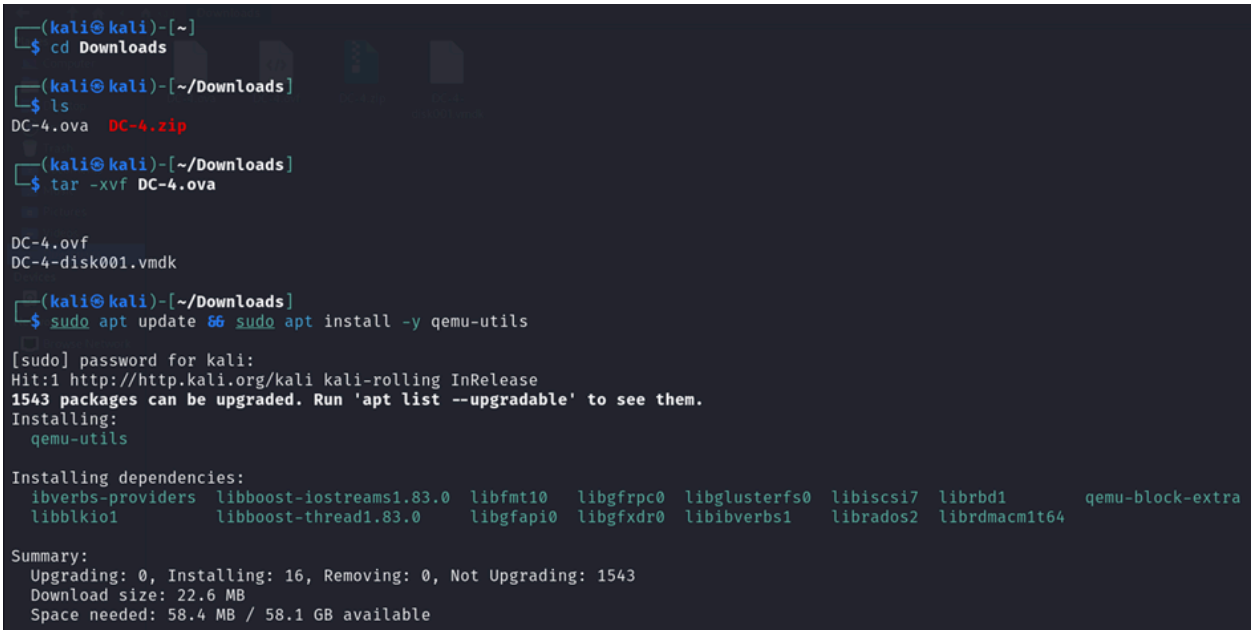
Step 1: Extract the OVA File

First, extract the **DC-4.ova** file to get the virtual disk (**.vmdk**) inside it.

tar -xvf DC-4.ova

This will extract files like:

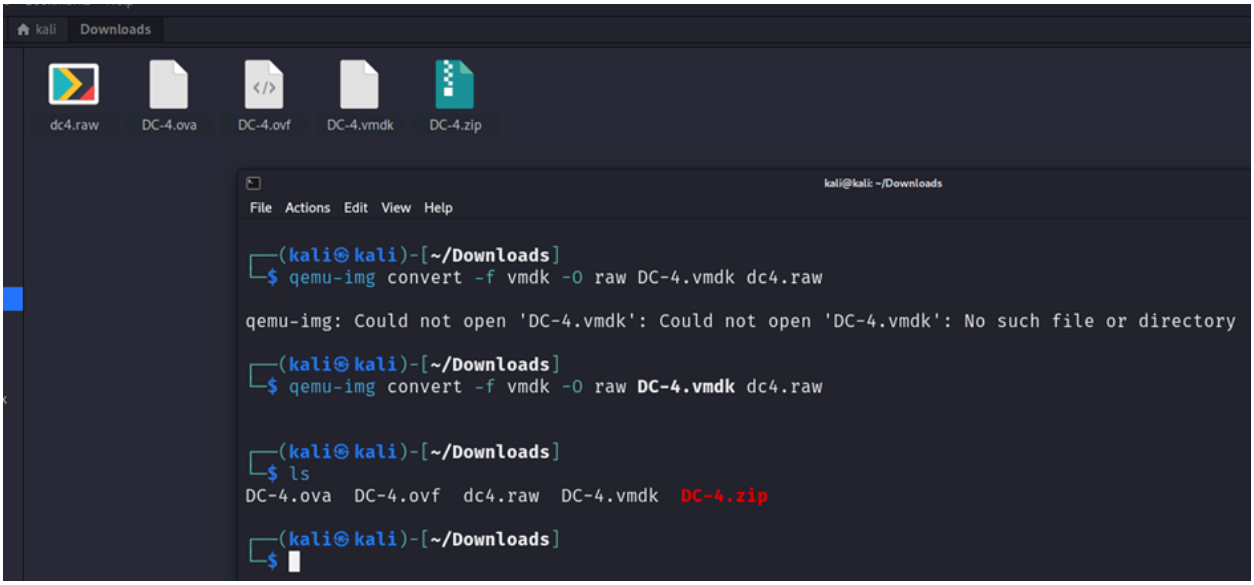
- **DC-4.ovf** (Open Virtualization Format descriptor)
- **DC-4.vmdk** (Virtual disk file)



Step 2: Convert VMDK to a Raw Image

Use qemu-img to convert the VMDK file to a raw image:

qemu-img convert -f vmdk -O raw DC-4.vmdk dc4.raw



Step 3: Create a Docker Image from the Raw Disk

Now, create a Docker image using the raw disk file.

1. Create a directory for the image: **mkdir dc4-docker && cd dc4-docker**
2. Move the raw image into the directory: **mv ../dc4.raw .**
3. Create a Dockerfile in the dc4-docker directory: **nano Dockerfile**

##on terminal

sudo apt update && sudo apt install -y qemu-system-x86 qemu-utils

RUN apt update && apt install -y qemu-system-x86 wget

sudo apt update && sudo apt install -y docker.io qemu-system-x86\n

sudo systemctl start docker

sudo systemctl status docker

sudo systemctl enable --now docker

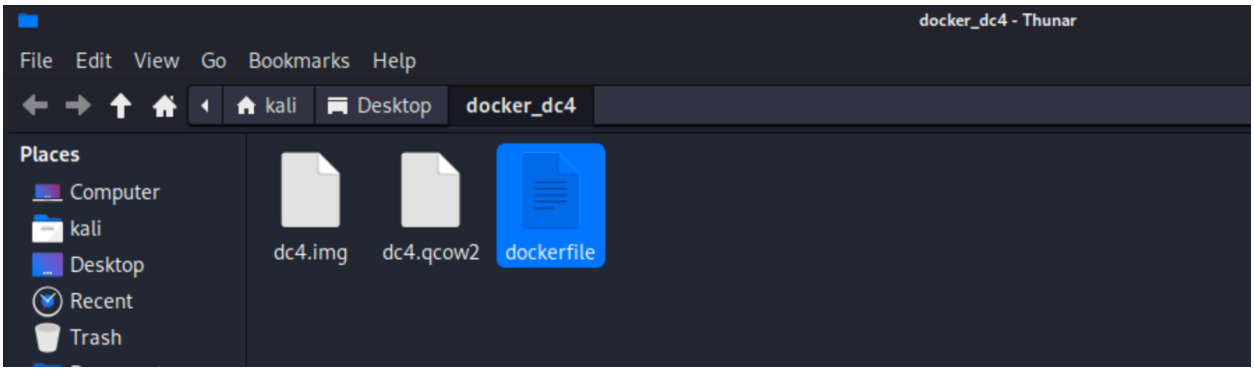
mkdir docker_dc4

#keep a copy of dc4.img and dc4.q2cow

#steps to extract .img

qemu-img convert -O raw DC4-disk001.vmdk dc4.img

sudo qemu-img convert -f raw -O qcow2 dc4.img dc4.qcow2



touch dockerfile

##paste this inside dockerfile

FROM ubuntu:latest

RUN apt update && apt install -y qemu-system-x86

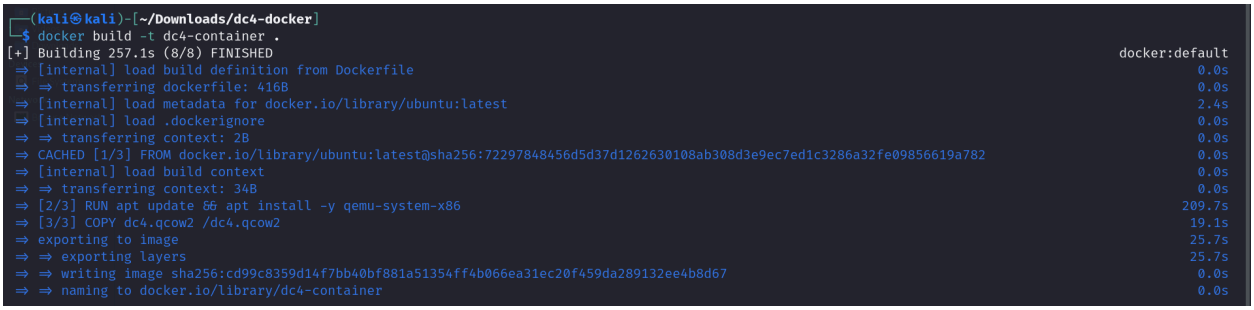
COPY dc4.qcow2 /dc4.qcow2

EXPOSE 80

CMD ["qemu-system-x86_64", "-m", "2048", "-hda", "/dc4.qcow2", "-net", "nic", "-net", "user,hostfwd=tcp::80-:80", "-nographic"]

4. Build the Docker image:

docker build -t dc4-container .



Step 4: Run the DC-4 Container

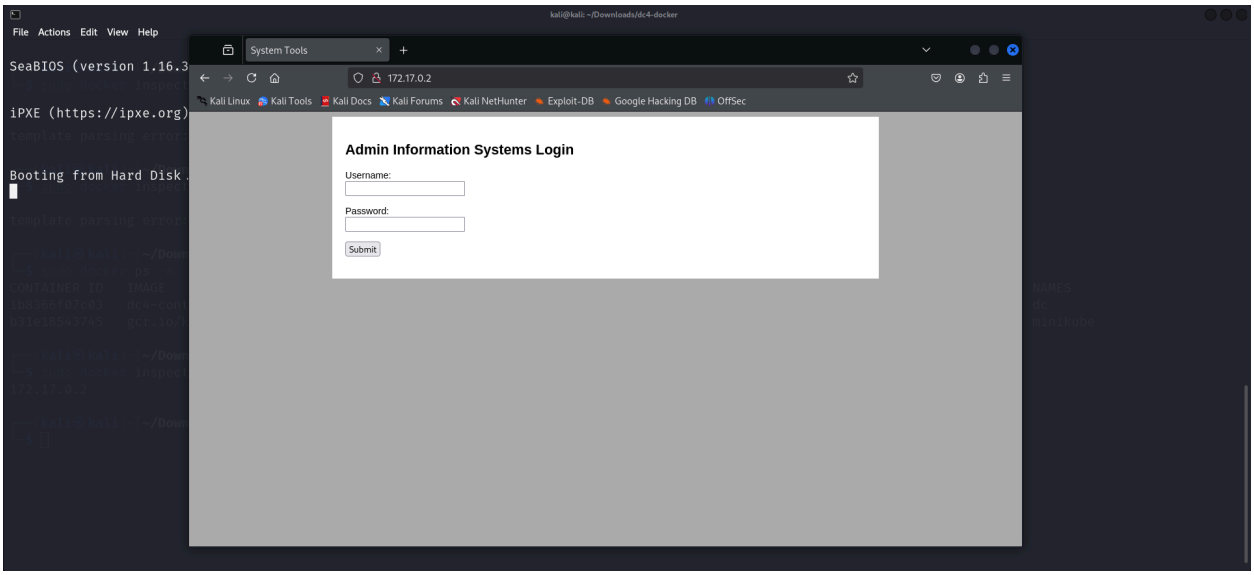
sudo docker run --name dc -it dc4-container

##it will start to boot

NEW TERMINAL;

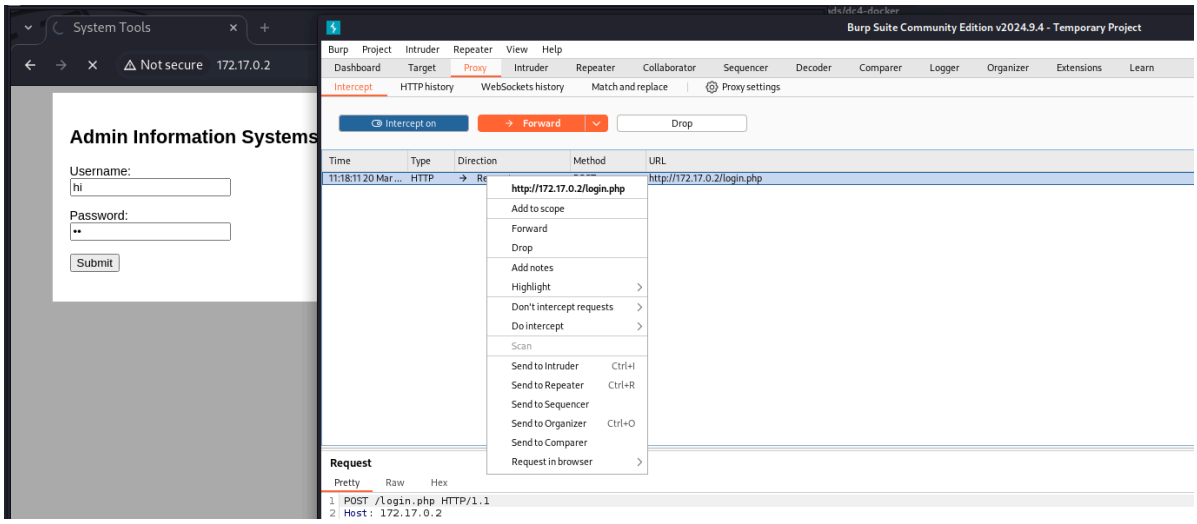
sudo docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}} {{end}}' dc4-container

copy ip address and paste it in mozilla

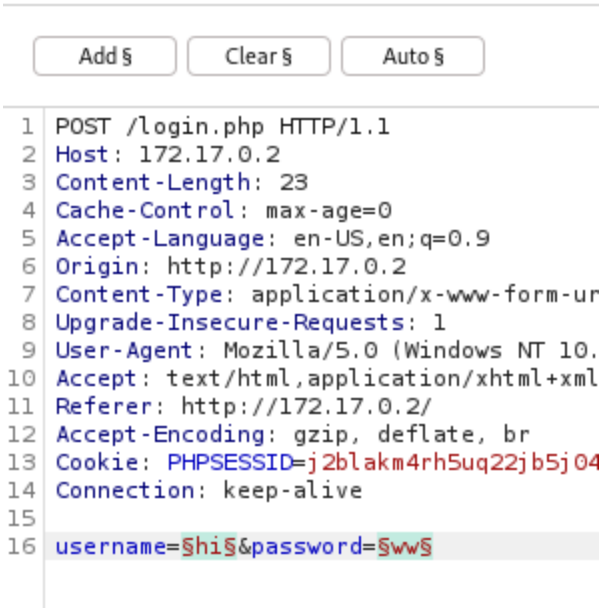


Start burpsuit bruteforce attack

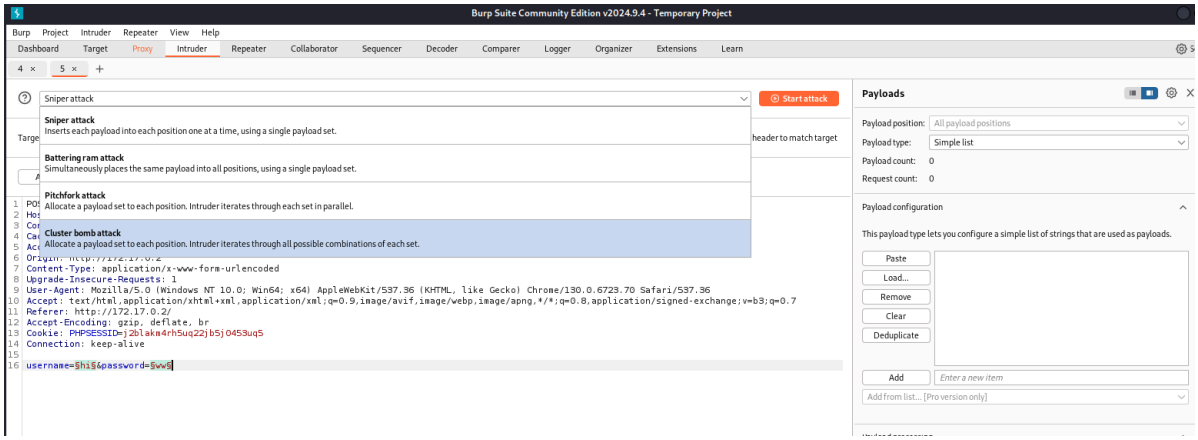
1. Send the login request to intruder

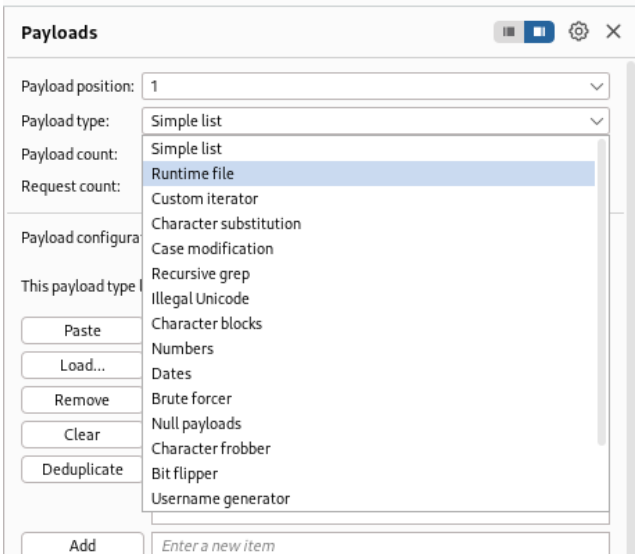


2. Add the section to add payload

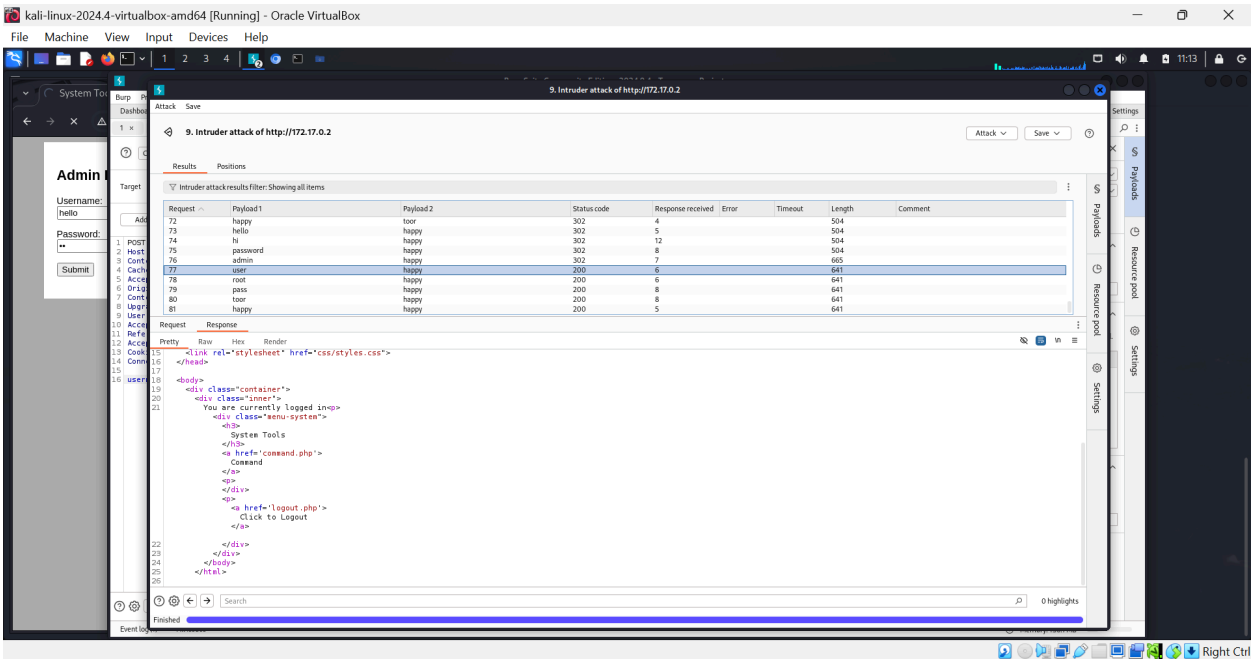


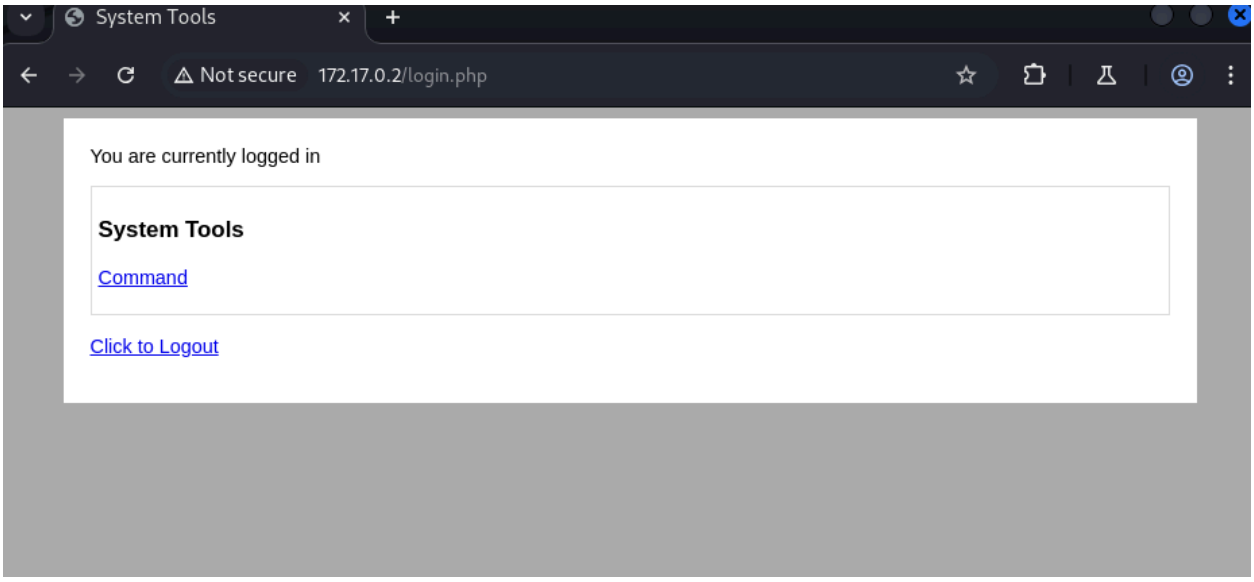
3. Use the required type of brute force attack and payload type





4. Add wordlist
5. Start attack check the payload for where status code is 200 you have the password Congrats😎



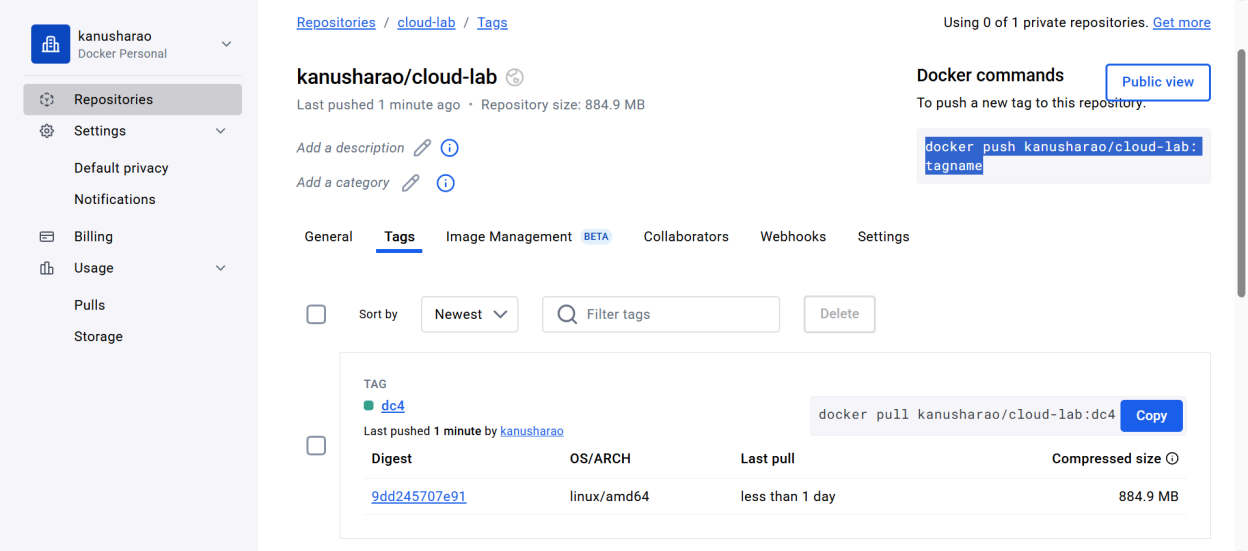


How to push image to docker Hub?

1. Create an account in docker hub
2. Create a repository
3. Login to the docker hub account from local using **docker login**
4. If your image is not already tagged correctly, tag it using: **docker tag <local-image-id> kanusharao/cloud-lab:tagname**
5. Find your **local image ID** using: **docker images**
6. Run the following command to push the image: **docker push kanusharao/cloud-lab:tagname**
7. Repo Link: https://hub.docker.com/repository/docker/kanusharao/cloud-lab/tags

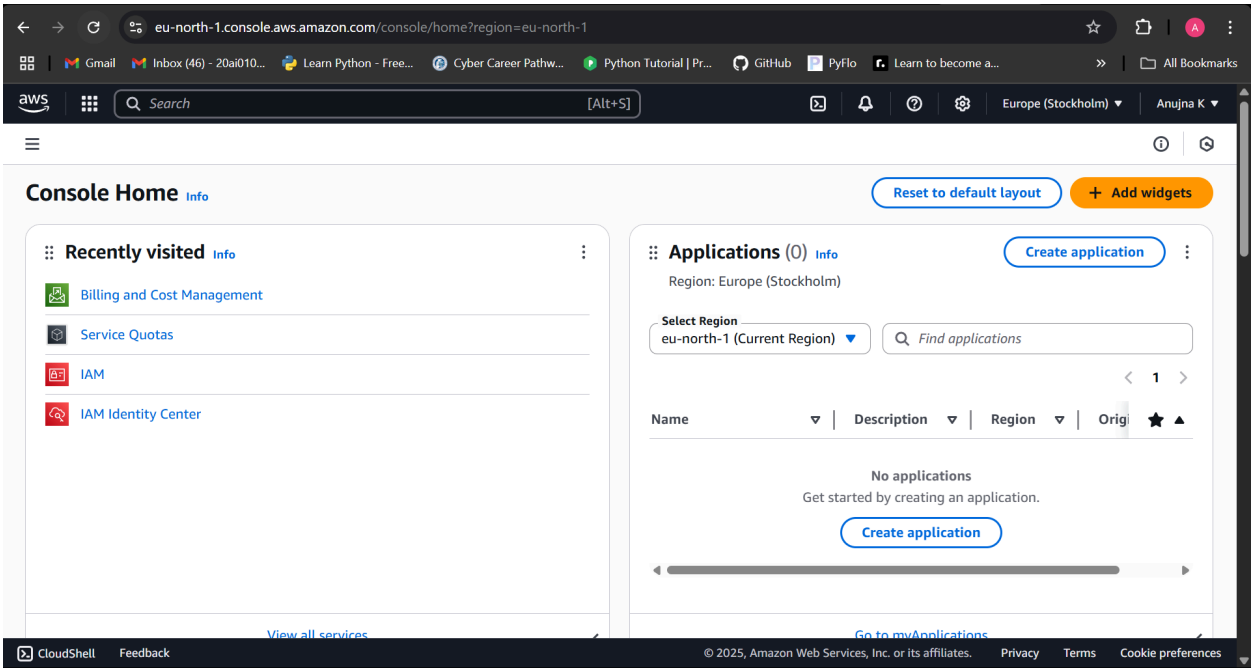
```
┌──(kali㉿kali)-[~/Downloads/dc4-docker]
└─$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: anusharao4520@gmail.com
Password:
WARNING! Your password will be stored unencrypted in /home/kali/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded

┌──(kali㉿kali)-[~/Downloads/dc4-docker]
└─$ docker images
REPOSITORY                   TAG       IMAGE ID       CREATED        SIZE
<none>                       <none>    48a0db00d404   2 hours ago    2.42GB
dc4-container                latest    c5256f14b4ce   2 hours ago    2.42GB
<none>                       <none>    cd99c8359d14   2 hours ago    2.42GB
<none>                       <none>    147a9431e4e5   2 hours ago    2.42GB
dc4                          latest    e050eb2e5992   22 hours ago   5.44GB
<none>                       <none>    2a3e68654697   22 hours ago   5.37GB
gcr.io/k8s-minikube/kicbase  v0.0.46   e72c4cbe9b29   2 months ago   1.31GB
```

```
┌──(kali㉿kali)-[~/Downloads/dc4-docker]
└─$ docker tag c5256f14b4ce kanusharao/cloud-lab:dc4

┌──(kali㉿kali)-[~/Downloads/dc4-docker]
└─$ docker images
REPOSITORY                   TAG       IMAGE ID       CREATED        SIZE
<none>                       <none>    147a9431e4e5   2 hours ago    2.42GB
<none>                       <none>    48a0db00d404   2 hours ago    2.42GB
kanusharao/cloud-lab         dc4       c5256f14b4ce   2 hours ago    2.42GB
dc4-container                latest    c5256f14b4ce   2 hours ago    2.42GB
<none>                       <none>    cd99c8359d14   2 hours ago    2.42GB
dc4                          latest    e050eb2e5992   22 hours ago   5.44GB
<none>                       <none>    2a3e68654697   22 hours ago   5.37GB
gcr.io/k8s-minikube/kicbase  v0.0.46   e72c4cbe9b29   2 months ago   1.31GB

┌──(kali㉿kali)-[~/Downloads/dc4-docker]
└─$ docker push kanusharao/cloud-lab:dc4

The push refers to repository [docker.io/kanusharao/cloud-lab]
058270631f00: Pushed
738005c3537e: Pushed
4b7c01ed0534: Mounted from library/ubuntu
dc4: digest: sha256:9dd245707e91835fa2b36d1a043b77d83479fc16c9ebd3a0bacd9b6668b518e3 size: 955
```

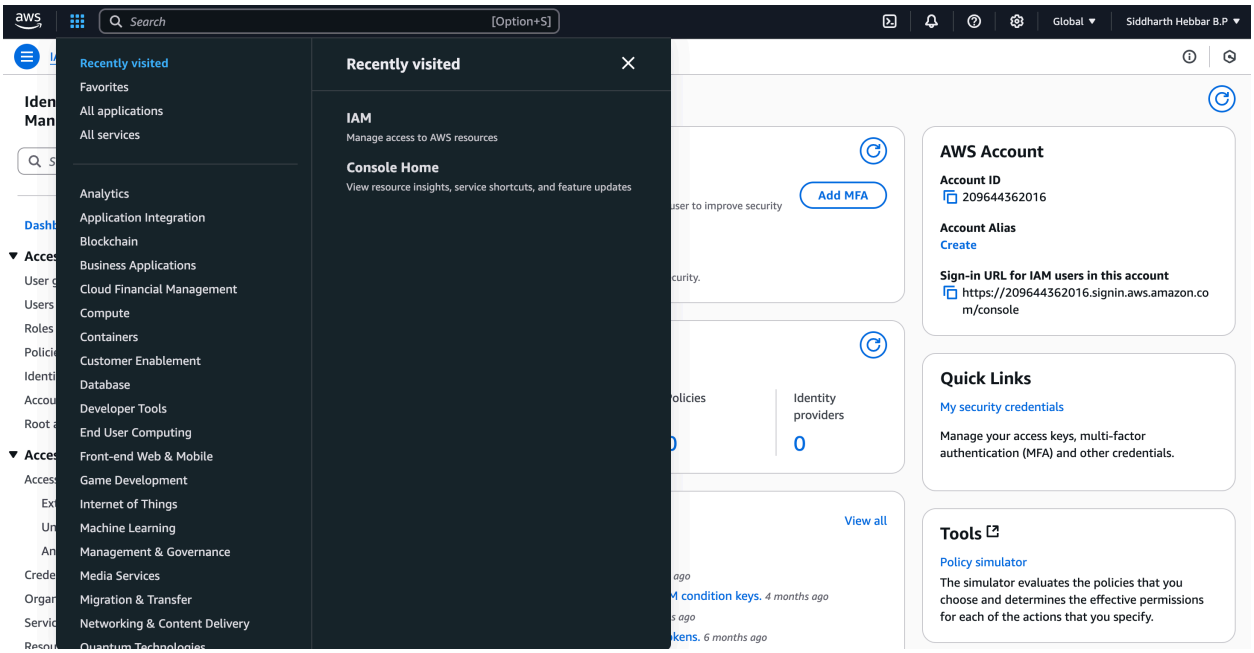


AWS

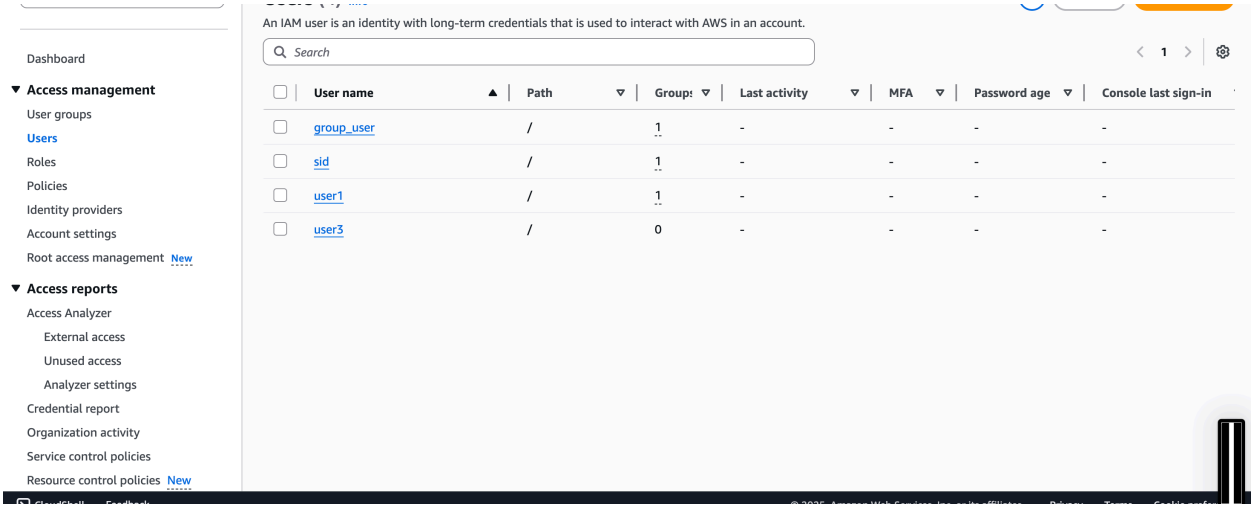
After signup console:



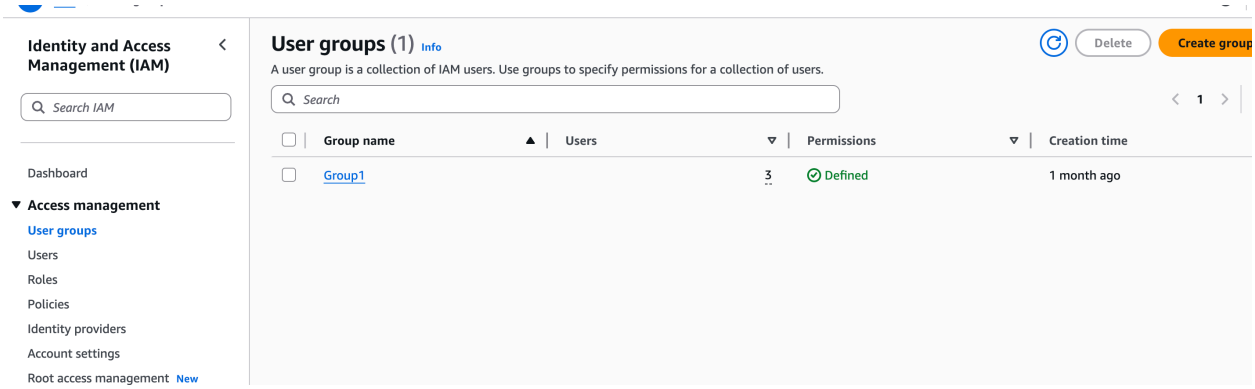
NAVIGATE TO IAM:



add users , grant him certain permission/Policies



create group with few policies

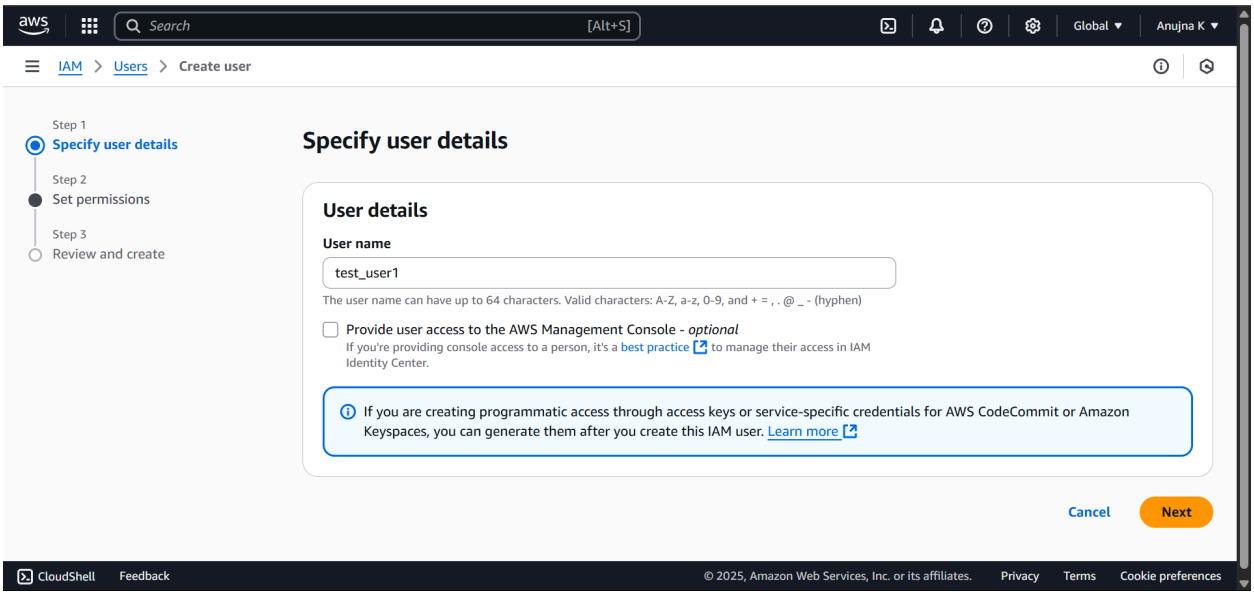


permission overide for user in group

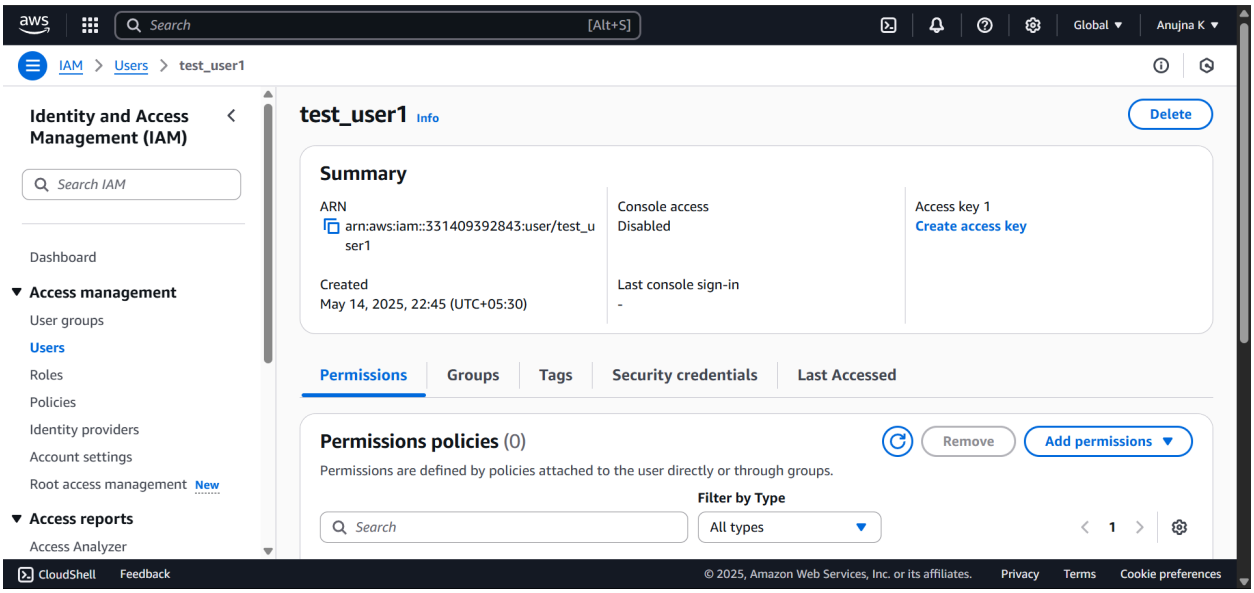
Using IAM to assign a user to access S3 bucket

→ Go to IAM [root user]

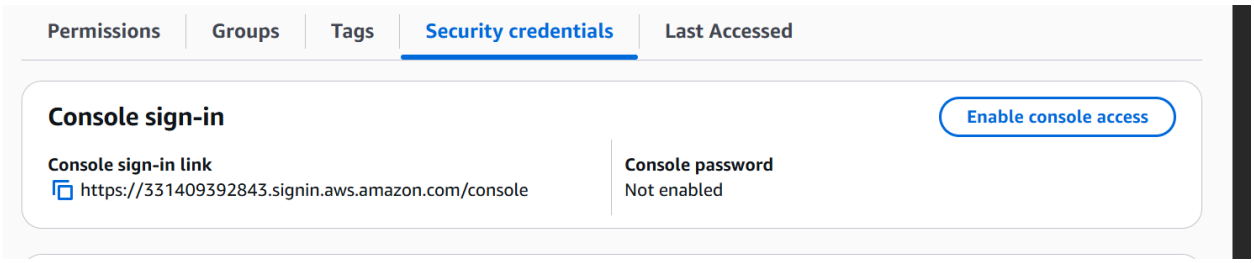
→ create users



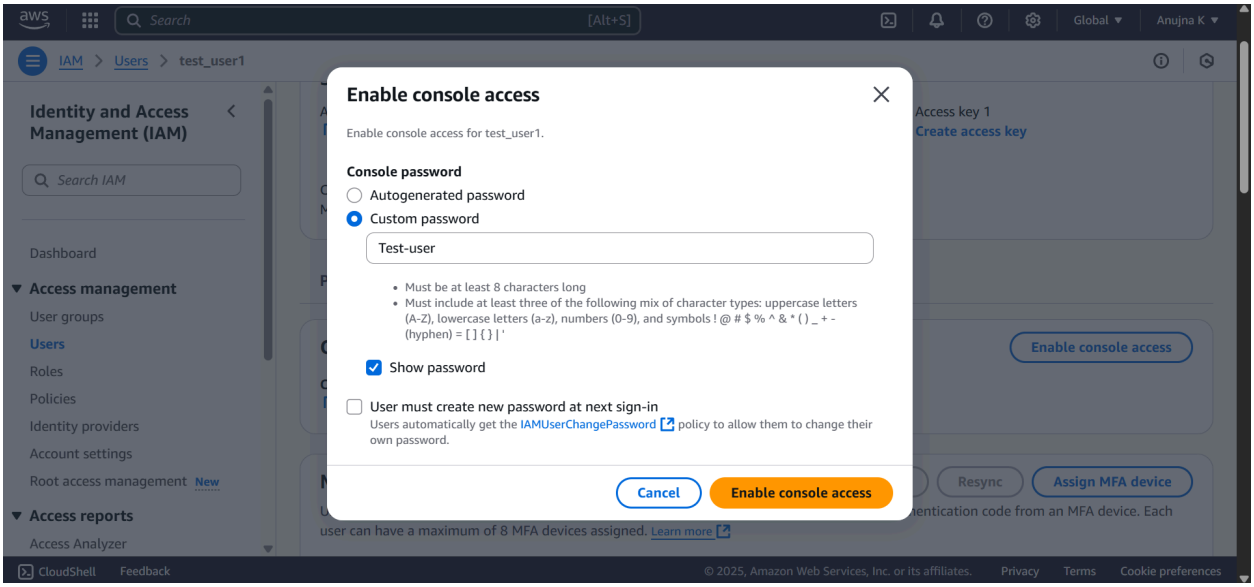
⭐ARN

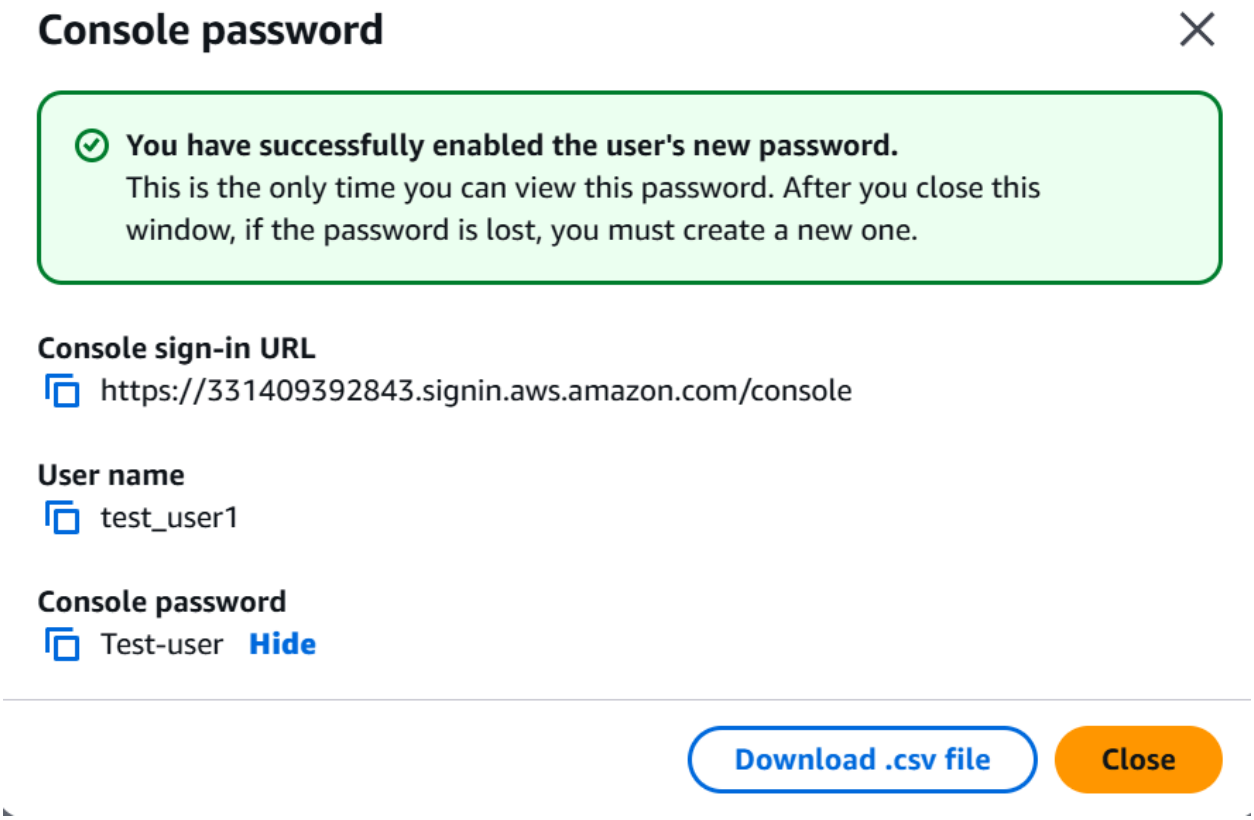


→ to enable console access to that user [test-user1]



password → Test-user

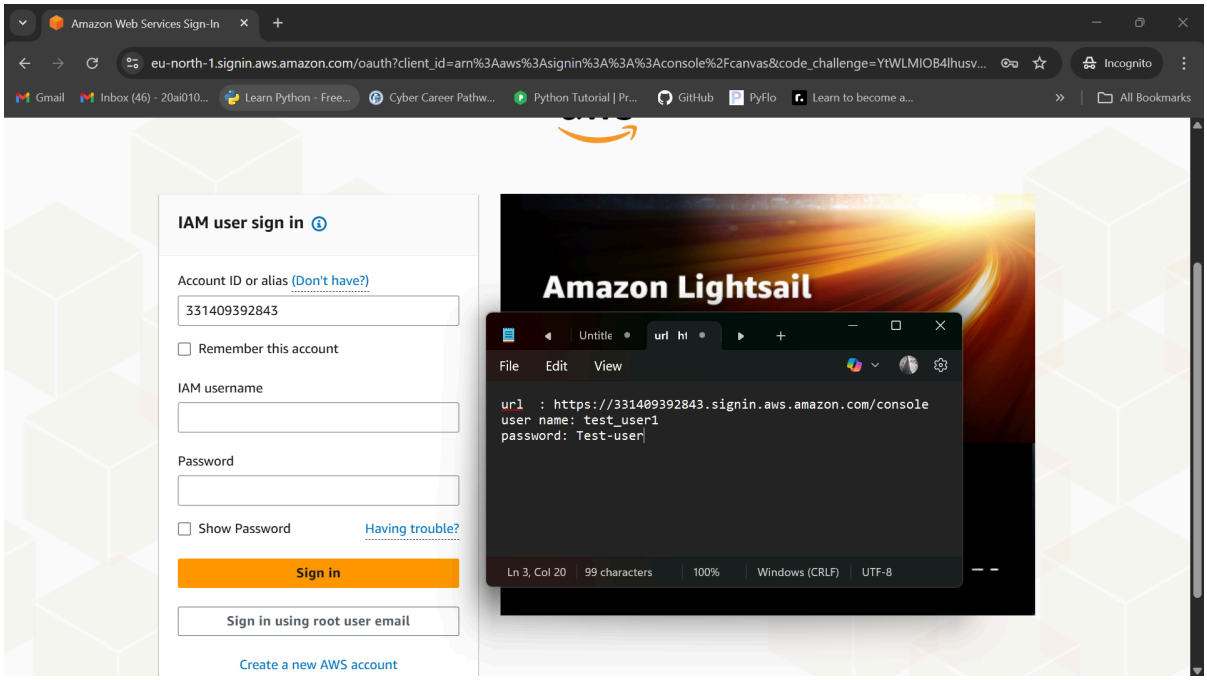


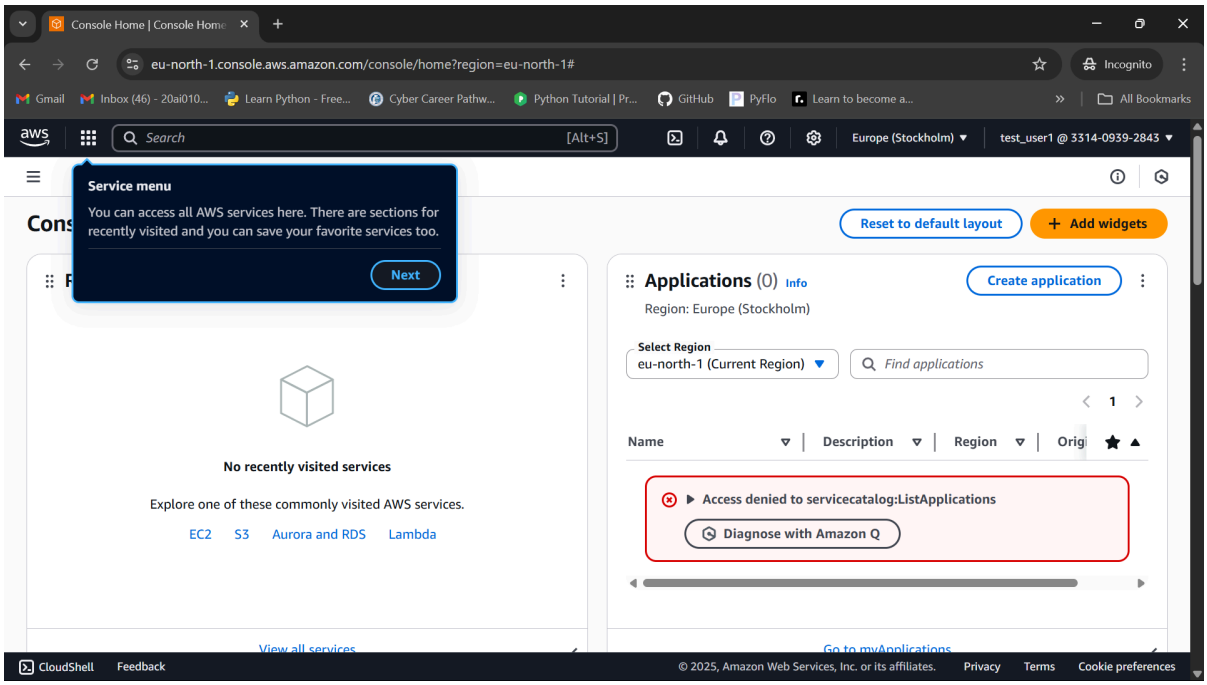


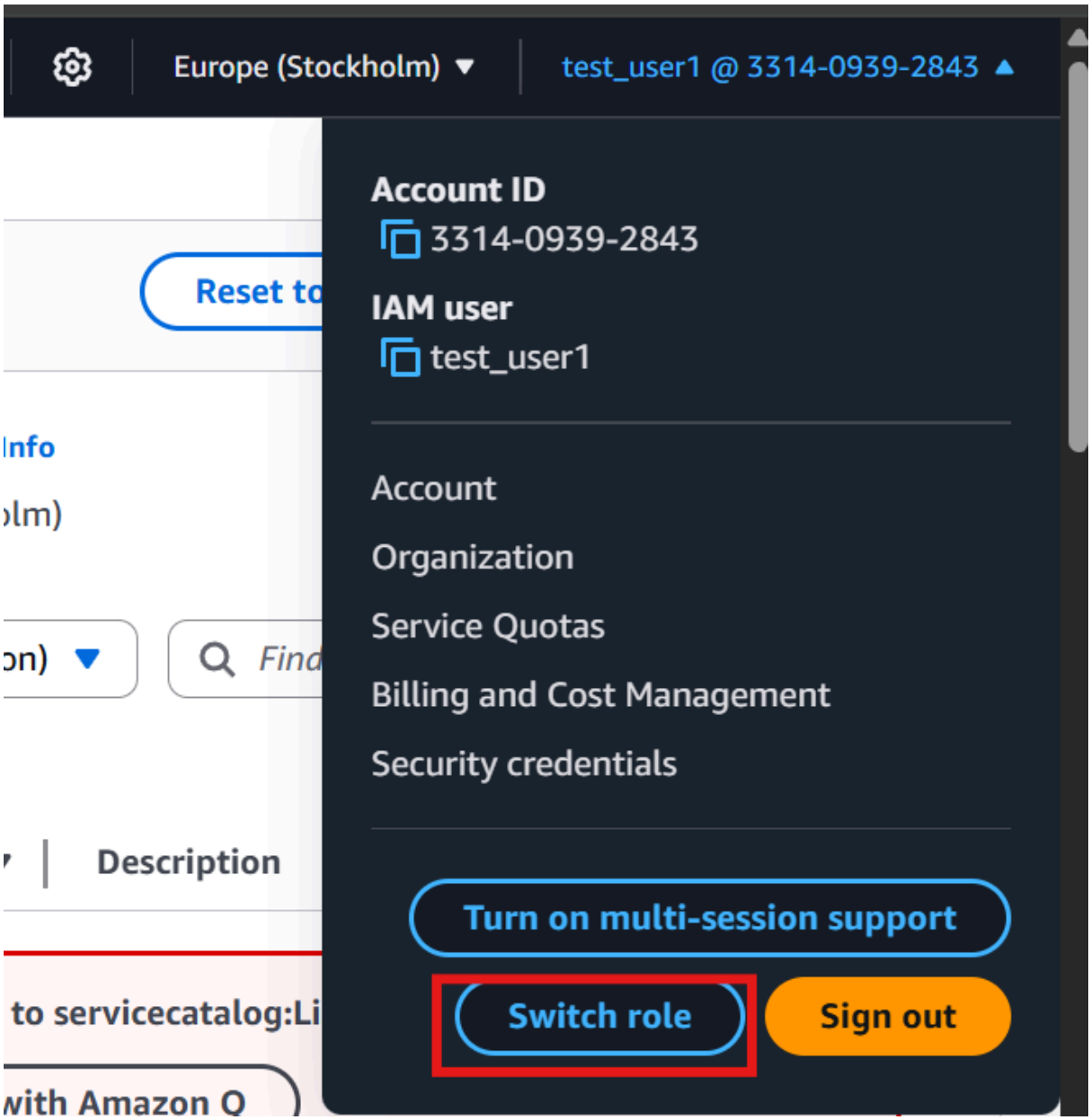
→ to perform login for this test user :

→ open a new window and copy the console sign-in url [can use incognito mode]

→ copy and paste username and password

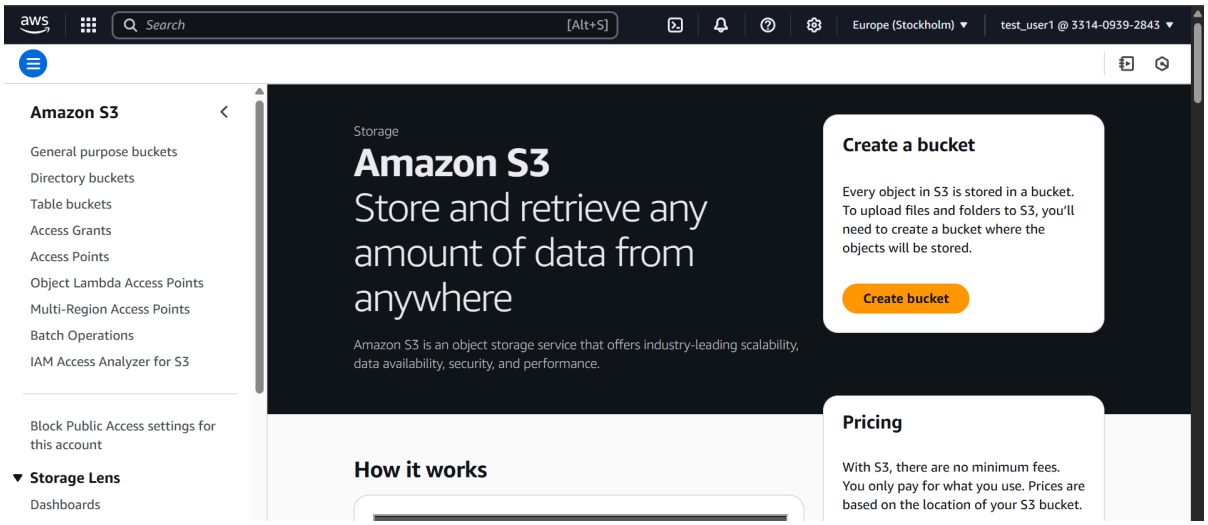


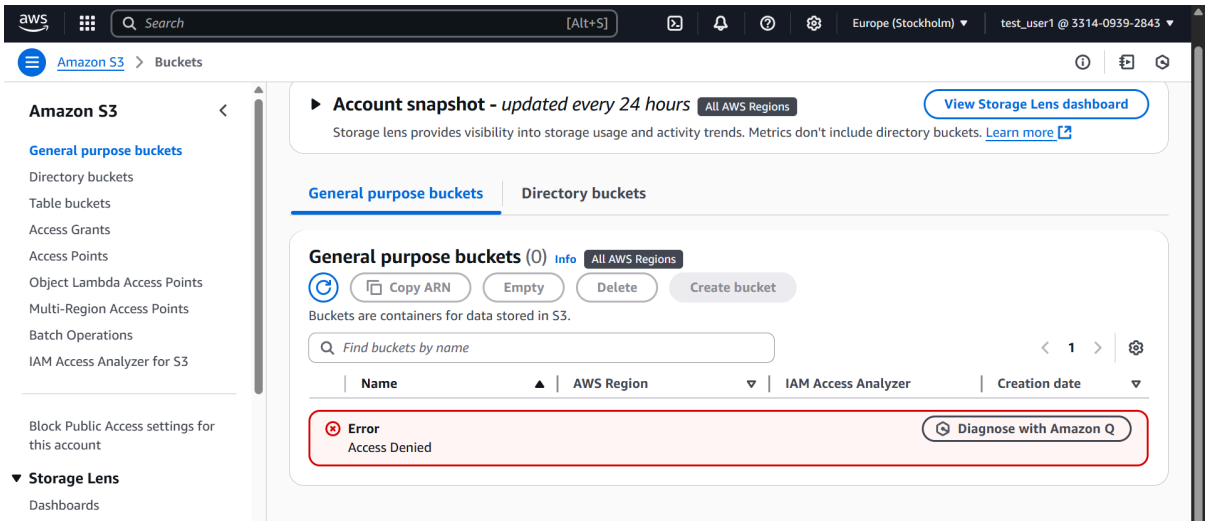




→ switch role means its not a root user, is an IAM user

→ click on S3 on dashboard:



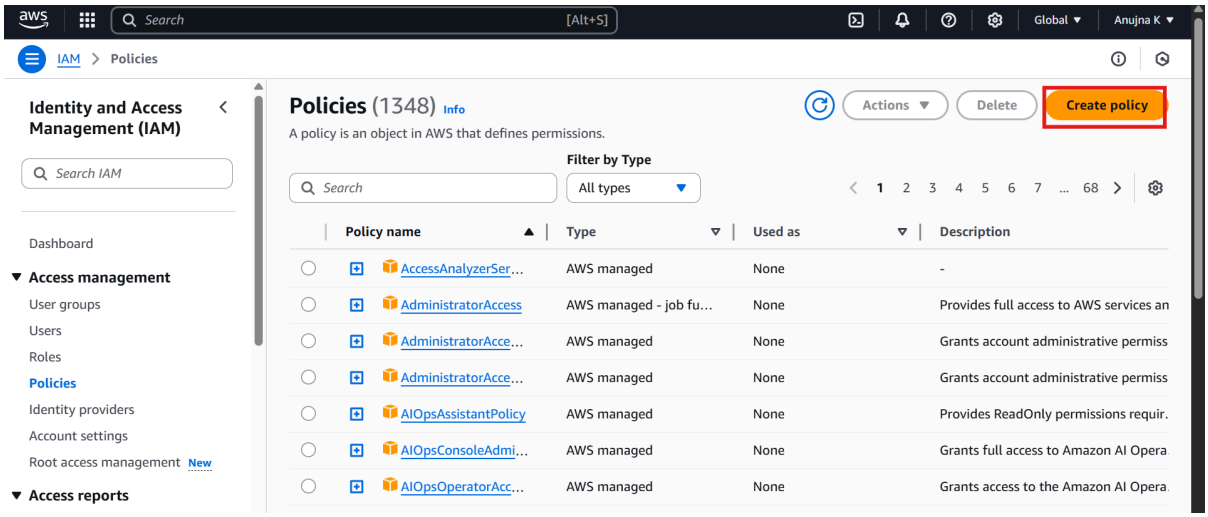


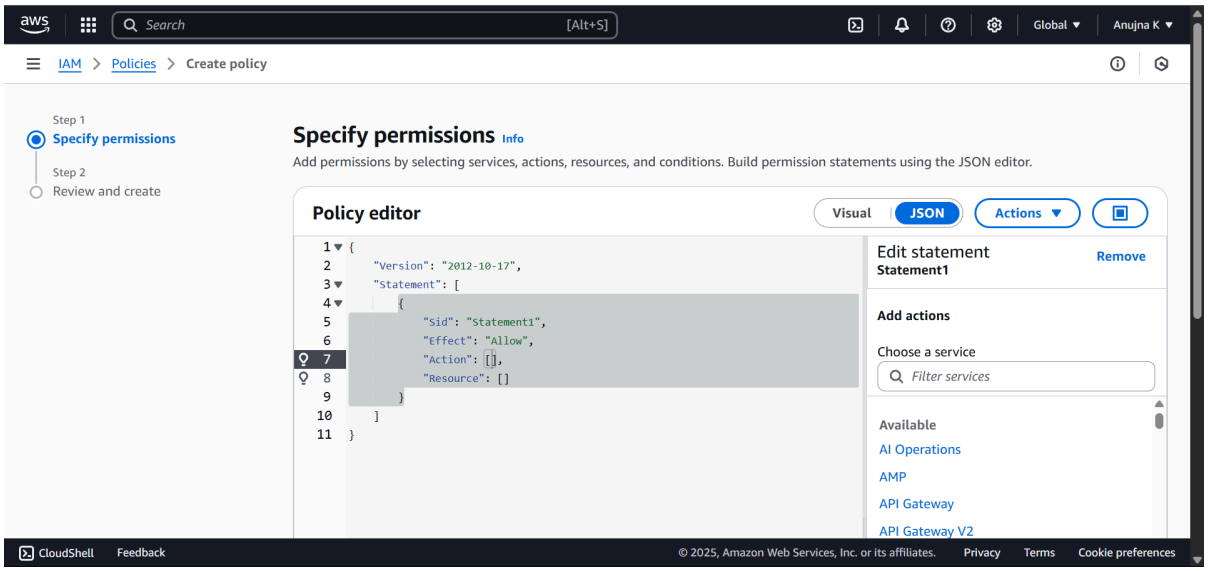
This user doesn't have the permission to access S3 buckets. The root user have the permission to access S3 buckets.

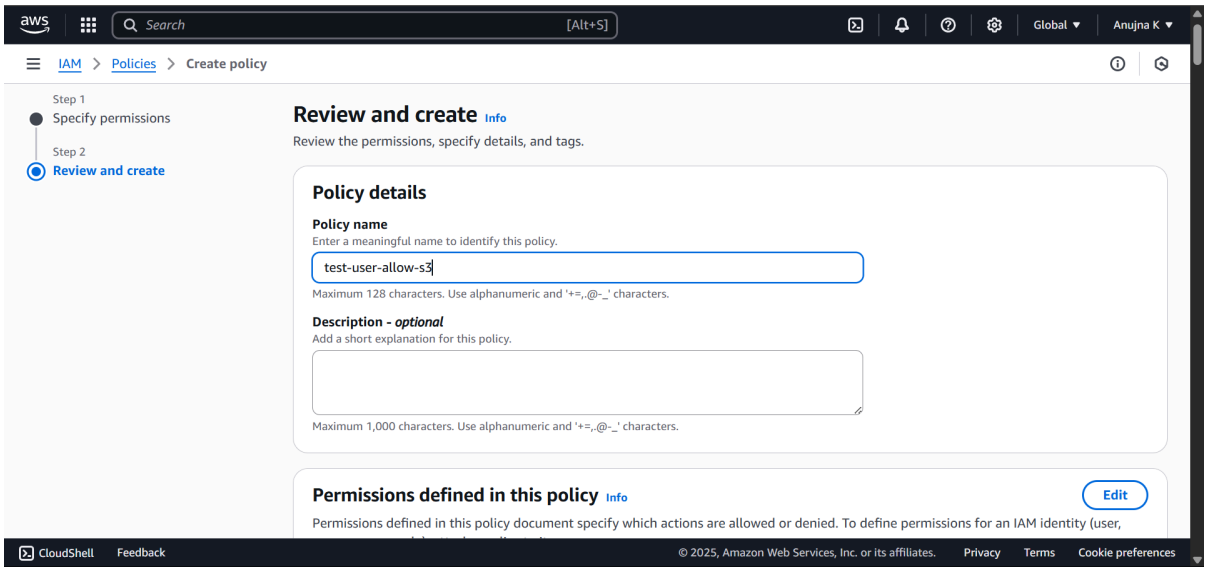
→ To create a policy for accessing S3 bucket for that user:

→ go to root user → then IAM → Policies → Create policy

we can then use the built-in policies or create a default policy(as in our case)

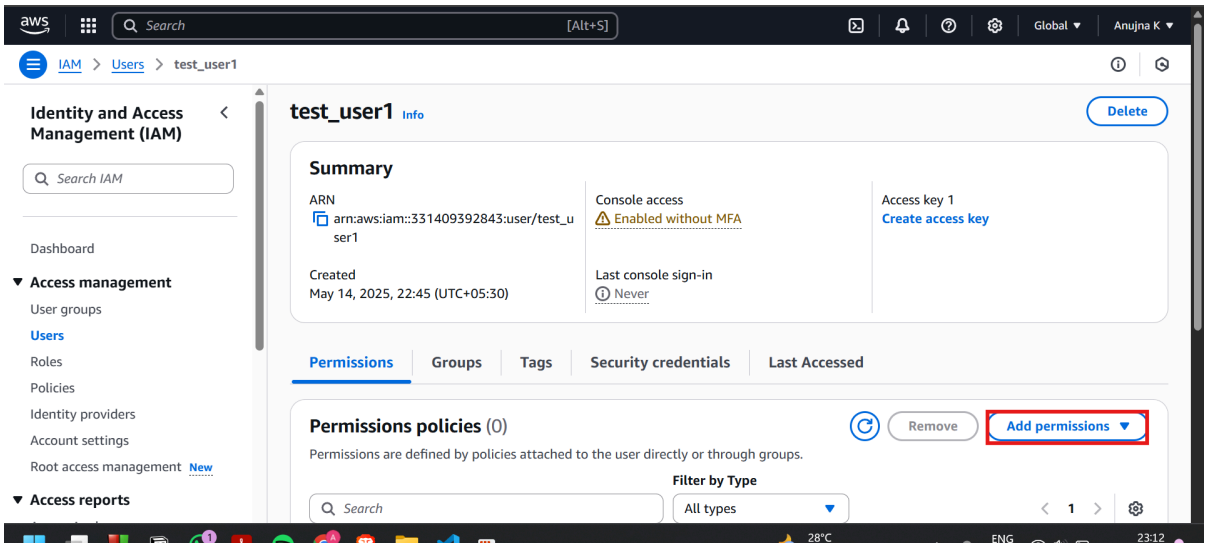

Search [Alt+S]
Global
Anujna K
IAM > Policies
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
Policies (1348) Info
A policy is an object in AWS that defines permissions.
Actions
Delete
Create policy
Filter by Type
Search
All types
1 2 3 4 5 6 7 ... 68
Policy name
Type
Used as
Description
AccessAnalyzerSer... AWS managed None -
AdministratorAccess AWS managed - job fu... None Provides full access to AWS services an
AdministratorAcce... AWS managed None Grants account administrative permiss
AdministratorAcce... AWS managed None Grants account administrative permiss
AIOpsAssistantPolicy AWS managed None Provides ReadOnly permissions requir.
AIOpsConsoleAdmi... AWS managed None Grants full access to Amazon AI Opera
AIOpsOperatorAcc... AWS managed None Grants access to the Amazon AI Opera

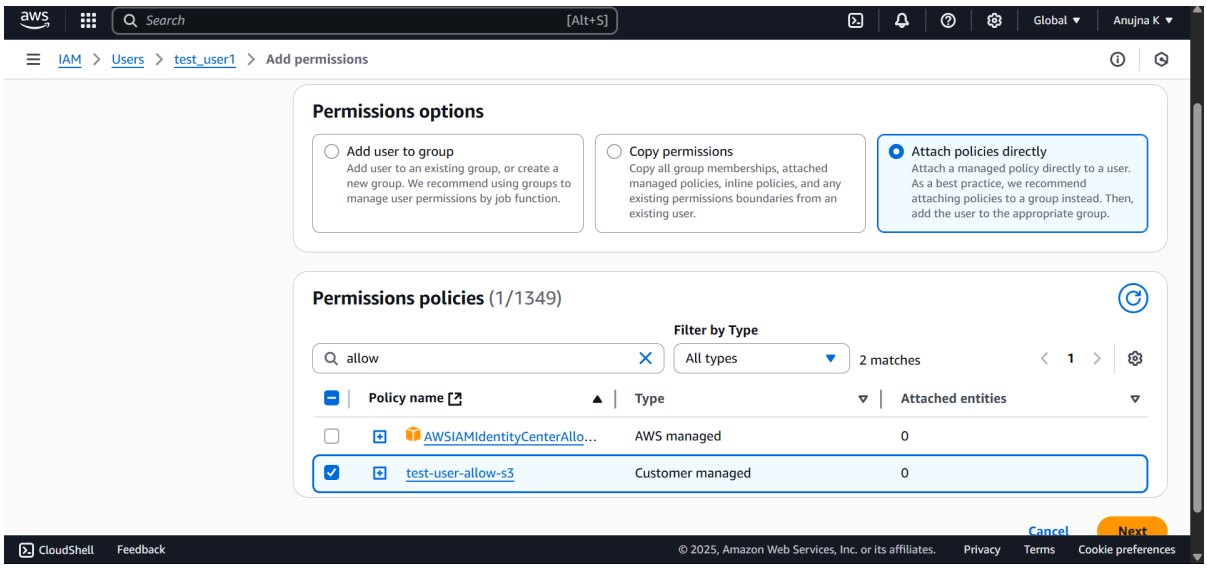

Search [Alt+S]
Global
Anujna K
IAM > Policies > Create policy
Step 1
Specify permissions
Step 2
Review and create
Specify permissions Info
Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.
Policy editor
Visual
JSON
Actions
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "Statement1",
"Effect": "Allow",
"Action": [],
"Resource": []
}
]
}
Edit statement
Statement1
Remove
Add actions
Choose a service
Filter services
Available
AI Operations
AMP
API Gateway
API Gateway V2
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

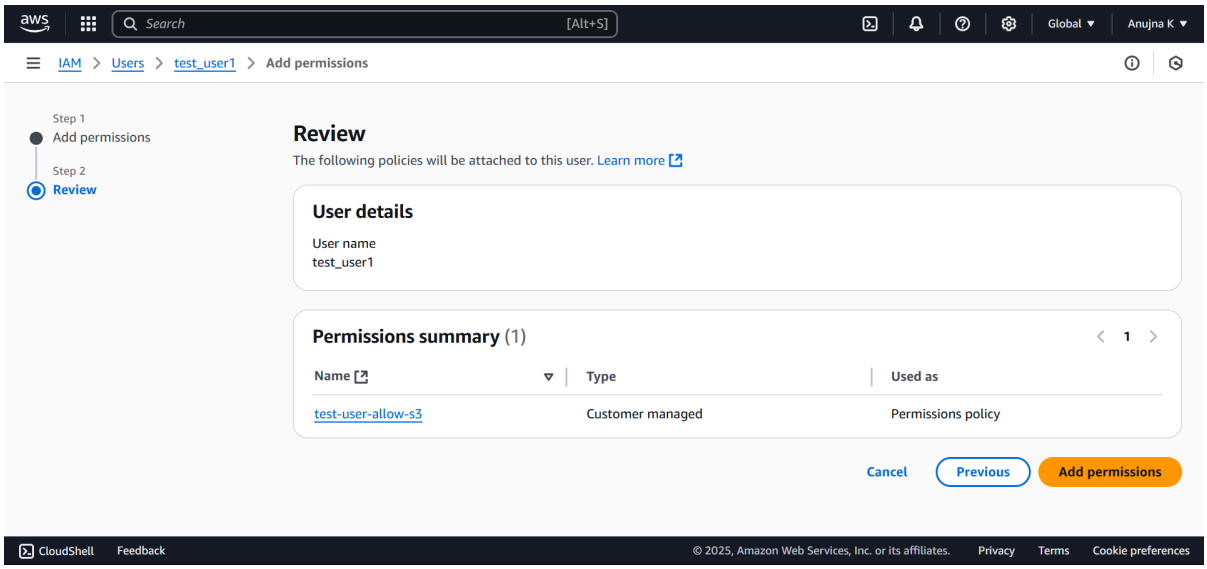


→ Assigning the policy to the user:

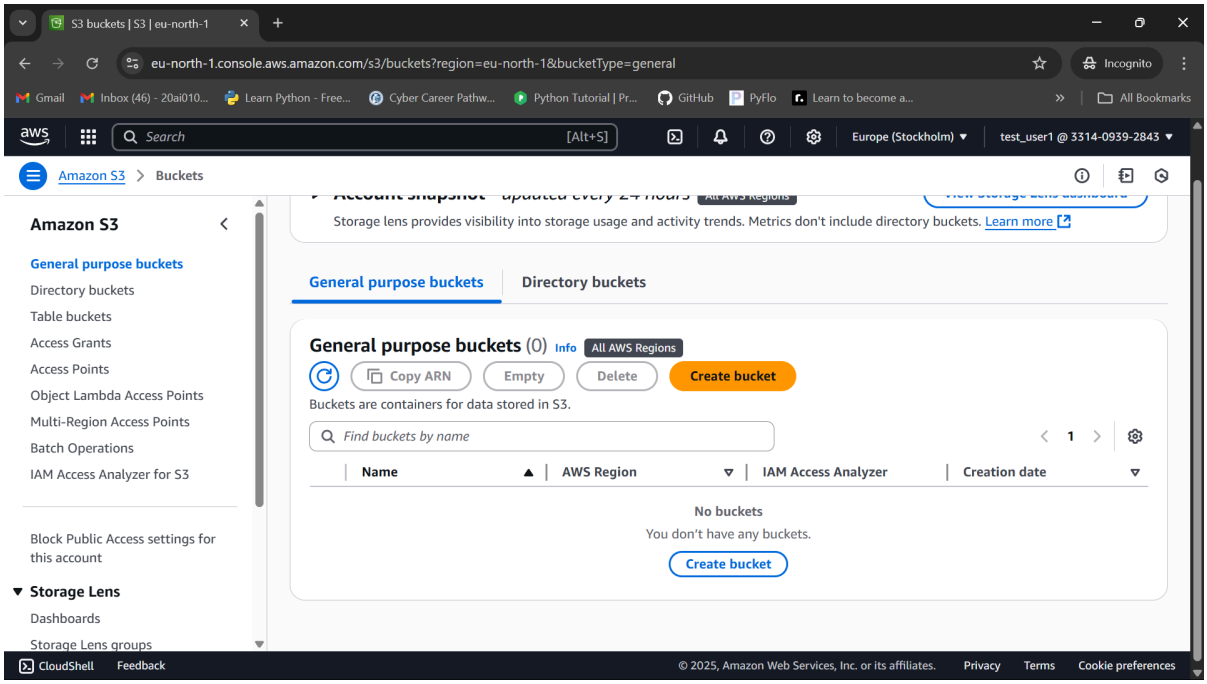
IAM → users → test_user → Add Permission


Search [Alt+S]
Global
Anujna K
IAM > Users > test_user1
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
test_user1 Info
Delete
Summary
ARN
arn:aws:iam::331409392843:user/test_user1
Console access
Enabled without MFA
Access key 1
Create access key
Created
May 14, 2025, 22:45 (UTC+05:30)
Last console sign-in
Never
Permissions
Groups
Tags
Security credentials
Last Accessed
Permissions policies (0)
Remove
Add permissions
Permissions are defined by policies attached to the user directly or through groups.
Filter by Type
Search
All types
1

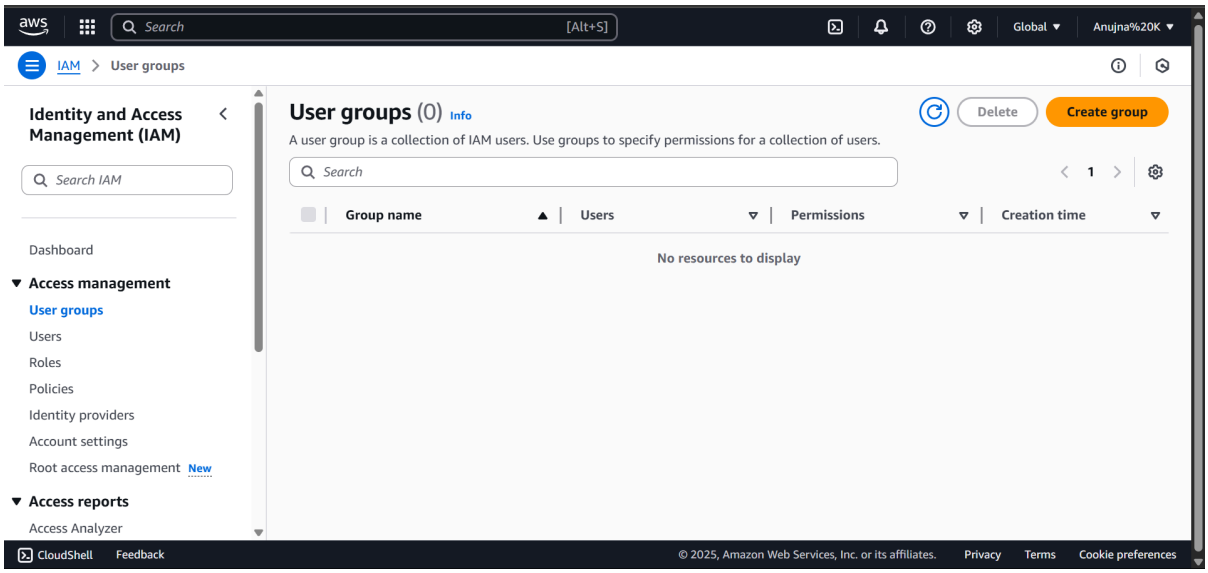

Search [Alt+S]
Global
Anujna K
IAM > Users > test_user1 > Add permissions
Permissions options
Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
Copy permissions
Copy all group memberships, attached managed policies, inline policies, and any existing permissions boundaries from an existing user.
Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.
Permissions policies (1/1349)
Filter by Type
allow
All types
2 matches
1
Policy name
Type
Attached entities
AWSIAMIdentityCenterAllo...
AWS managed
0
test-user-allow-s3
Customer managed
0
Cancel
Next
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



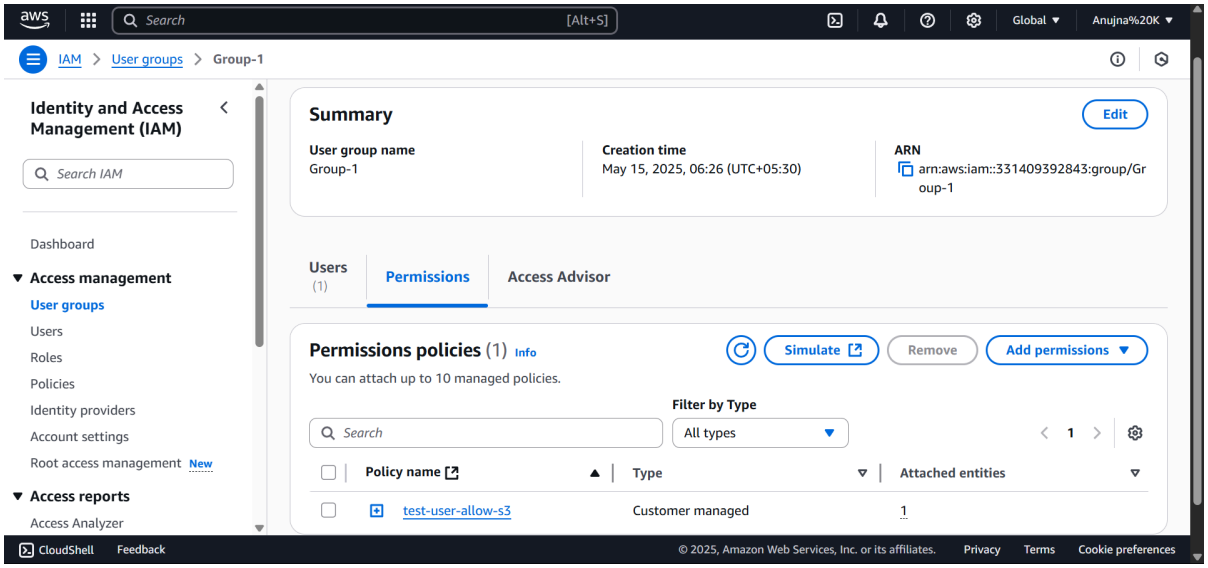
After adding the policy, now the test user has the access to S3 buckets -



→ Creating a group with multiple users



attach policies to the group, those policies are applicable to all users in the group



Doker

✅ Recommended: GUI via Web Browser (VNC/noVNC)

This method lets you run a full Linux GUI inside Docker, and access it via your browser on localhost.

🔧 Option A: Use

dorowu/ubuntu-desktop-lxde-vnc

as a base

🔁 Step 1: Pull GUI-Enabled Docker Base

docker pull dorowu/ubuntu-desktop-lxde-vnc

This gives you:

- Full GUI desktop
- Accessible on localhost:8080 via browser (noVNC)
- Also exposes traditional VNC on 5901

⚙️ Step 2: Create Your Dockerfile with Vulnerable App

Create a Dockerfile like this:

FROM dorowu/ubuntu-desktop-lxde-vnc

USER root

Install vulnerable web app dependencies

RUN apt-get update && apt-get install -y \

apache2 php php-mysqli git unzip

Set up DVWA or any other vulnerable web app

RUN git clone https://github.com/digininja/DVWA /var/www/html && \

chmod -R 755 /var/www/html && \

```
chown -R www-data:www-data /var/www/html

# Start Apache automatically

RUN echo "service apache2 start" >> /root/.bashrc

EXPOSE 8080
```

🛠️ **Step 3: Build the Docker Image**

```
docker build -t vuln-gui-app .
```

▶️ **Step 4: Run the Container**

```
docker run -d -p 8080:80 -p 6080:80 vuln-gui-app
```

- Access GUI via browser: http://localhost:6080
- Access DVWA web app via: http://localhost:8080
- Intercept with Burp by routing browser traffic through Burp's proxy.

🧪 What You Can Do

- Launch a Linux desktop GUI in your browser.
- Launch Firefox inside container, browse your vulnerable app.
- Route traffic through Burp Suite (configure Firefox proxy to 127.0.0.1:8080).
- Perform attacks (SQLi, XSS, etc.) visually.

❗ Notes

- The ISO itself isn't directly used in this method — instead, you build a vulnerable GUI system in Docker, similar to what the ISO contains.
- You can mimic an ISO system by copying its files or setting up the same services inside the Docker image.
- If you absolutely need to convert an ISO directly, you'll need to extract its filesystem and run it with VNC manually configured — more fragile and complex.

Do You Want:

- A prebuilt GUI + vulnerable system image to try right away?
- Instructions on using your ISO directly inside a VNC-enabled Docker container?
- Help setting up DVWA/bWAPP/Metasploitable-like environments visually?

Let me know your ISO name or send it over, and I'll tailor it exactly for you.