Saurabh Sanjiv Jadhav

AWS Project.

Cloud-Based File Sharing System

Overview:

The Cloud-Based File Sharing System is a secure, scalable, and cost-efficient web application developed using Python (Flask framework) and deployed on AWS infrastructure. The system enables users to upload, manage, and download files using a user-friendly web interface while ensuring that access to files is protected using IAM policies and signed URLs.

Key Features :

- **Web Application Interface:**
  Developed a responsive and interactive user interface using **Flask** and HTML templates. Integrated animations and styling for a professional user experience.
- **Secure File Upload & Download:**
  Files uploaded via the web app are stored directly in an **Amazon S3 bucket**. Access to files is secured using **pre-signed URLs**, allowing time-limited access to download the files securely.
- **User-Based Access Control:**
  Configured **IAM roles and policies** to control who can upload/download/view files. This ensures that only authorized users or applications have access to the storage resources.
- **Hosting on EC2 with Nginx & Gunicorn:**
  The Flask app is hosted on an **Amazon EC2 instance (Linux AMI)** using **Gunicorn** as the WSGI server and **Nginx** as a reverse proxy. This setup ensures high performance and production readiness.
- **Static and Dynamic Content Handling:**
  Utilized **Amazon S3** for file storage and **CloudFront** as a CDN (Content Delivery Network) to speed up file access globally.
- **Domain-Free Access (Public IP-Based):**
  The system was deployed using the **EC2 public IP** without a custom domain name. All services were configured to work seamlessly via IP access.

AWS Services Used :

Service	Purpose
EC2	Hosting the Flask application on a Linux virtual machine
S3	Secure and scalable file storage
IAM	Role and policy management for access control
CloudFront	Optional use for faster file delivery via CDN

Tech Stack :

- Backend: Python (Flask)
- Frontend: HTML, CSS (with animations)
- Server: Gunicorn, Nginx
- Cloud: AWS EC2, S3, IAM, CloudFront
- Tools: systemd, pip3, boto3

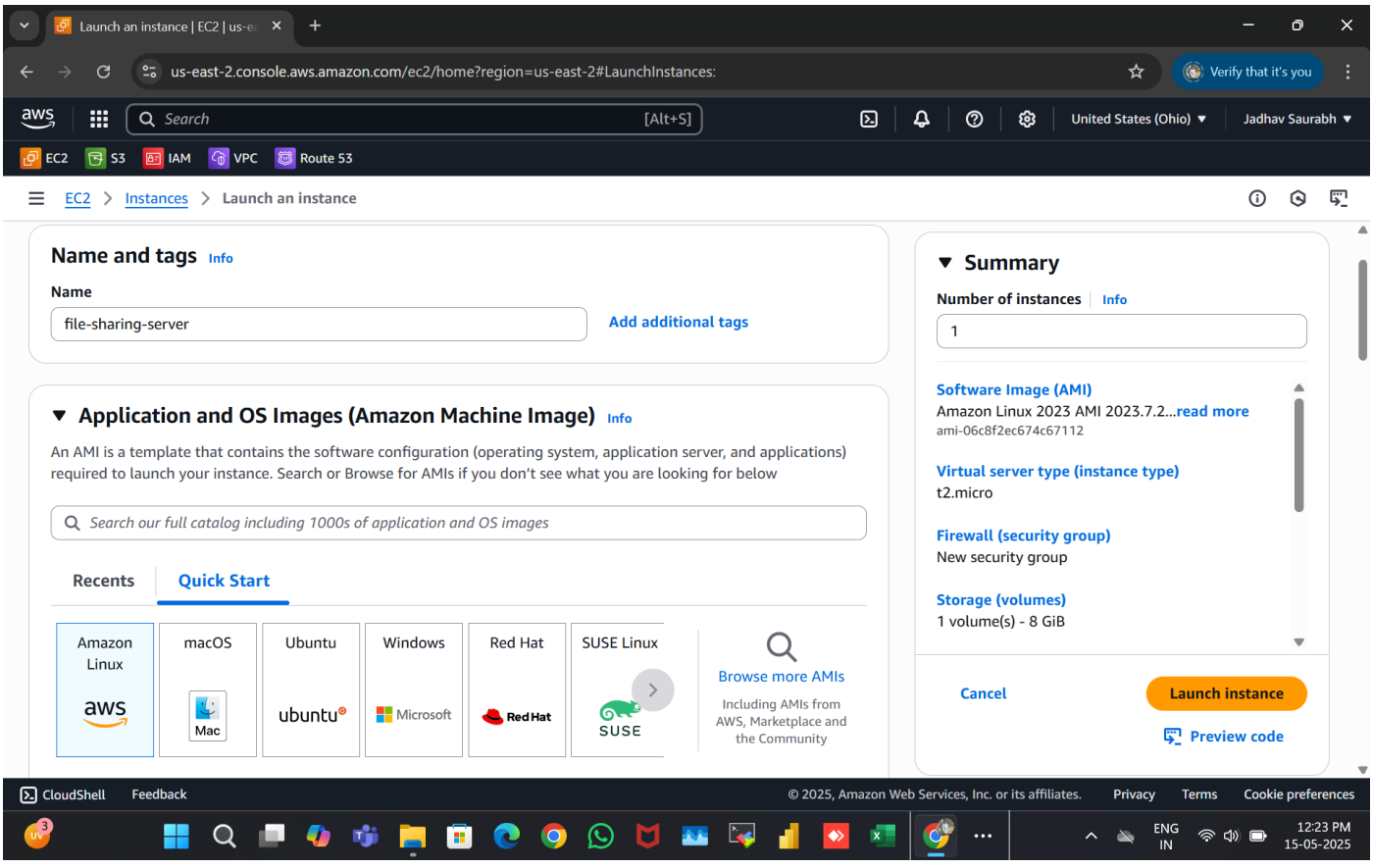
Security Implementation :

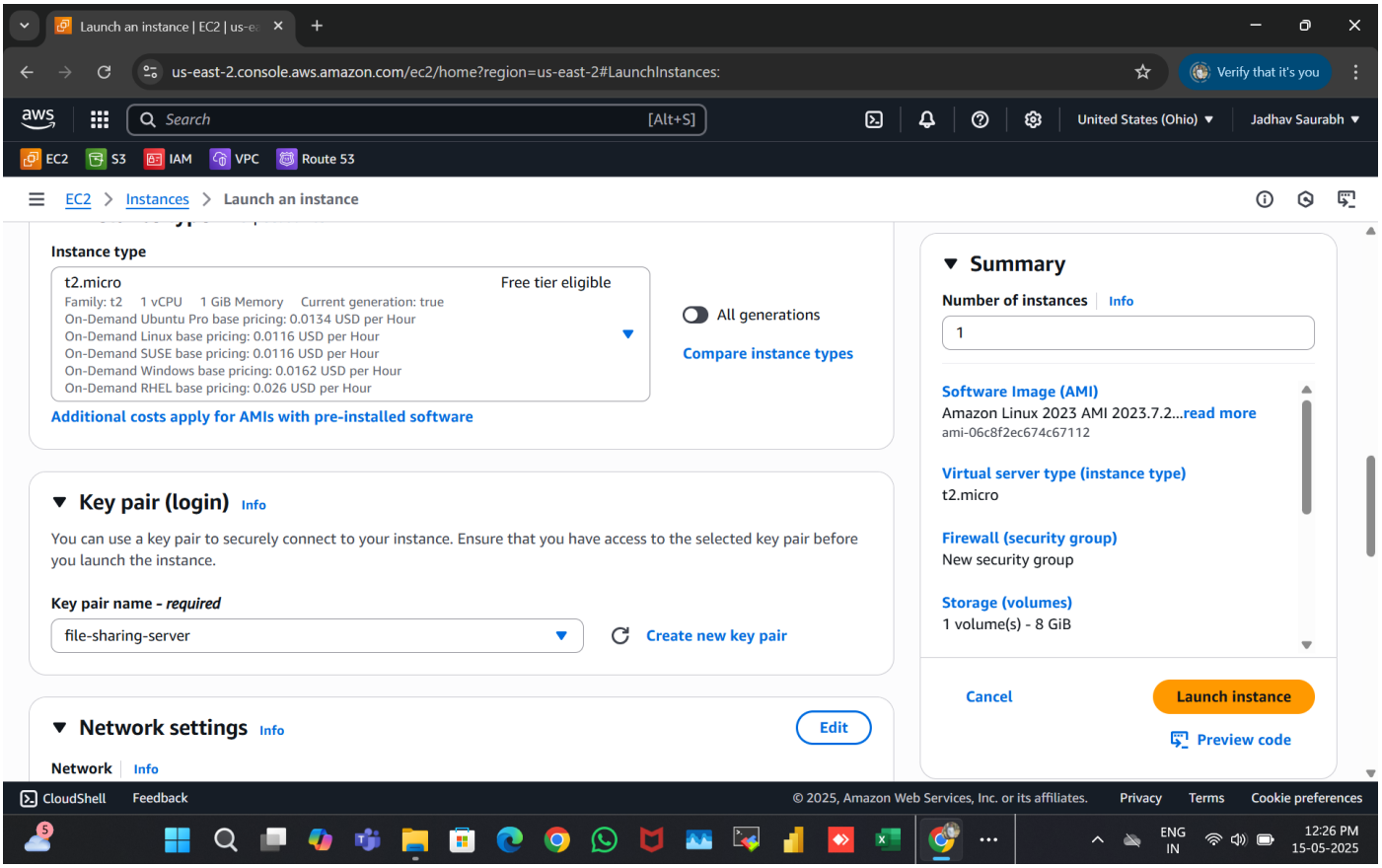
- Used IAM policies to restrict S3 access based on roles.
- Ensured HTTPS access to CloudFront URLs for encrypted file access (if using CloudFront).

Deployment Process Summary :

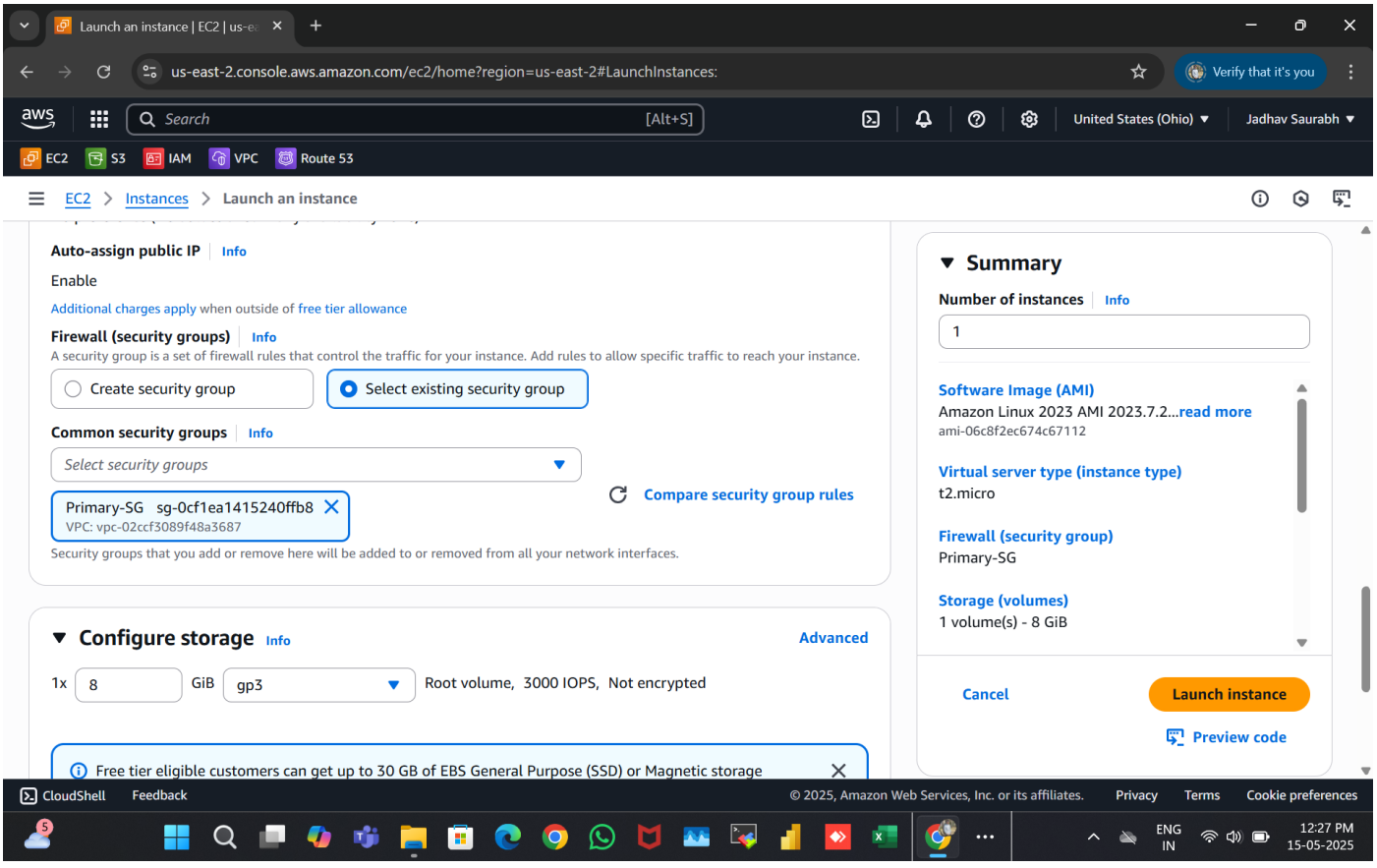
<u>Configure as follows :</u>

- **Name**: file-sharing-server
- **AMI**: Amazon Linux 2 (64-bit, x86)
- **Instance Type**: t2.micro (Free Tier eligible)
- **Key Pair**: Create new or use existing key pair (e.g., file-sharing-server.pem)
- **Network Settings**:
  - Allow **SSH (port 22)** from your IP.
  - Allow **HTTP (port 80)** from anywhere (0.0.0.0/0).
  - Allow **HTTPS (port 443)** from anywhere (0.0.0.0/0).
- **Storage**: Minimum 8 GB (can increase if needed)

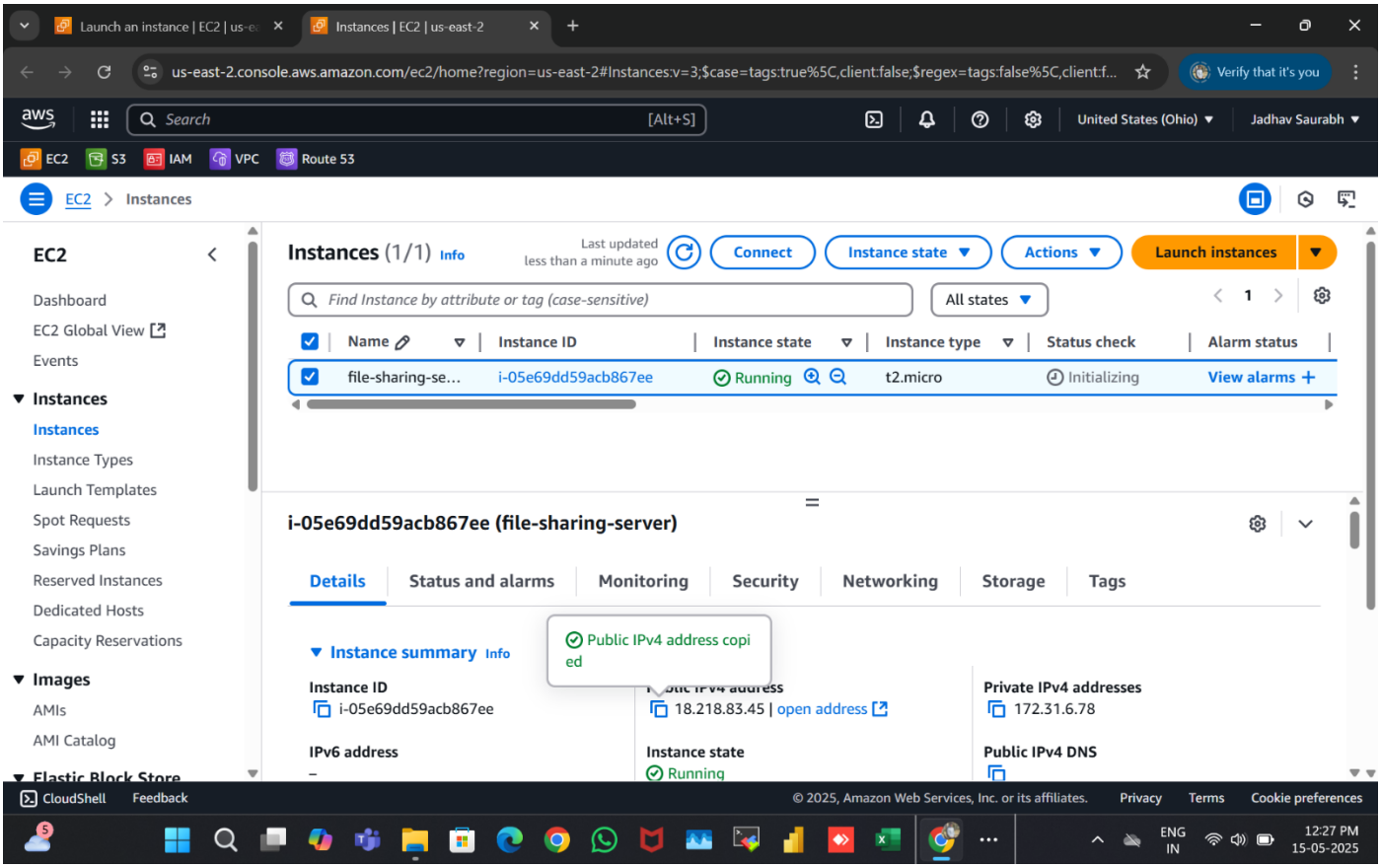




Assign Security groups.



Ec2 Instance is Successfully Launch. And Copy IP Address



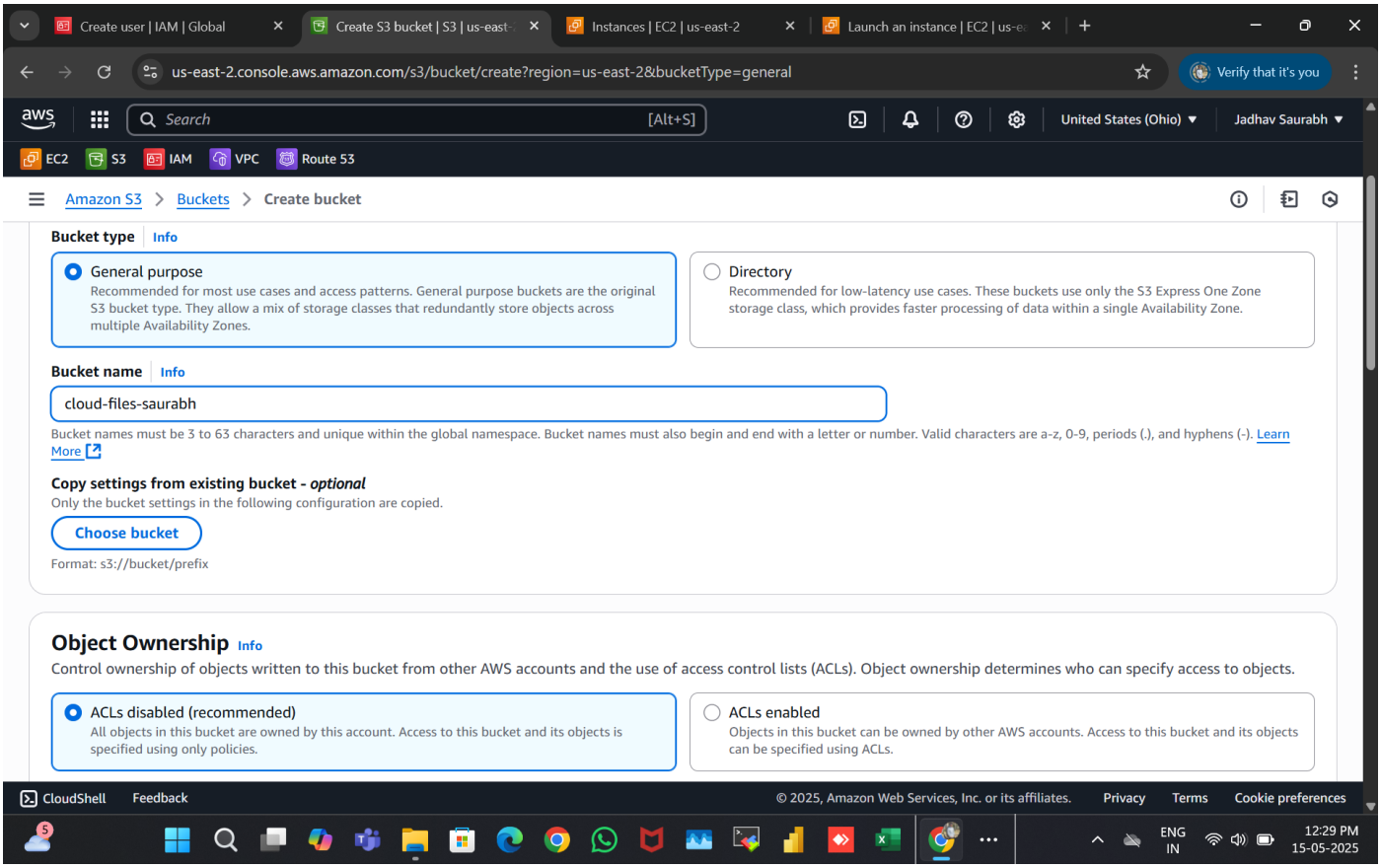
Purpose of Amazon S3 in the Project

Amazon S3 (Simple Storage Service) is used as the **central file storage system** for the Cloud-Based File Sharing Web App. Here's the purpose it serves:

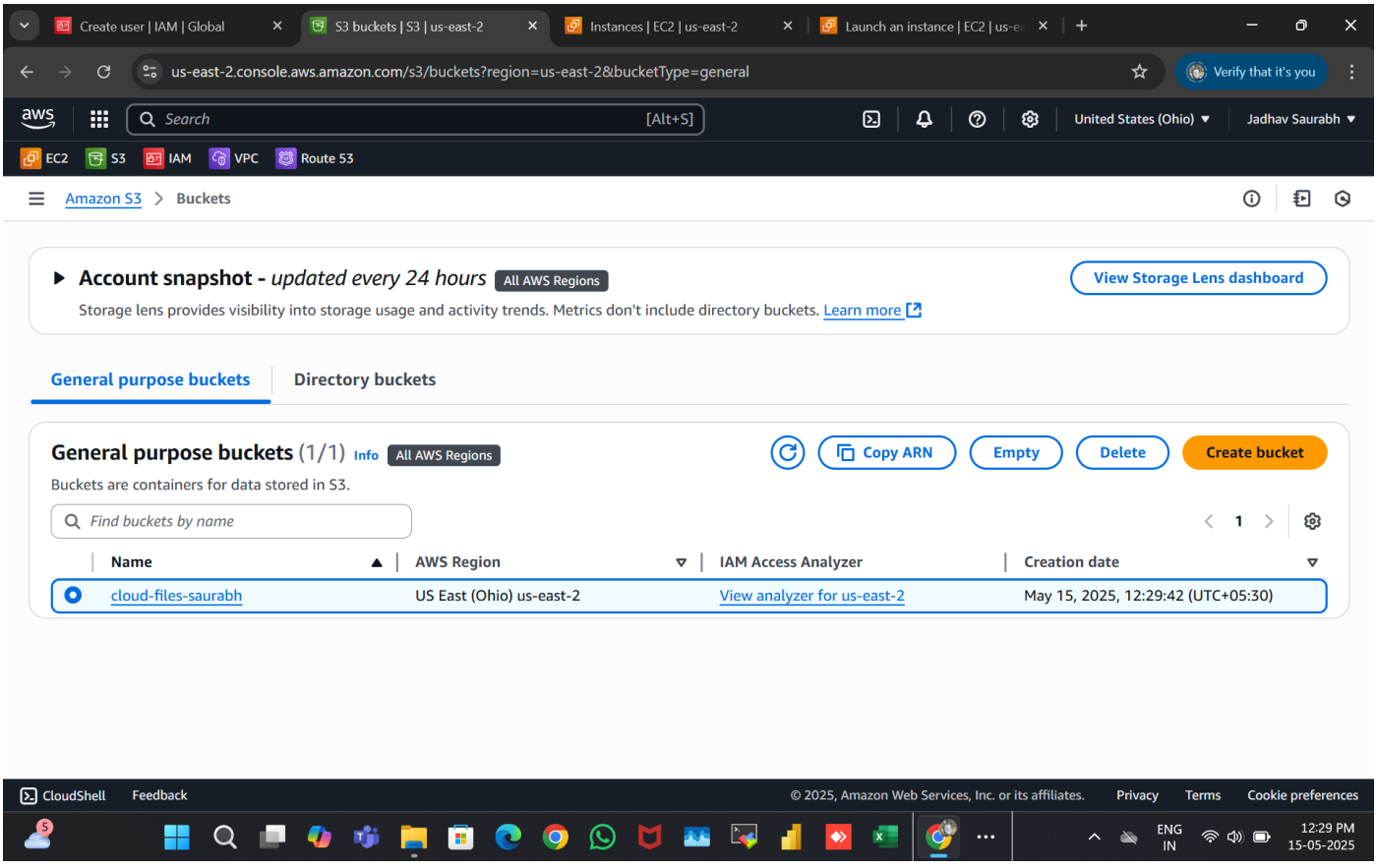
Create a New Bucket

Create an S3 Bucket

1. Go to AWS Console -S3 - **Create bucket**.
2. Name: **cloud-files-saurabh**
3. Region: Choose same as EC2 (e.g., us-east-1)
4. Uncheck “Block all public access”
5. Enable **Bucket versioning**
6. Click **Create Bucket**



Bucket is ready.

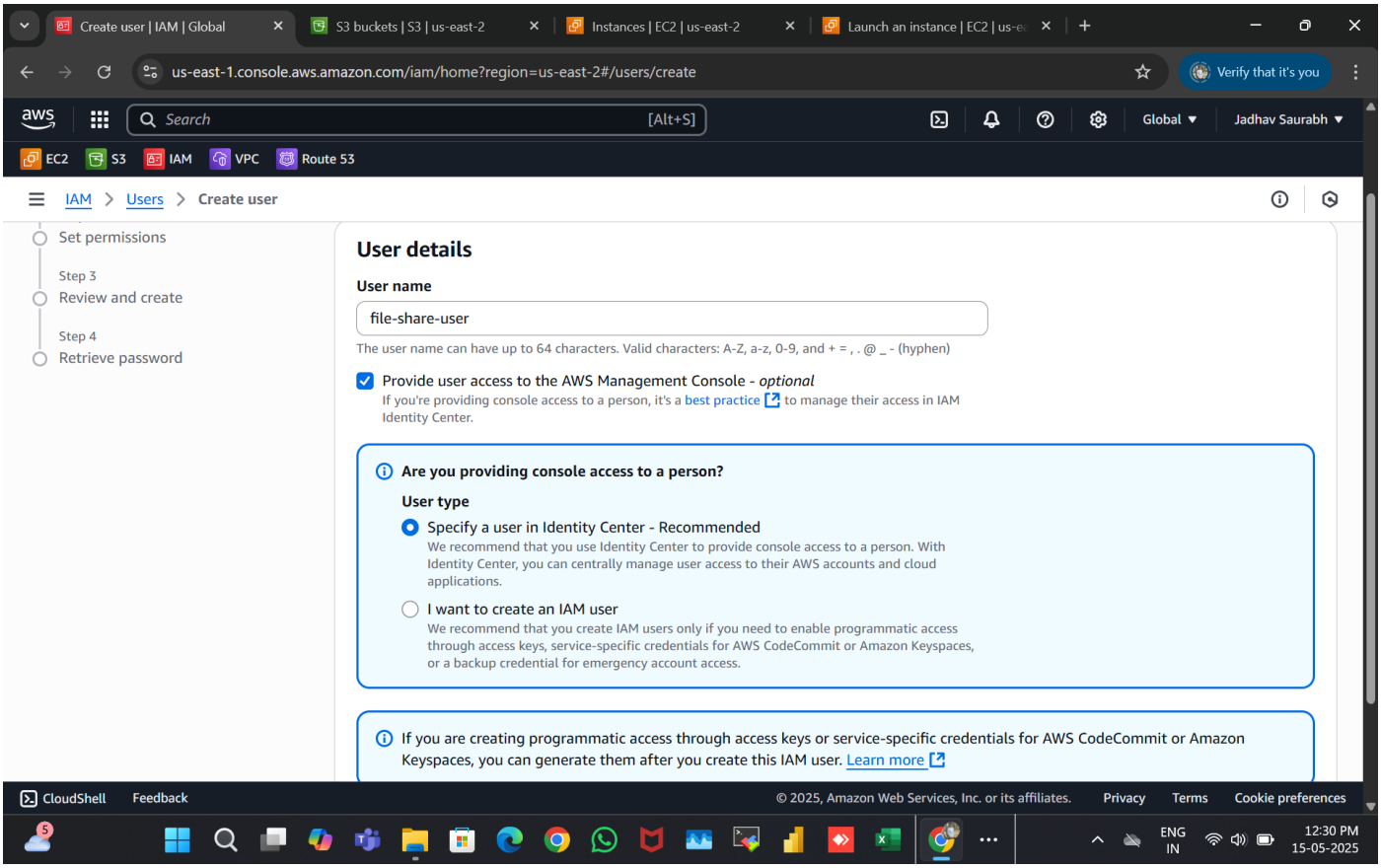


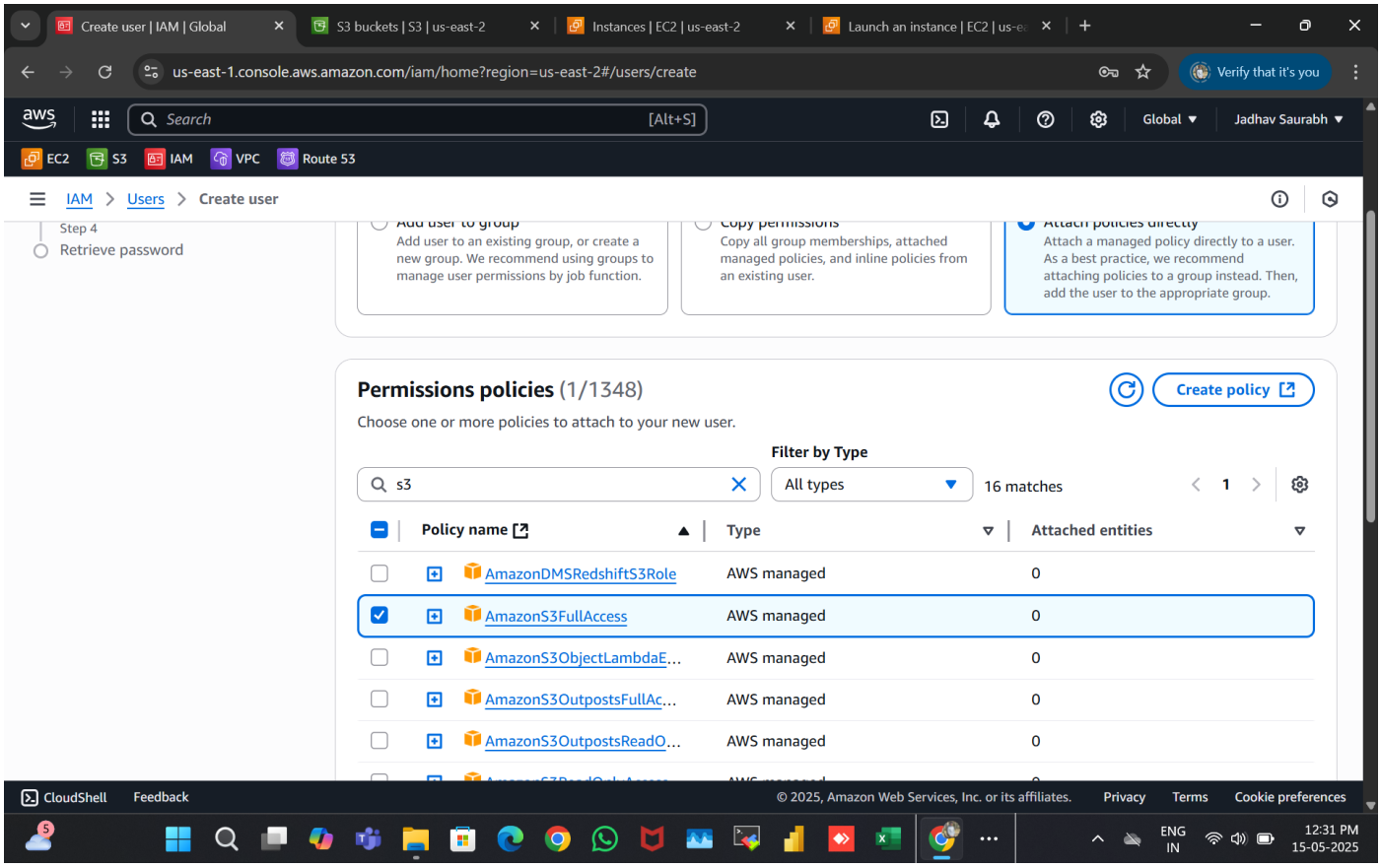
Purpose of IAM in the Project:

- Manage User Permissions: Control who can access AWS resources like S3 bucket and EC2.
- Secure Access: Provide credentials (Access Keys) to your Flask app so it can securely upload/download files.
- Least Privilege: Grant only the necessary permissions to the app user (my-flask-app-user) for security.
- Enable Signed URLs: Help generate temporary, secure URLs for file access, protecting files from public exposure.
- Track and Audit: Allow monitoring and auditing of who accessed what resources in AWS.

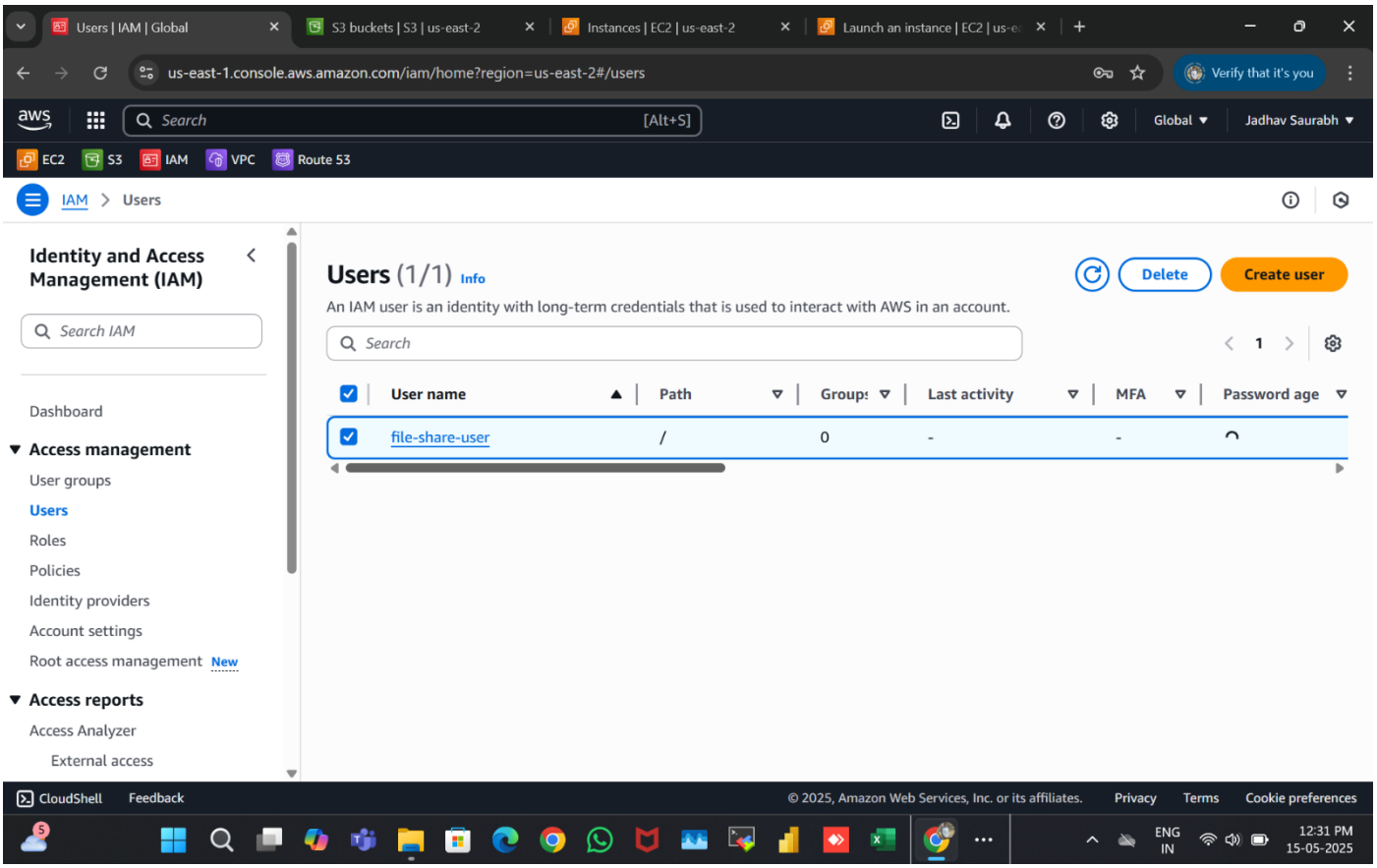
Create IAM User or Role

1. Go to **IAM - Users - Add User**
2. Name: file-share-user
3. Access type: Programmatic access ✅
4. Attach permissions:
    - AmazonS3FullAccess *(or custom policy for bucket)*
5. Save **Access Key ID** and **Secret Access Key**

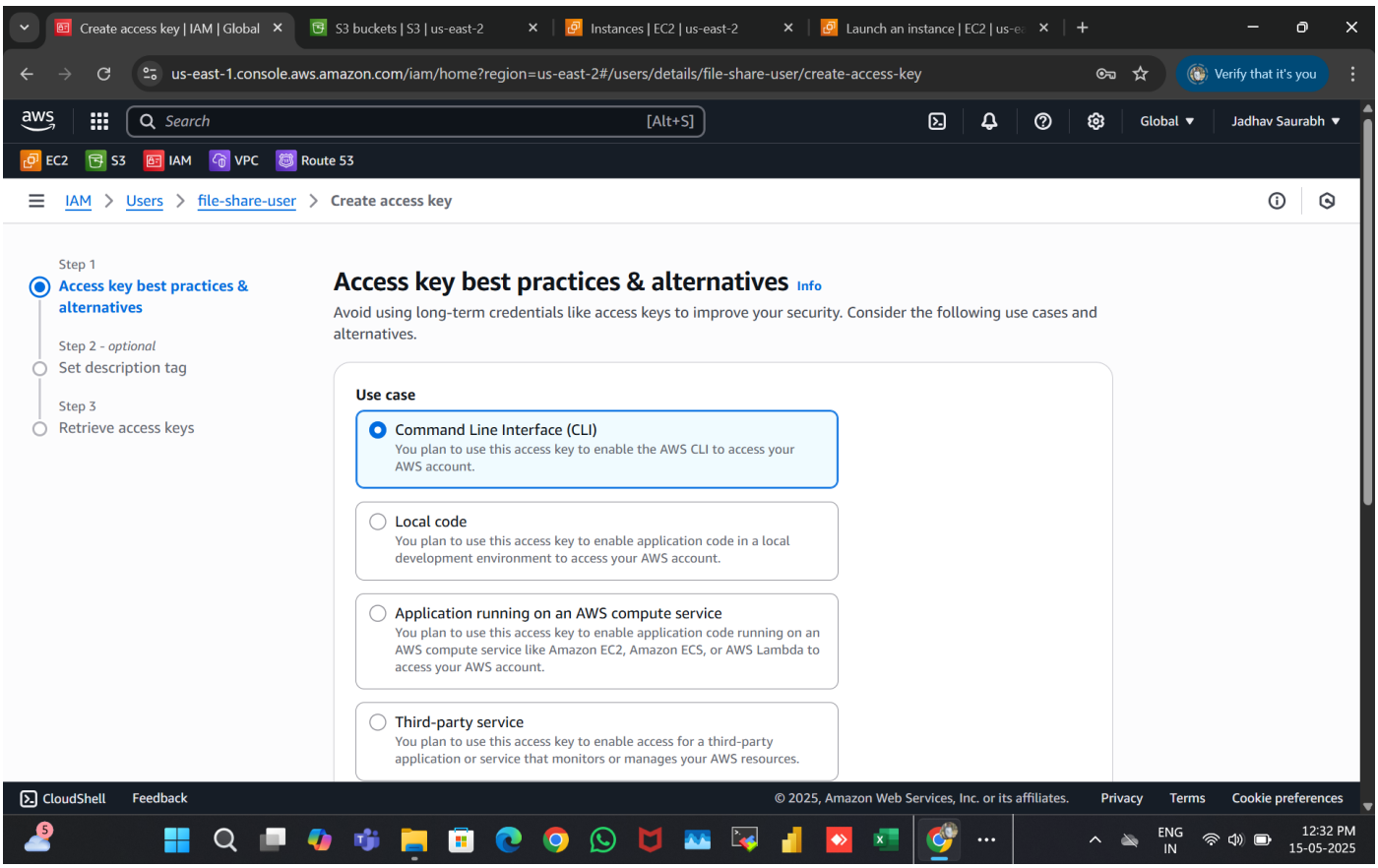




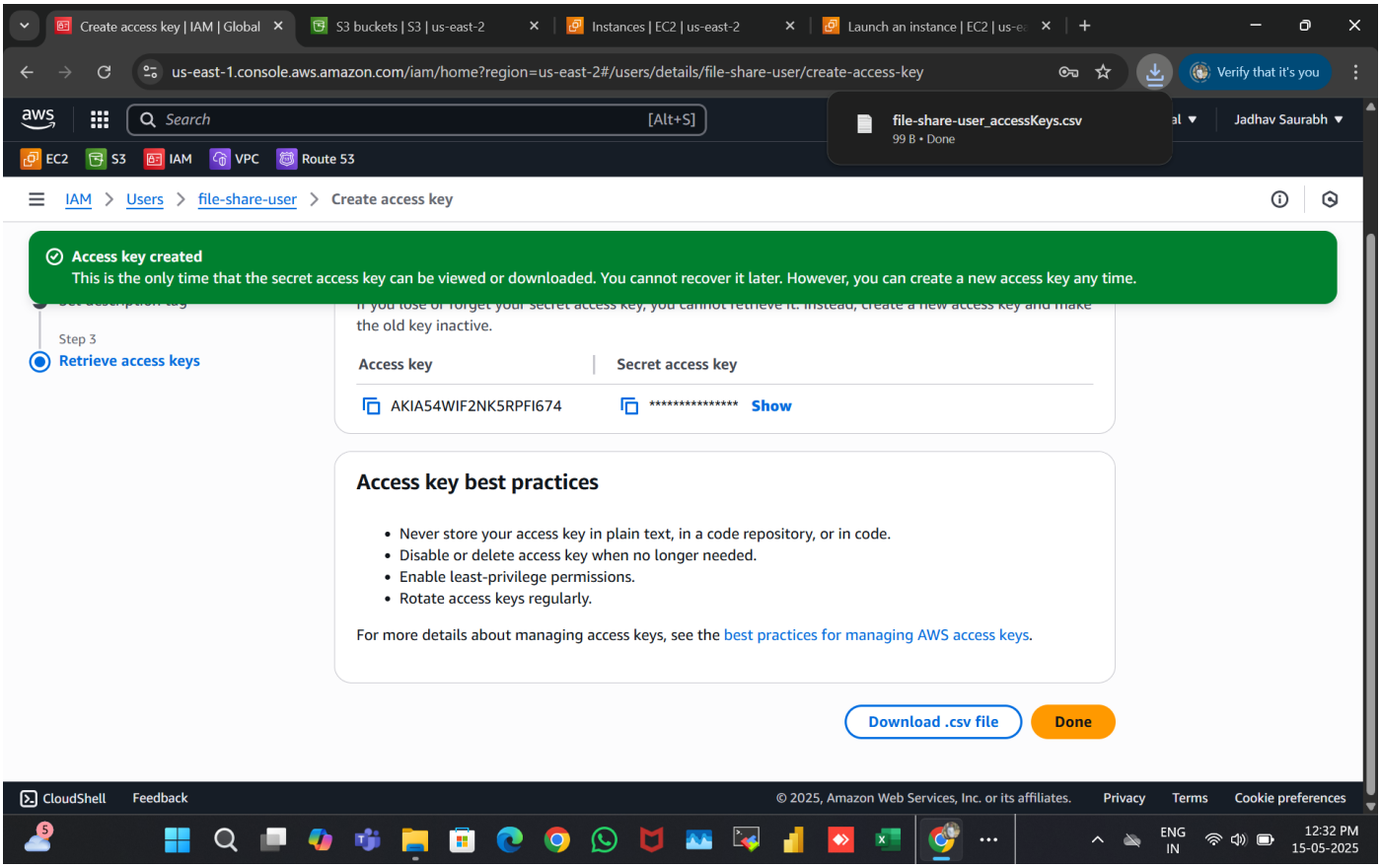
User Created.



Create a Access key .

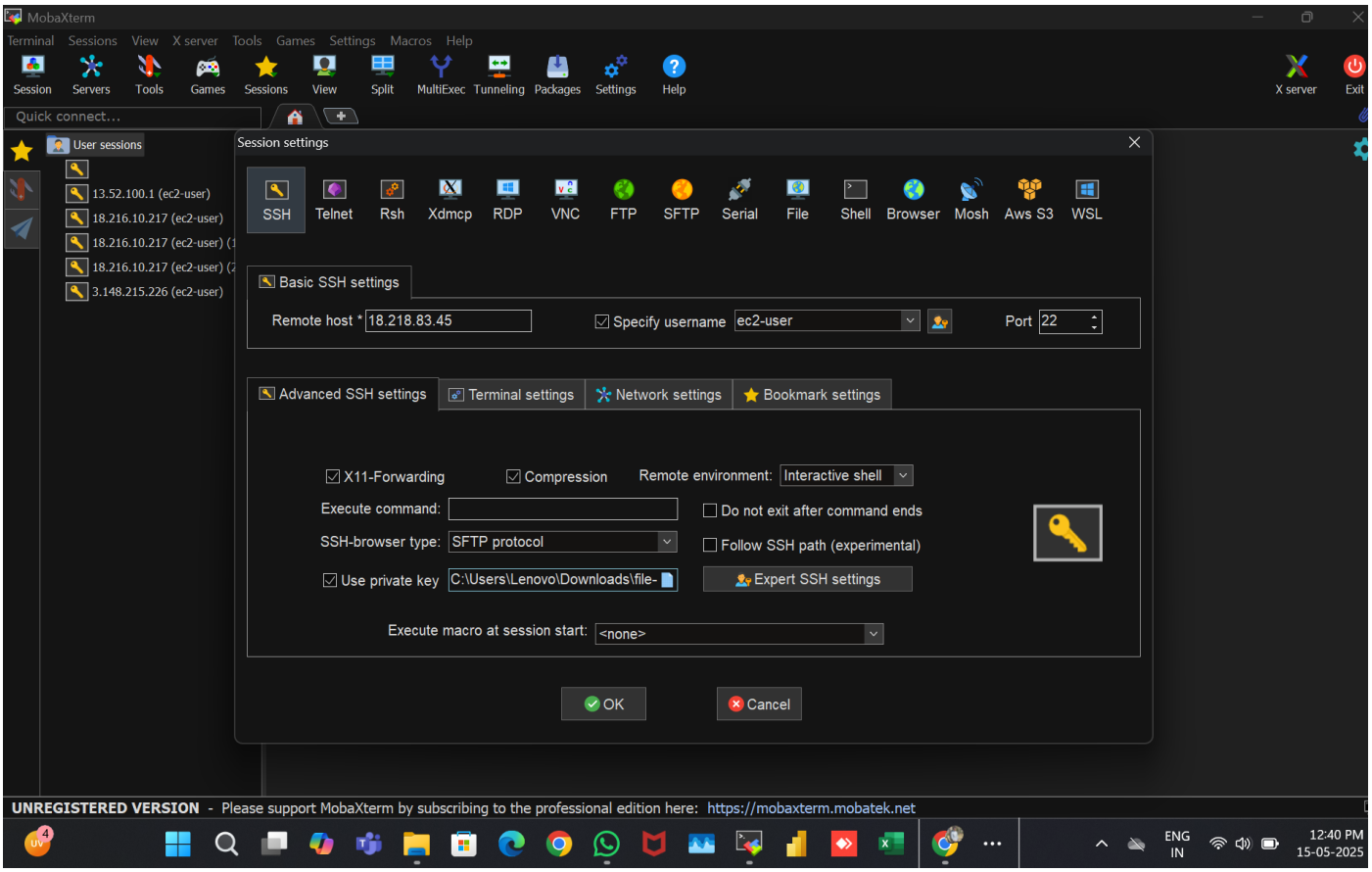


Download .CSV file.



SSH into an EC2 Instance on MobaXterm (if u have any another application then u can used.)

- **Public IP** or **Public DNS** of your EC2 instance
- **Key pair (.pem) file** that you downloaded while launching the instance



Installation & Dependencies

This section explains how to install and set up the environment required to run the Cloud-Based File Sharing System on an Amazon EC2 (Amazon Linux 2) instance.

Update the System

sudo yum update -y

Install Python 3, pip, and Git

sudo yum install python3 git ngnix -y

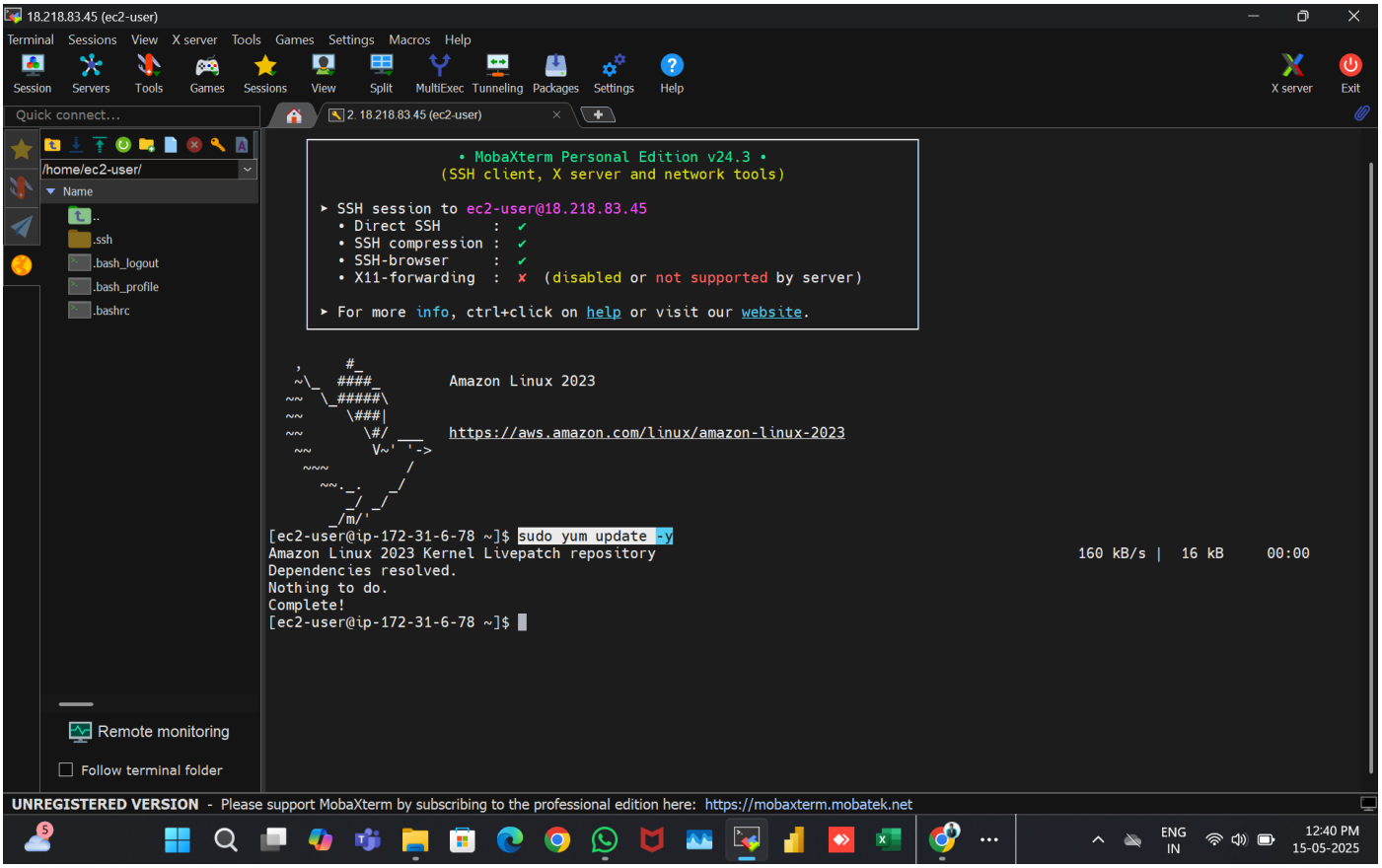
sudo yum install python3 -y

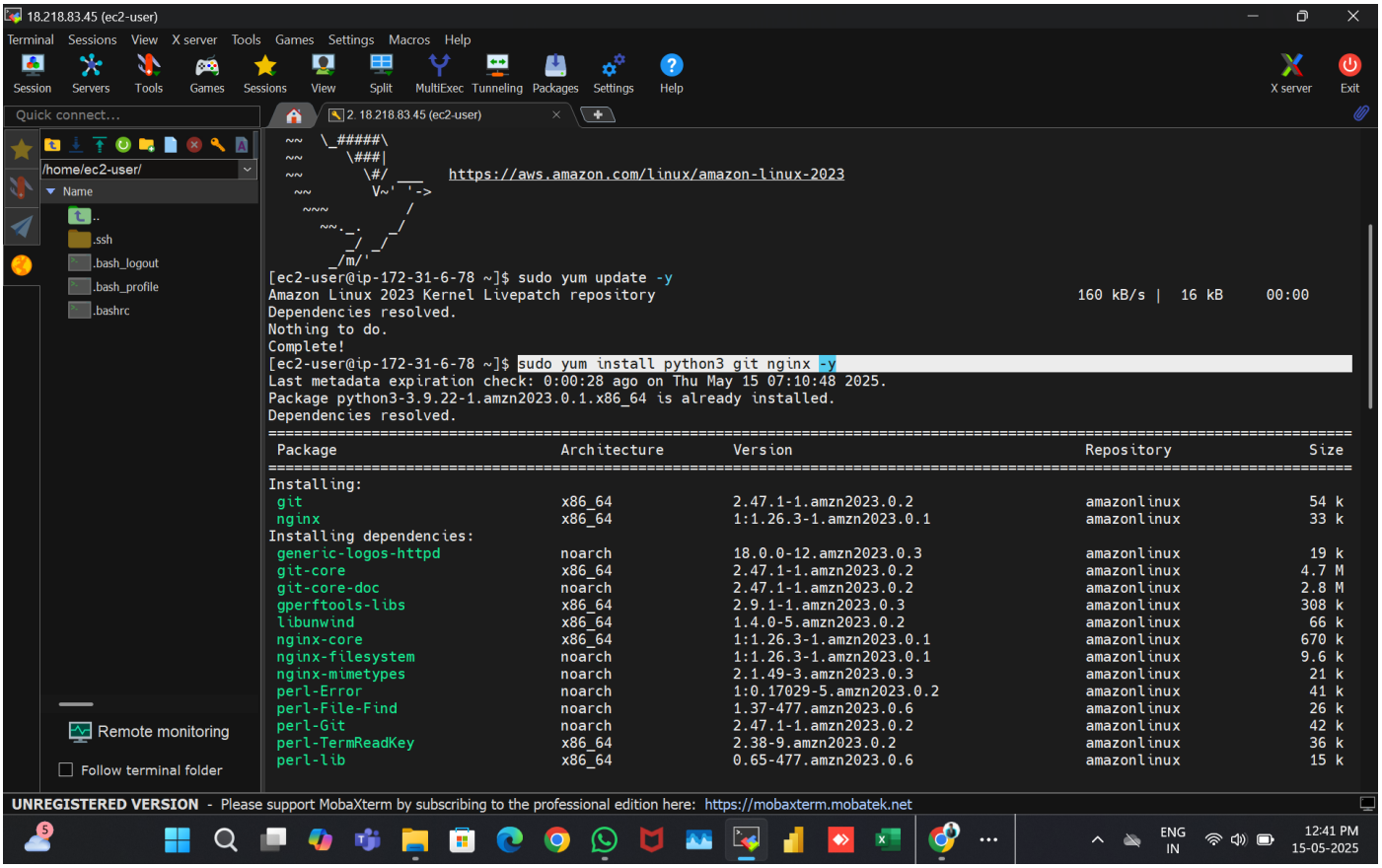
sudo python3 -m ensurepip

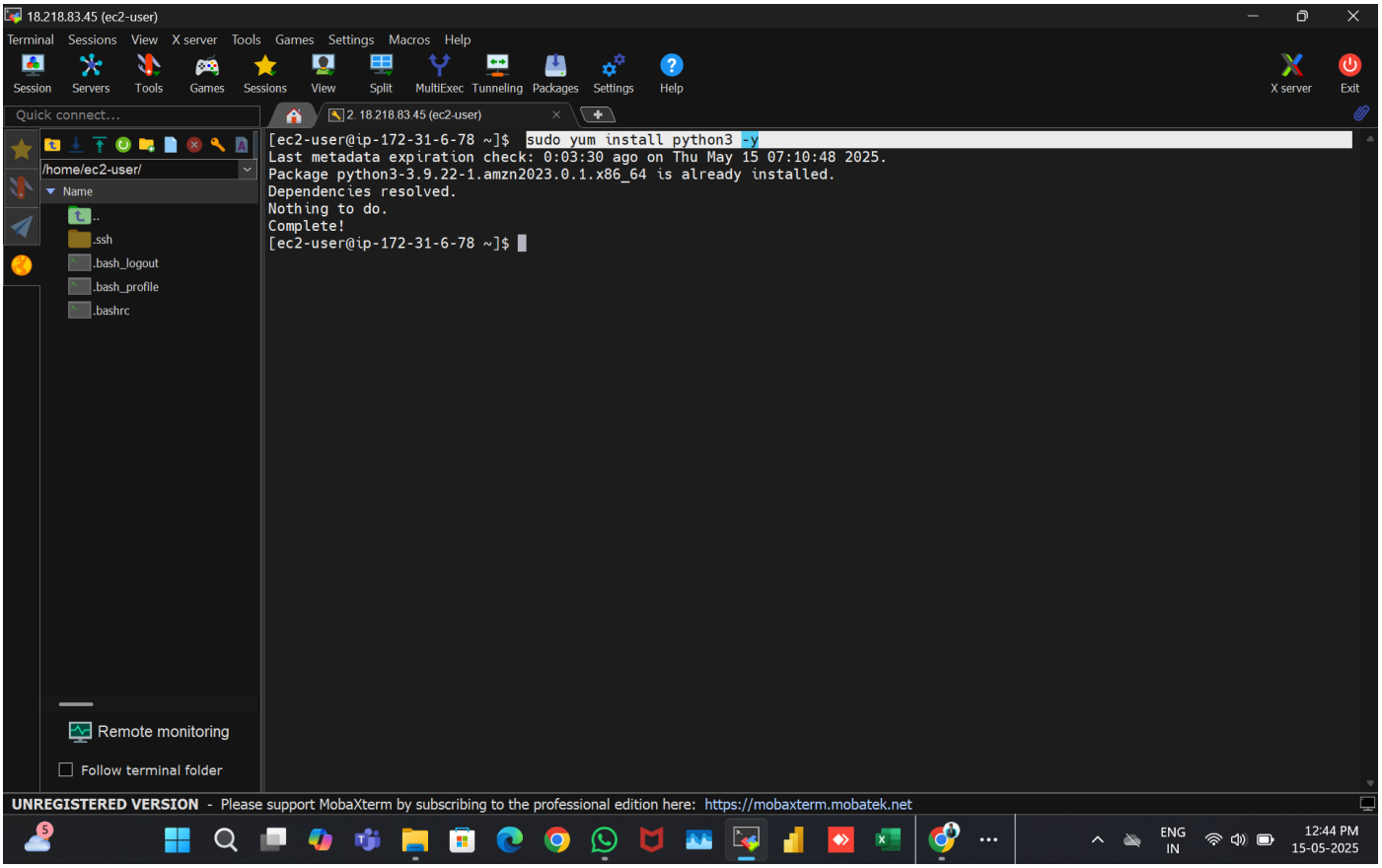
Verify installation :

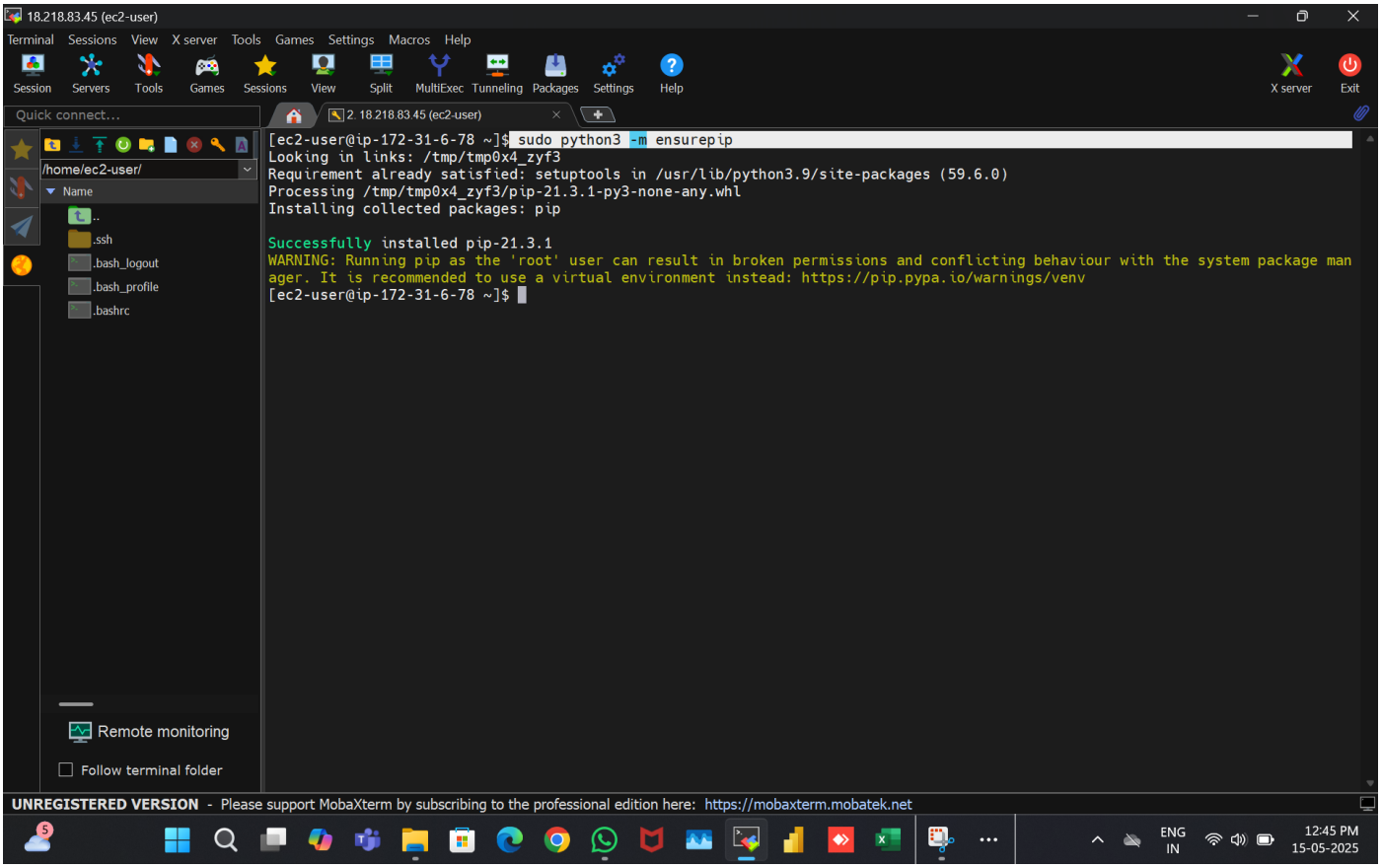
python3 --version

pip3 --version


18.218.83.45 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. 18.218.83.45 (ec2-user)
/home/ec2-user/
Name
..
.ssh
.bash_logout
.bash_profile
.bashrc
Remote monitoring
Follow terminal folder
• MobaXterm Personal Edition v24.3 •
(SSH client, X server and network tools)
▸ SSH session to ec2-user@18.218.83.45
• Direct SSH : ✓
• SSH compression : ✓
• SSH-browser : ✓
• X11-forwarding : ✗ (disabled or not supported by server)
▸ For more info, ctrl+click on help or visit our website.
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-6-78 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository 160 kB/s | 16 kB 00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-6-78 ~]$
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
ENG IN 12:40 PM 15-05-2025


18.218.83.45 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. 18.218.83.45 (ec2-user)
/home/ec2-user/
Name
..
.ssh
.bash_logout
.bash_profile
.bashrc
Remote monitoring
Follow terminal folder
https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-6-78 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository 160 kB/s | 16 kB 00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-6-78 ~]$ sudo yum install python3 git nginx -y
Last metadata expiration check: 0:00:28 ago on Thu May 15 07:10:48 2025.
Package python3-3.9.22-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Package Architecture Version Repository Size
Installing:
git x86_64 2.47.1-1.amzn2023.0.2 amazonlinux 54 k
nginx x86_64 1:1.26.3-1.amzn2023.0.1 amazonlinux 33 k
Installing dependencies:
generic-logos-httpd noarch 18.0.0-12.amzn2023.0.3 amazonlinux 19 k
git-core x86_64 2.47.1-1.amzn2023.0.2 amazonlinux 4.7 M
git-core-doc noarch 2.47.1-1.amzn2023.0.2 amazonlinux 2.8 M
gperftools-libs x86_64 2.9.1-1.amzn2023.0.3 amazonlinux 308 k
libunwind x86_64 1.4.0-5.amzn2023.0.2 amazonlinux 66 k
nginx-core x86_64 1:1.26.3-1.amzn2023.0.1 amazonlinux 670 k
nginx-filesystem noarch 1:1.26.3-1.amzn2023.0.1 amazonlinux 9.6 k
nginx-mimetypes noarch 2.1.49-3.amzn2023.0.3 amazonlinux 21 k
perl-Error noarch 1:0.17029-5.amzn2023.0.2 amazonlinux 41 k
perl-File-Find noarch 1.37-477.amzn2023.0.6 amazonlinux 26 k
perl-Git noarch 2.47.1-1.amzn2023.0.2 amazonlinux 42 k
perl-TermReadKey x86_64 2.38-9.amzn2023.0.2 amazonlinux 36 k
perl-lib x86_64 0.65-477.amzn2023.0.6 amazonlinux 15 k
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
ENG IN 12:41 PM 15-05-2025

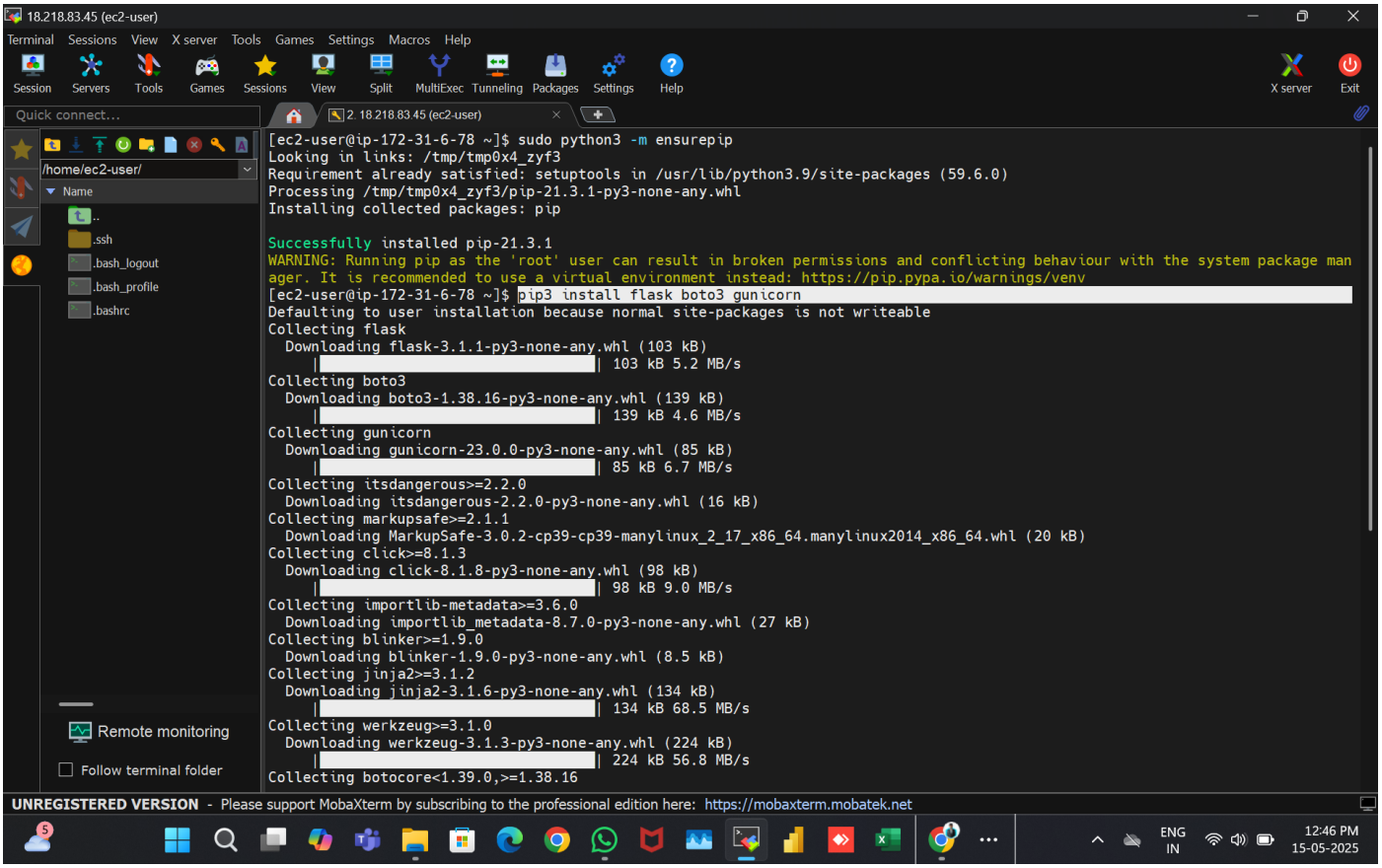

18.218.83.45 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. 18.218.83.45 (ec2-user)
/home/ec2-user/
Name
..
.ssh
.bash_logout
.bash_profile
.bashrc
Remote monitoring
Follow terminal folder
[ec2-user@ip-172-31-6-78 ~]$ sudo yum install python3 -y
Last metadata expiration check: 0:03:30 ago on Thu May 15 07:10:48 2025.
Package python3-3.9.22-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-6-78 ~]$
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
ENG IN
12:44 PM
15-05-2025


18.218.83.45 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. 18.218.83.45 (ec2-user)
/home/ec2-user/
Name
..
.ssh
.bash_logout
.bash_profile
.bashrc
Remote monitoring
Follow terminal folder
[ec2-user@ip-172-31-6-78 ~]$ sudo python3 -m ensurepip
Looking in links: /tmp/tmp0x4_zyf3
Requirement already satisfied: setuptools in /usr/lib/python3.9/site-packages (59.6.0)
Processing /tmp/tmp0x4_zyf3/pip-21.3.1-py3-none-any.whl
Installing collected packages: pip
Successfully installed pip-21.3.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-172-31-6-78 ~]$
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
ENG IN
12:45 PM
15-05-2025

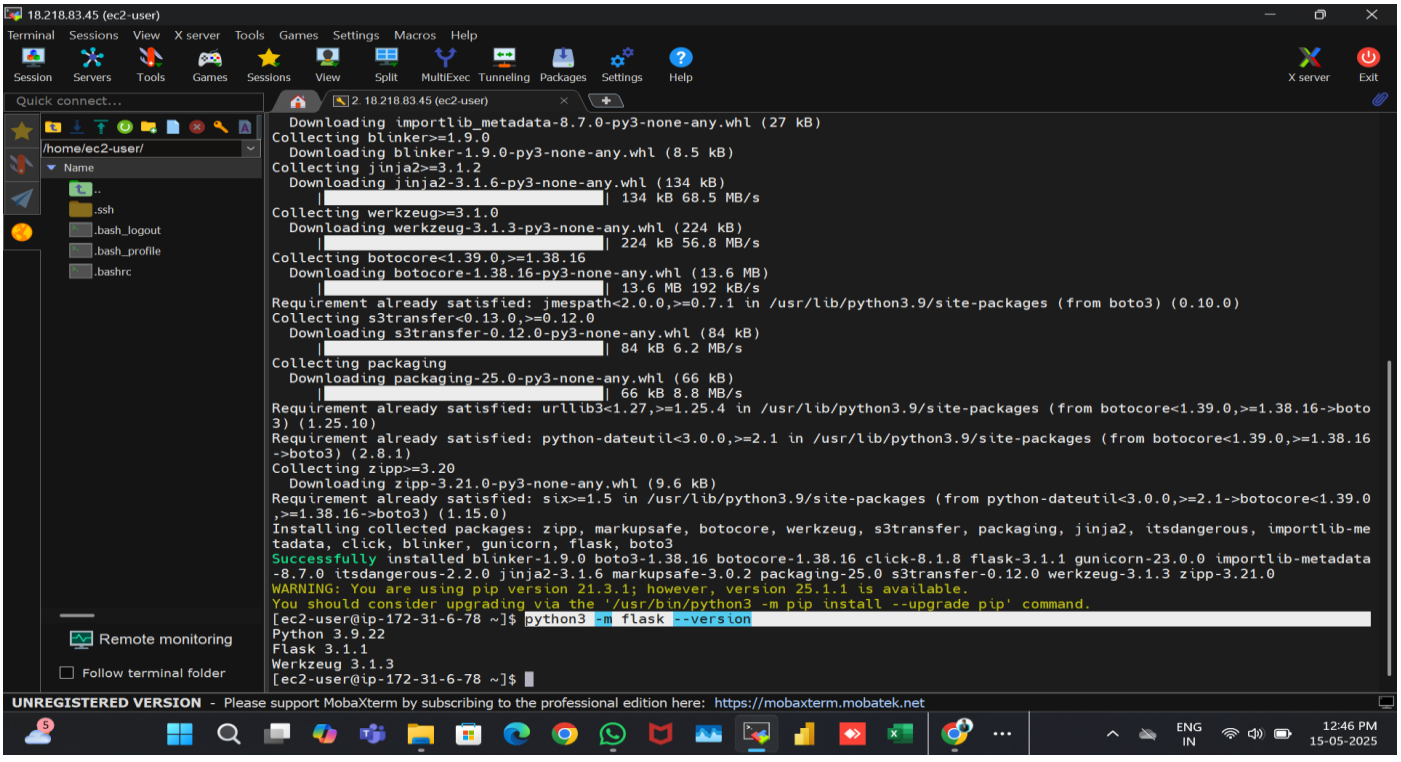
Install Python Dependencies

To ensure your Flask web application runs smoothly, you need to install essential Python packages that support your file-sharing system.

pip install flask boto3 gunicorn

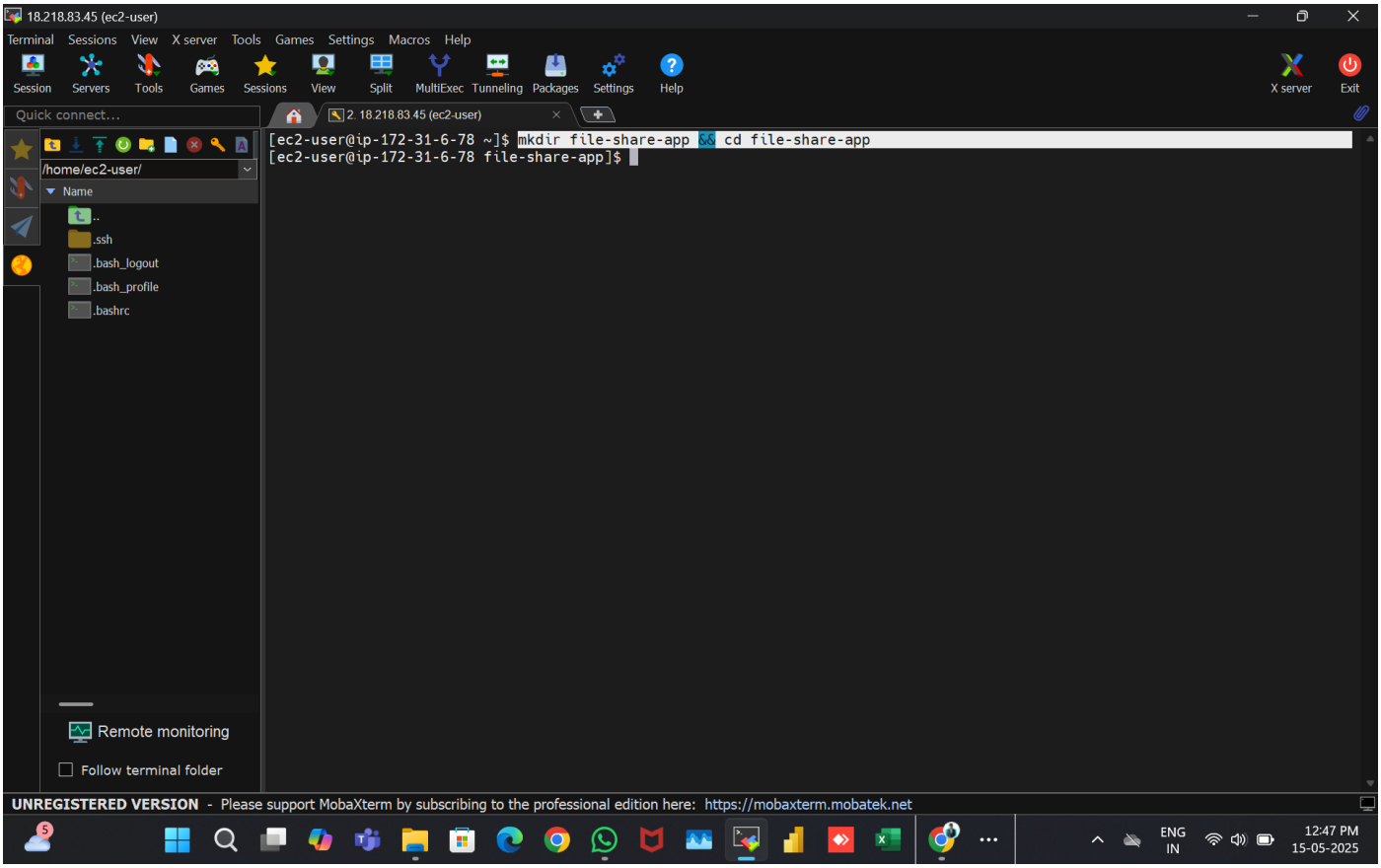


Check flask –version : python3 -m flask --version.



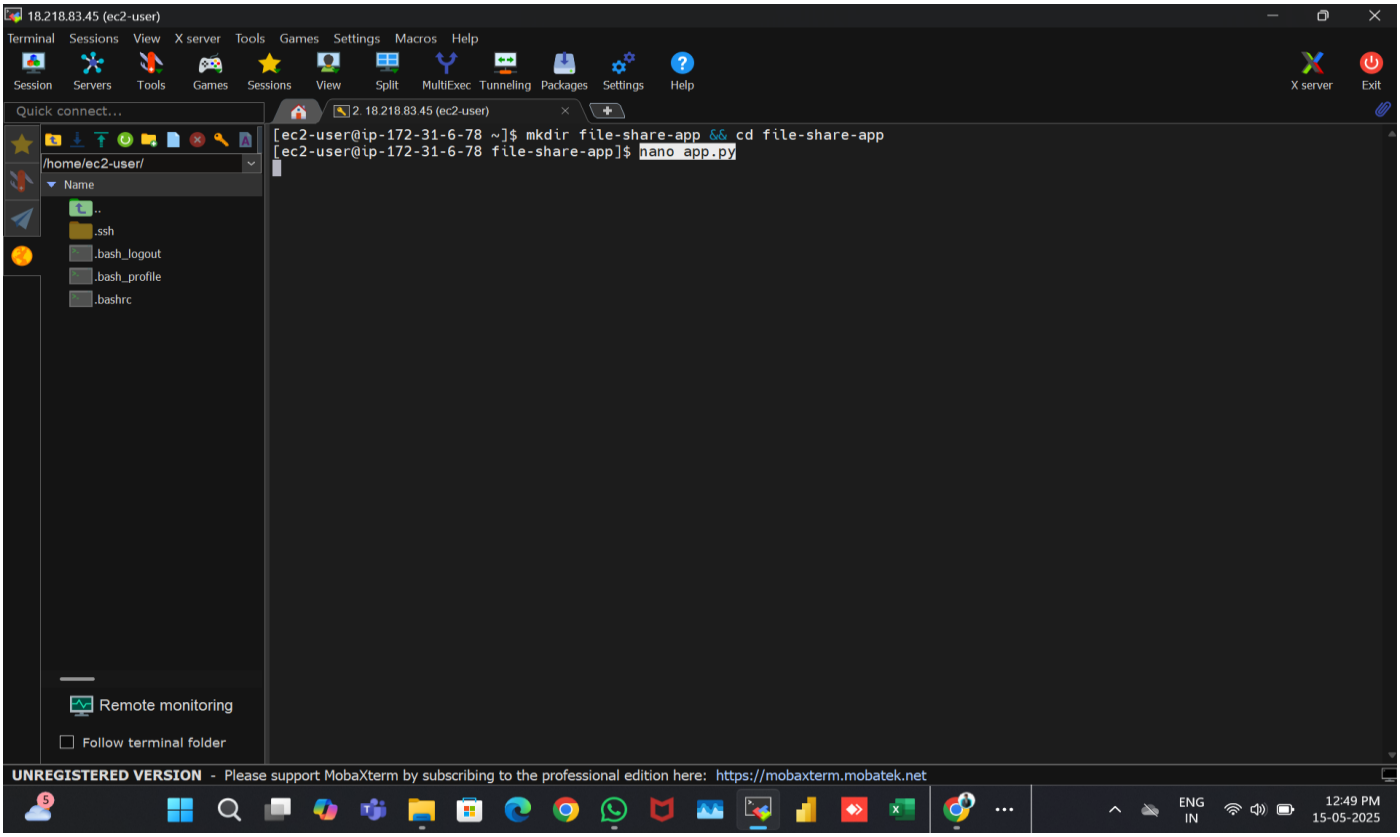
Create Project Directory and Navigate

mkdir file-share-app && cd file-share-app

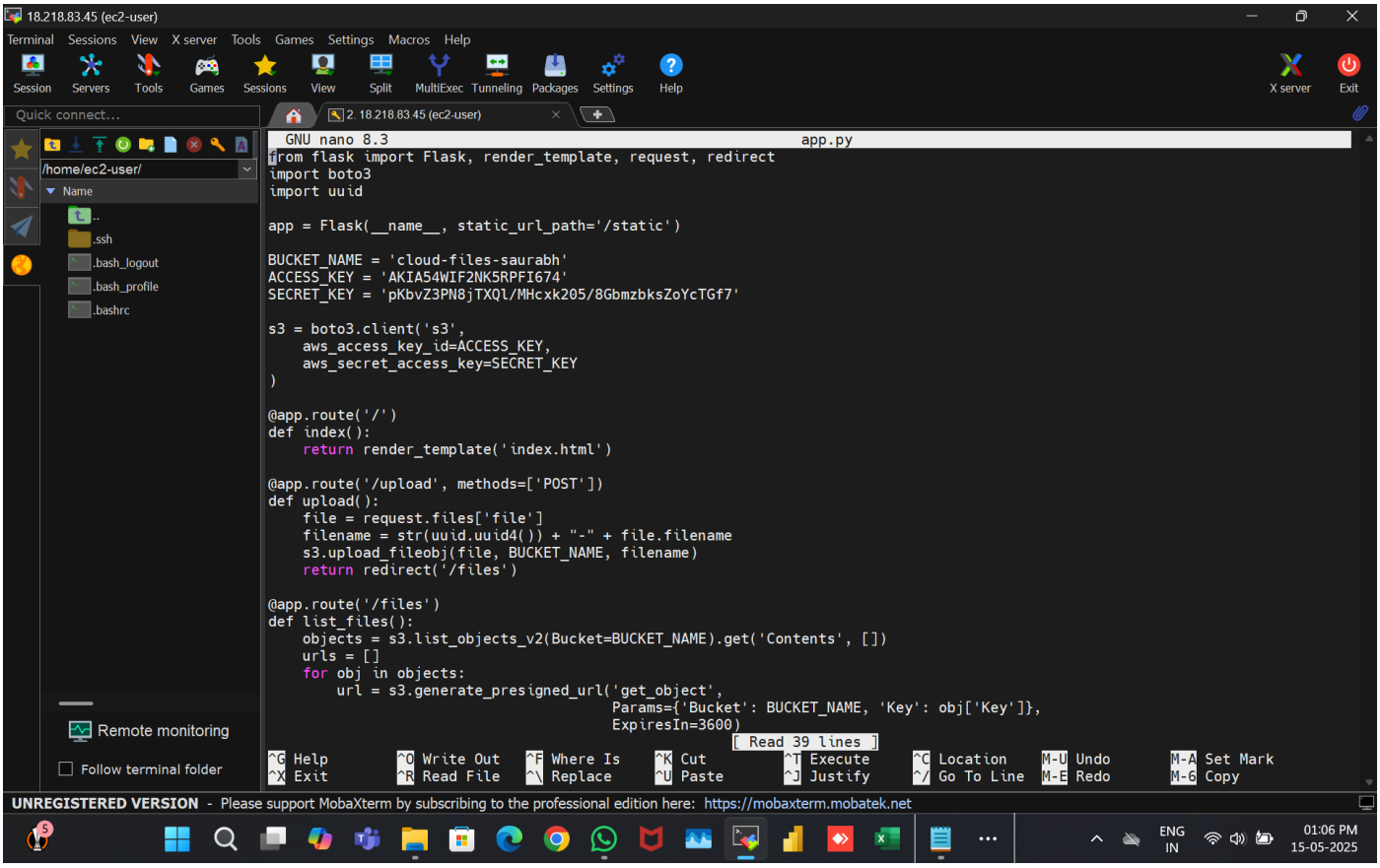


Now, create your main Flask application file and add following code :

nano app.py



```
from flask import Flask, render_template, request, redirect, url_for
import boto3
import os
app = Flask(__name__)
BUCKET_NAME = 'file-sharing-server'
s3 = boto3.client(
    's3',
    aws_access_key_id='AKIA54WIF2NKTVR4EX7G',
    aws_secret_access_key='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'
)
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        file = request.files['file']
        if file:
            s3.upload_fileobj(file, BUCKET_NAME, file.filename)
            return redirect(url_for('index'))
    objects = s3.list_objects_v2(Bucket=BUCKET_NAME).get('Contents', [])
    return render_template('index.html', files=objects)
@app.route('/download/<filename>')
def download(filename):
    url = s3.generate_presigned_url('get_object',
                                    Params={'Bucket': BUCKET_NAME, 'Key': filename},
                                    ExpiresIn=3600)
    return redirect(url)
if __name__ == '__main__':
    app.run()
```



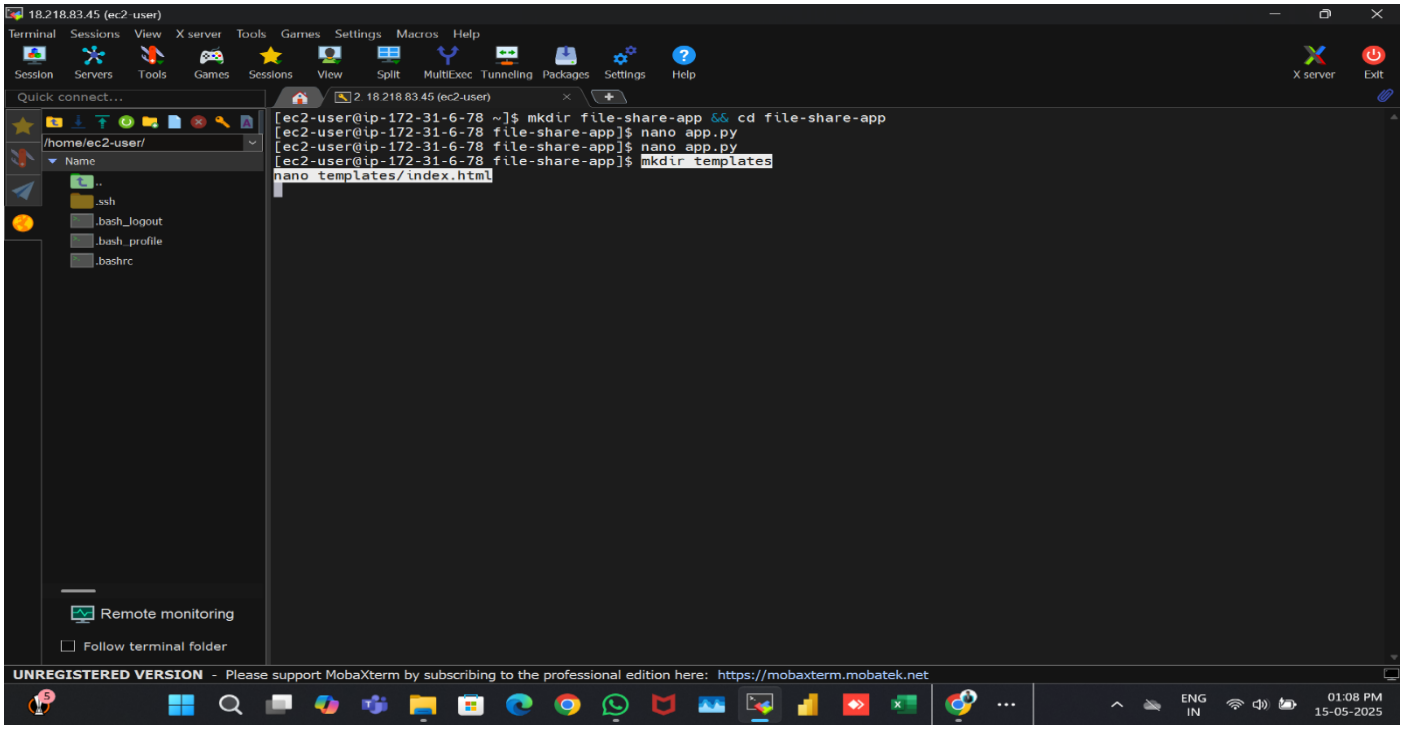
Create the HTML Template for Web Interface and add following code .

Now that you've created your Flask app (app.py), it's time to create the HTML templates used to render the web interface. Flask uses a templates folder by default to look for HTML files, so you need to create that folder and add an index.html file inside it.

mkdir -p templates

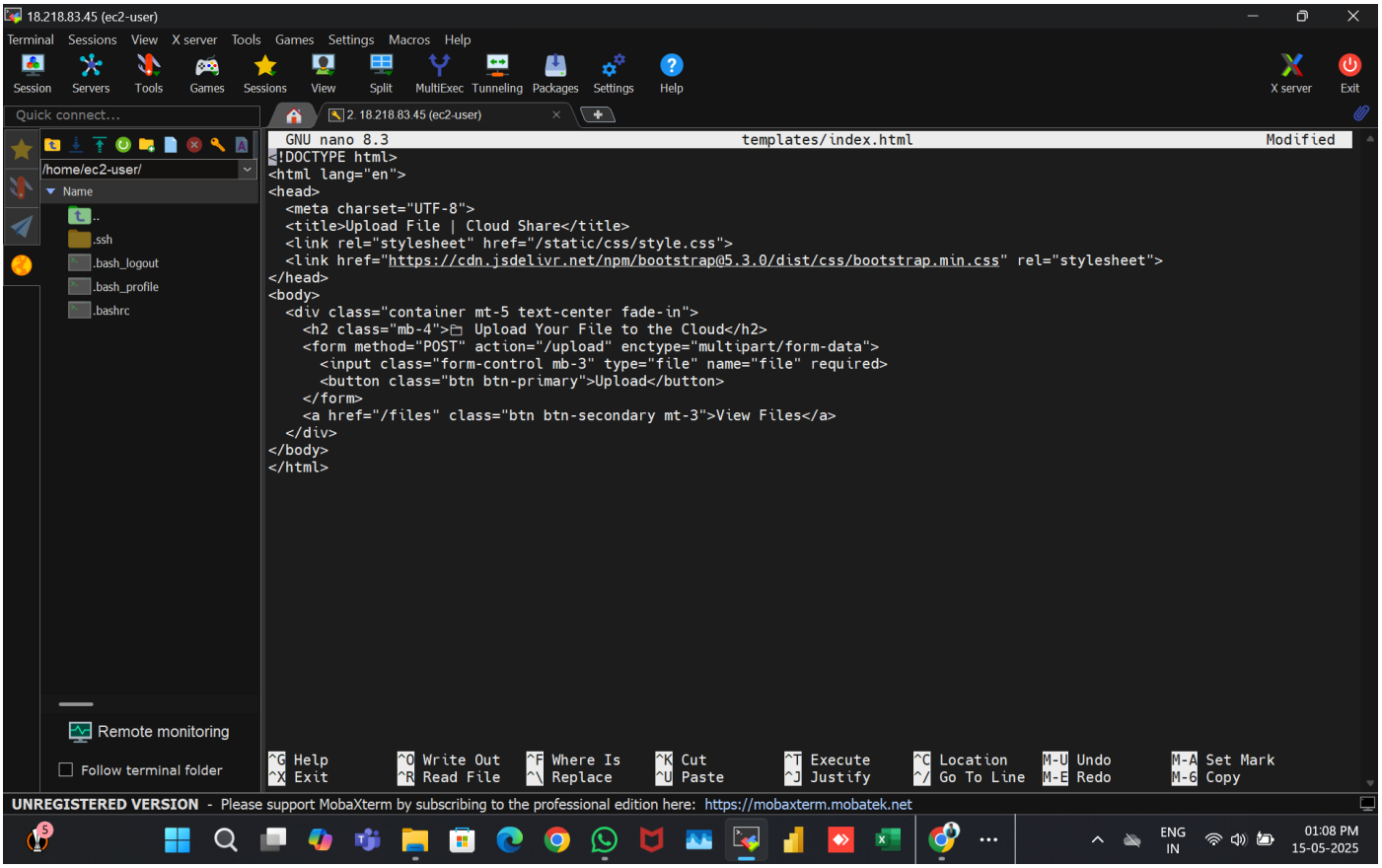
cd templates

nano index.html



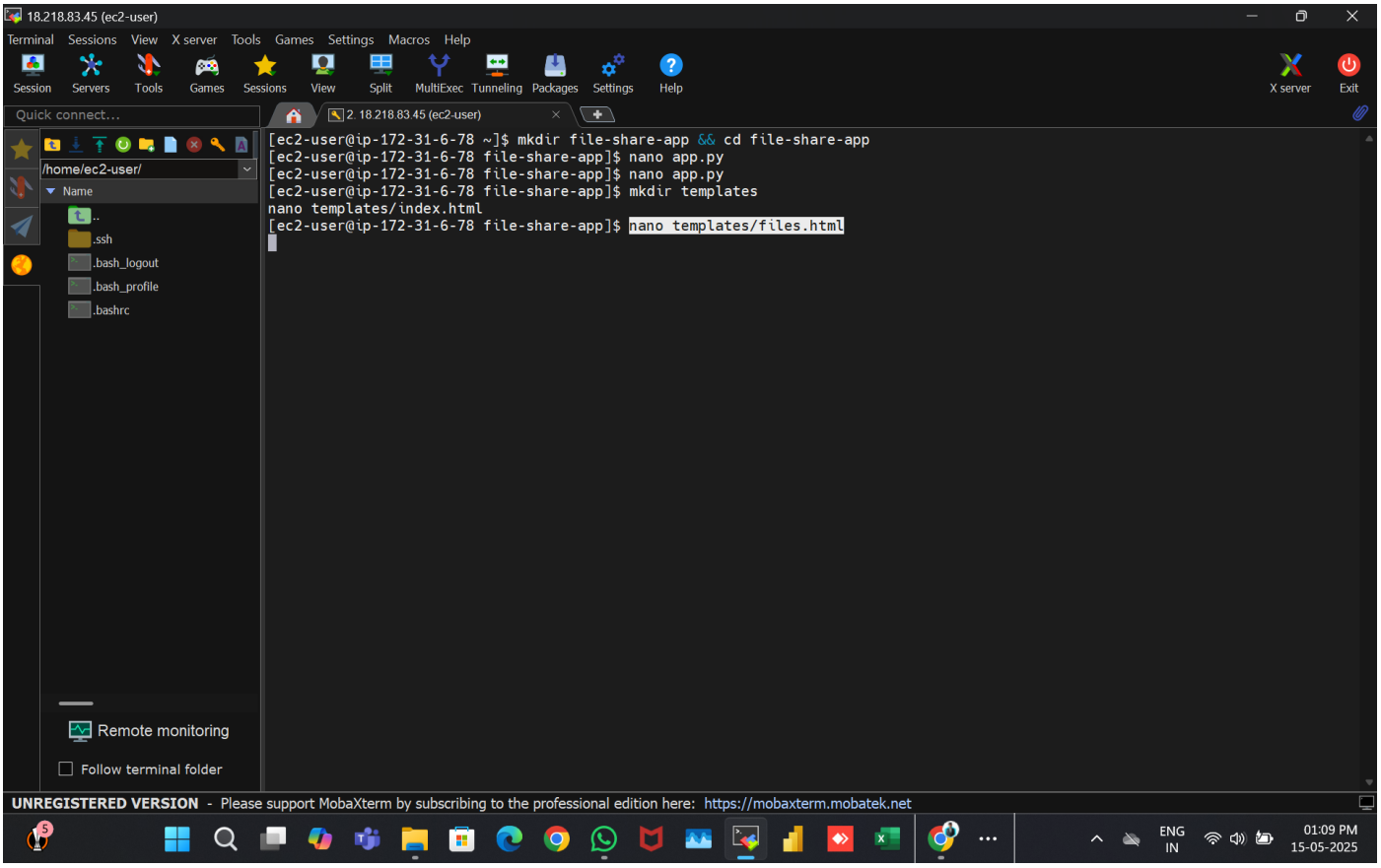
```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Cloud File Sharing</title>

  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
rel="stylesheet">

  <style>

    body {

      font-family: 'Poppins', sans-serif;

      background: linear-gradient(135deg, #2980b9, #6dd5fa);

      color: #fff;

      text-align: center;

      padding-top: 80px;

    }  h1 {

      font-size: 2.5rem;

      animation: fadeInDown 1s ease;

    }    form {

      background: rgba(255, 255, 255, 0.1);

      padding: 30px;

      margin: 40px auto;

      width: 40%;

      border-radius: 15px;

      box-shadow: 0 4px 15px rgba(0,0,0,0.2);

      animation: fadeInUp 1s ease;

    } input[type="file"] {

      margin-bottom: 20px;
```

```
    }
      input[type="submit"] {
      background-color: #27ae60;
      color: white;
      padding: 12px 25px;
      border: none;
      border-radius: 10px;
      cursor: pointer;
      transition: background-color 0.3s ease;
    }  input[type="submit"]:hover {
      background-color: #2ecc71;
    } @keyframes fadeInDown {
      from { opacity: 0; transform: translateY(-30px); }
      to { opacity: 1; transform: translateY(0); }
    } @keyframes fadeInUp {
      from { opacity: 0; transform: translateY(30px); }
      to { opacity: 1; transform: translateY(0); }
    }  </style>
</head>
<body>
  <h1>Upload File to Cloud</h1>
  <form action="/upload" method="POST" enctype="multipart/form-data">
    <input type="file" name="file" required><br>
    <input type="submit" value="Upload">
  </form>
</body>
</html>
```



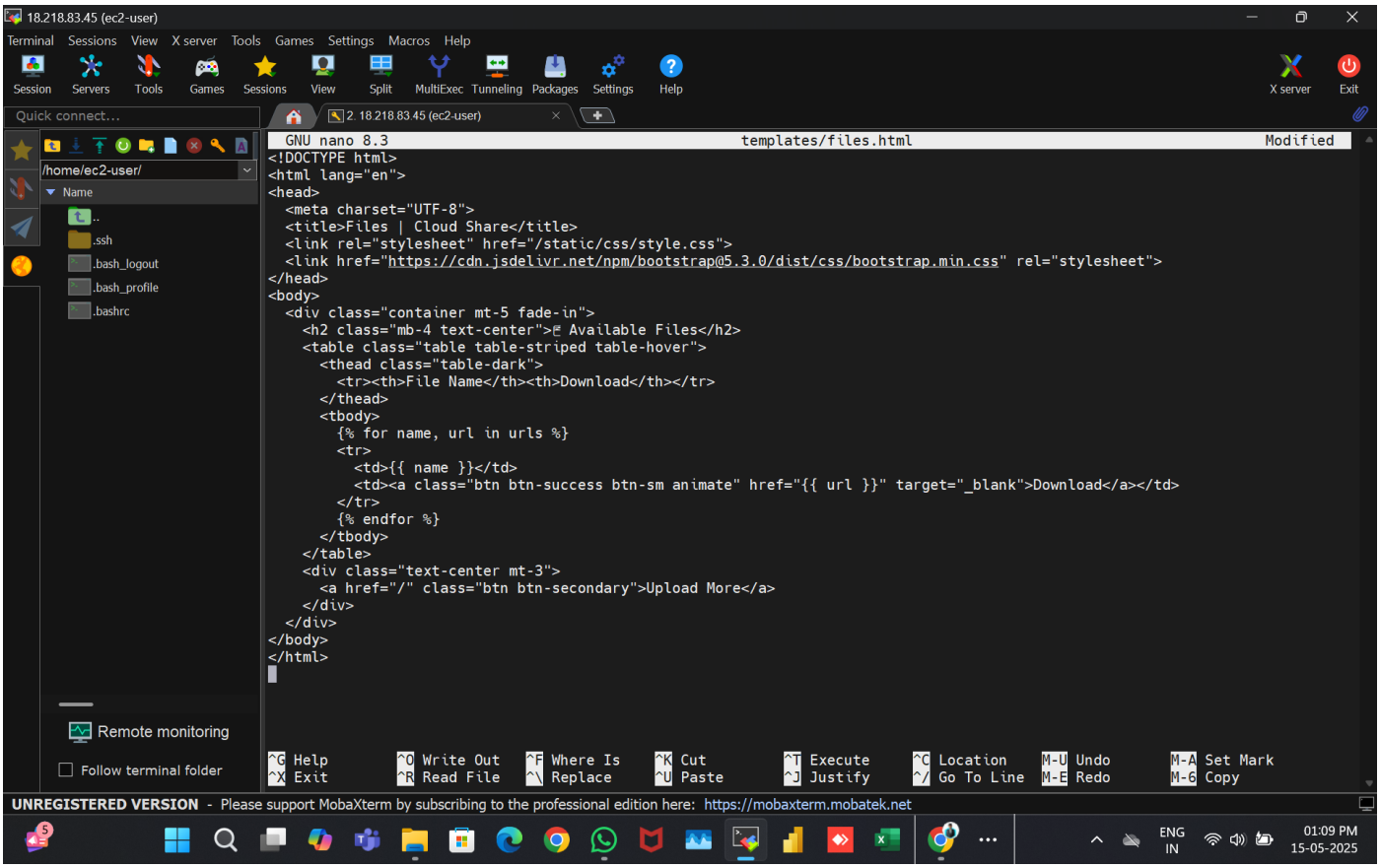
Create or Edit the File and add following code .

nano templates/files.html



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Files | Cloud Share</title>
  <link rel="stylesheet" href="/static/css/style.css">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
  <div class="container mt-5 fade-in">
    <h2 class="mb-4 text-center">📁 Available Files</h2>
    <table class="table table-striped table-hover">
      <thead class="table-dark">
        <tr>
          <th>File Name</th>
          <th>Download</th>
        </tr>
      </thead>
      <tbody>
        {% for name, url in urls %}
        <tr>
          <td>{{ name }}</td>
          <td><a class="btn btn-success btn-sm animate" href="{{ url }}"
target="_blank">Download</a></td>
        </tr>
        {% endfor %}
      </tbody>
```

```
    </table>

    <div class="text-center mt-3">

      <a href="/" class="btn btn-secondary">Upload More</a>

    </div>

  </div>

</body>

</html>
```



create style.css

This will add custom styles (colors, animations, spacing, etc.) to your files.html and index.html pages, enhancing your Cloud File Sharing UI.

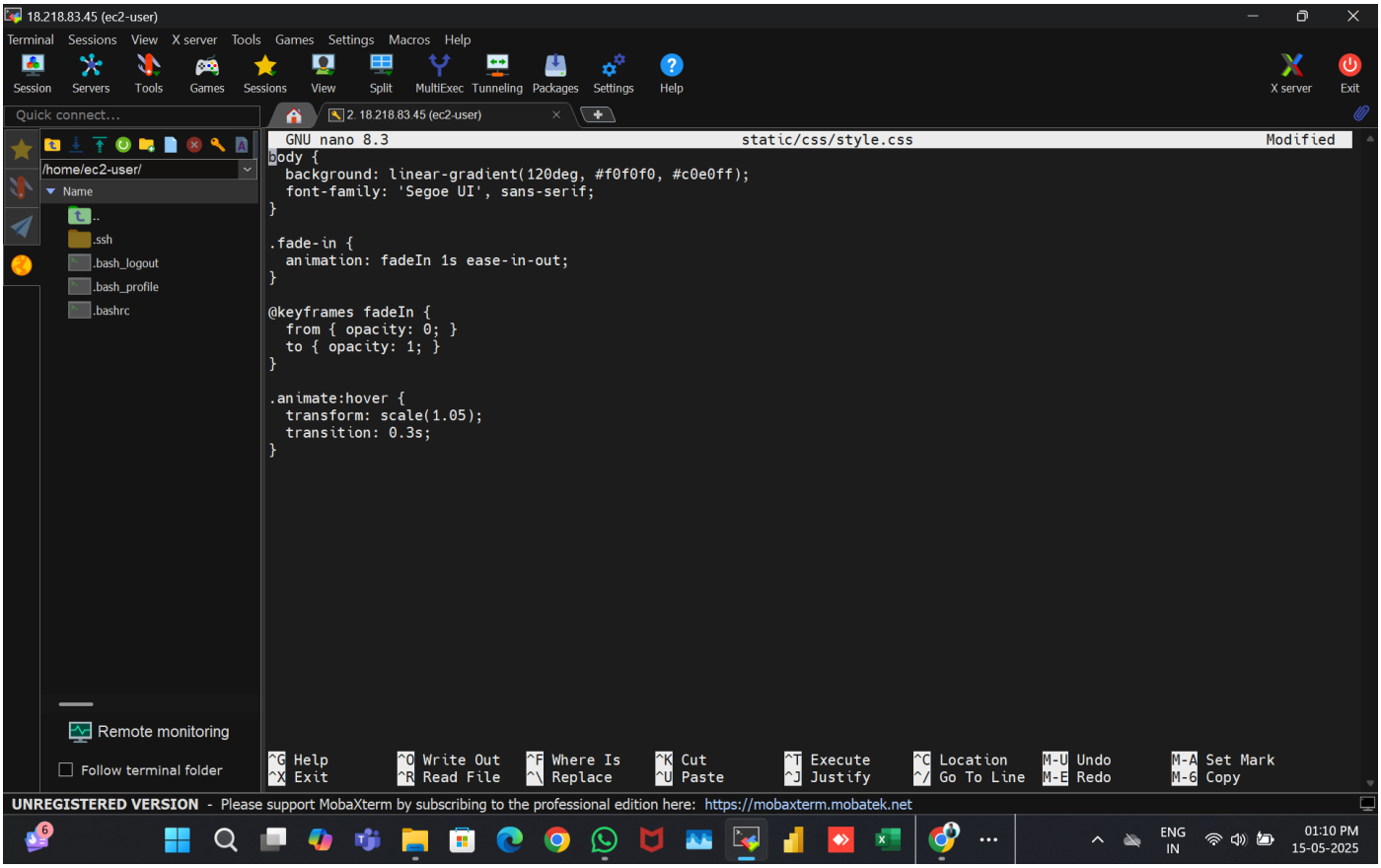
Commands & Code :

mkdir -p static/css

nano static/css/style.css

Add :

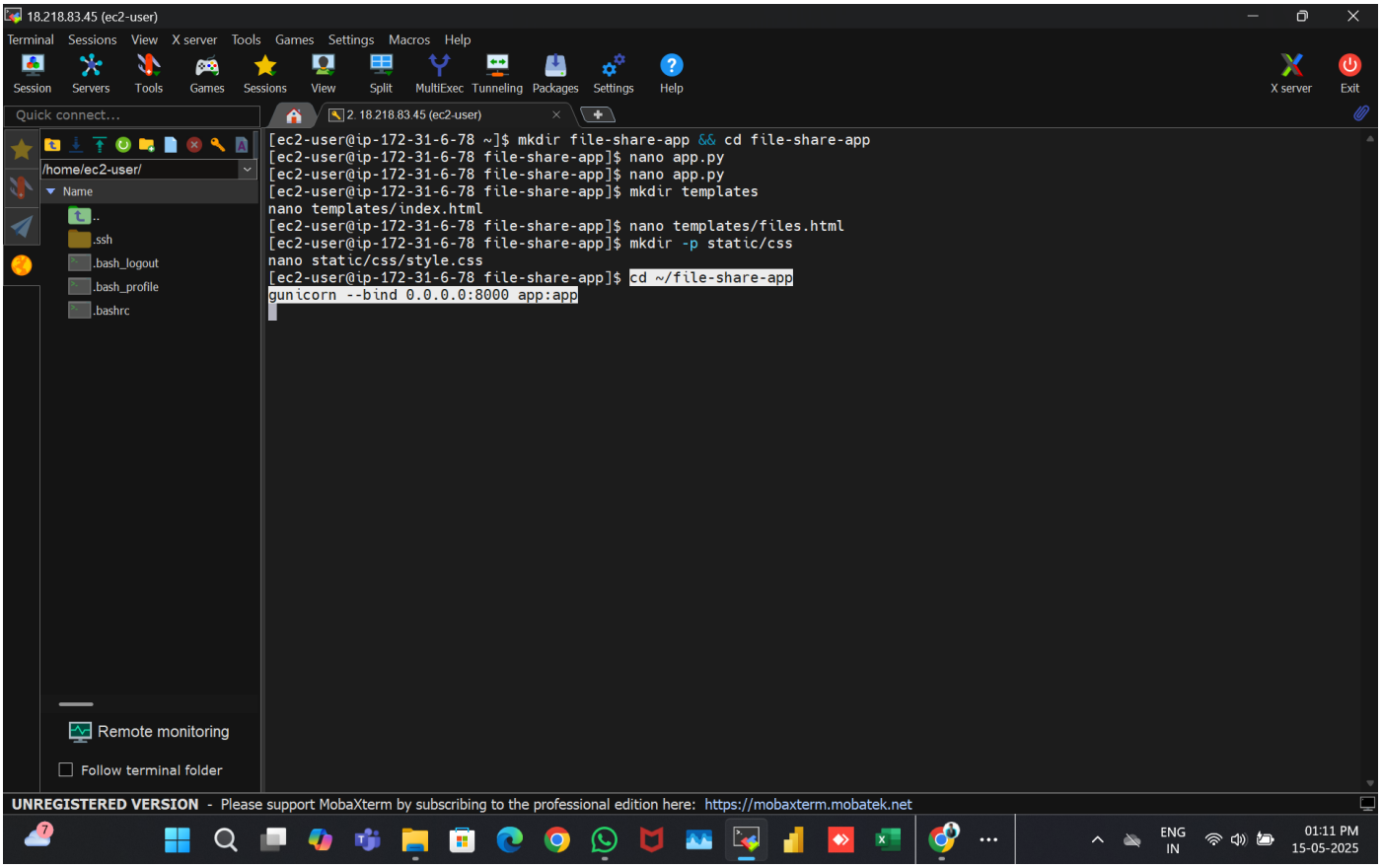
```
body {
    background: linear-gradient(120deg, #f0f0f0, #c0e0ff);
    font-family: 'Segoe UI', sans-serif;
}

.fade-in {
    animation: fadeIn 1s ease-in-out;
}
@keyframes fadeIn {
    from { opacity: 0; }
    to { opacity: 1; }
}
.animate:hover {
    transform: scale(1.05);
    transition: 0.3s;
}
```



Run the following command :

Cd ~/file-share-app

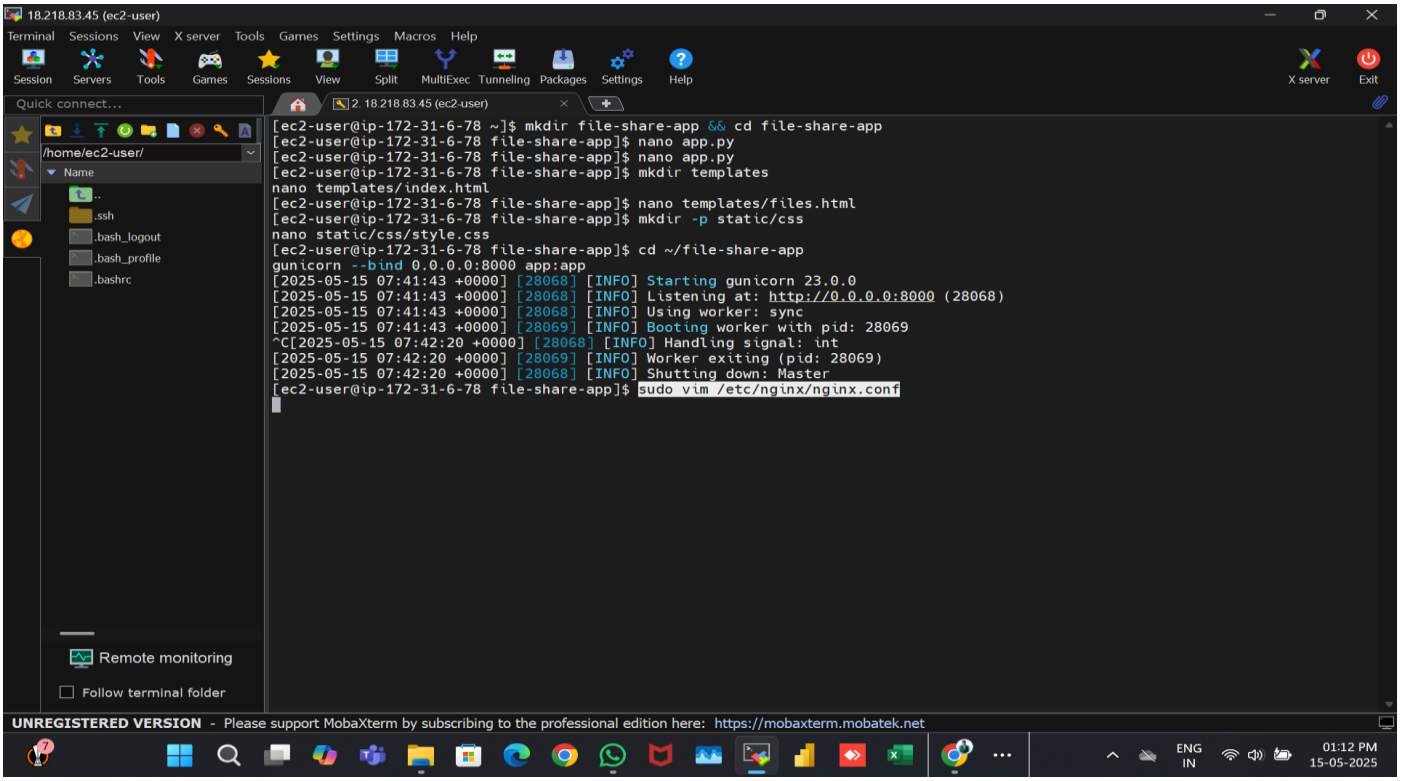
gunicorn --bind 0.0.0.0:8000 app:app



Flask app is now running on port 8000 using Gunicorn, accessible via :

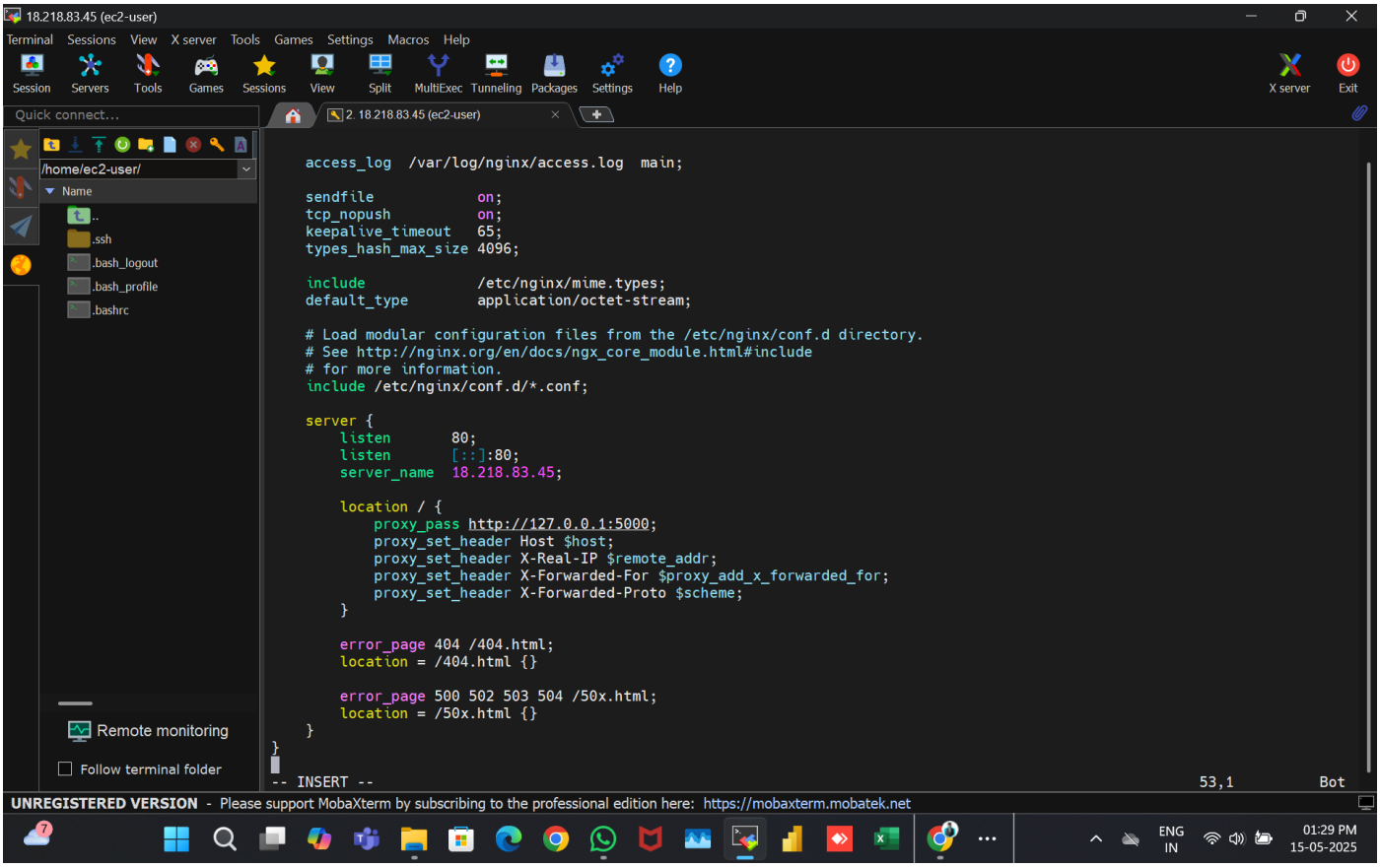
open the Nginx config file with vim using sudo :

sudo vim /etc/nginx/conf.d/file-share-app.conf



Scroll to the end of the file (after the http { ... } block) or locate the http { block and add the server block inside it like this :

```
server {
    listen 80;
    server_name your_domain_or_IP;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

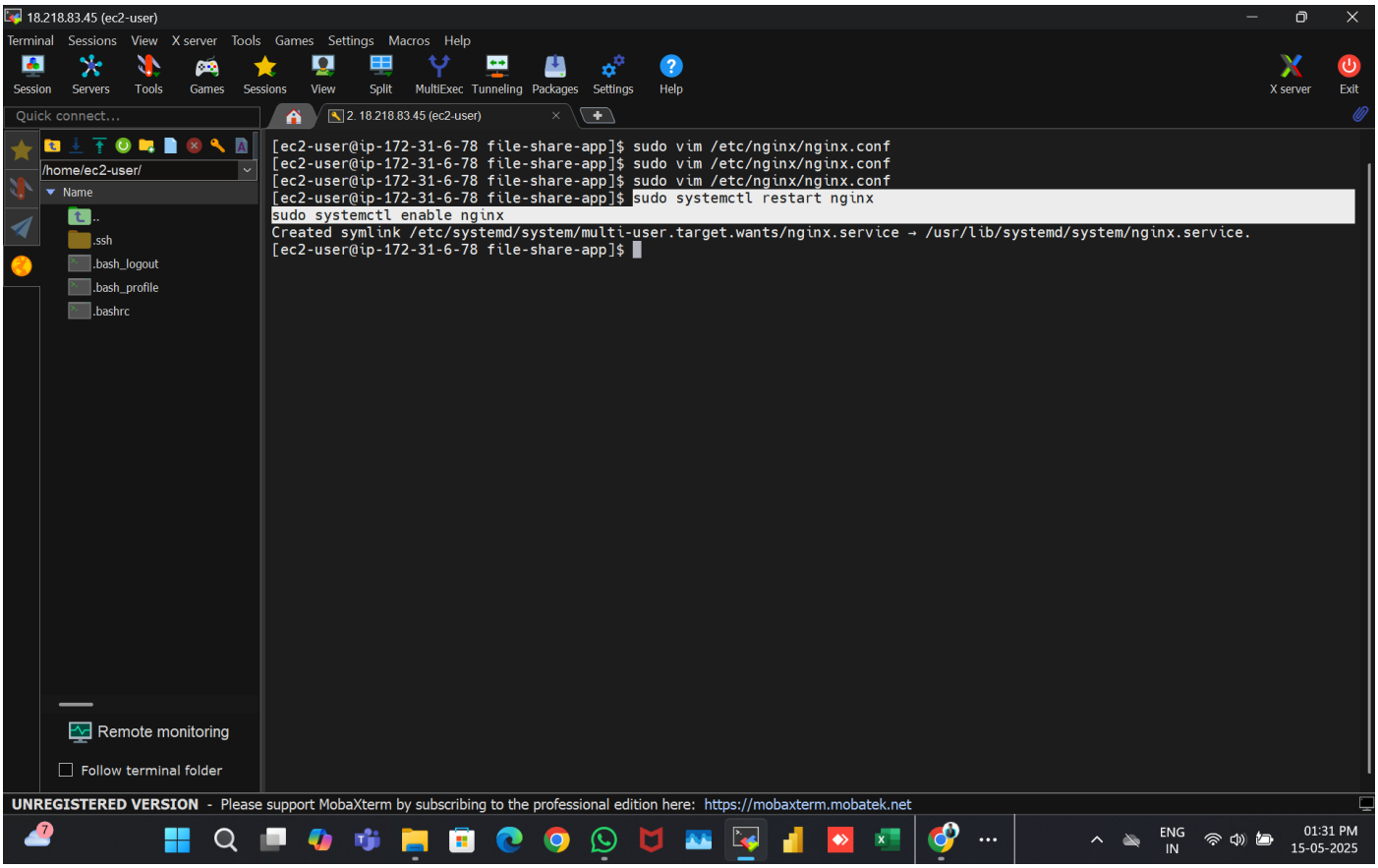


Purpose/Action:

- This command **enables** the Nginx service to **automatically start on boot**.
- It creates a symbolic link from the Nginx service file to the system's multi-user.target.wants directory.

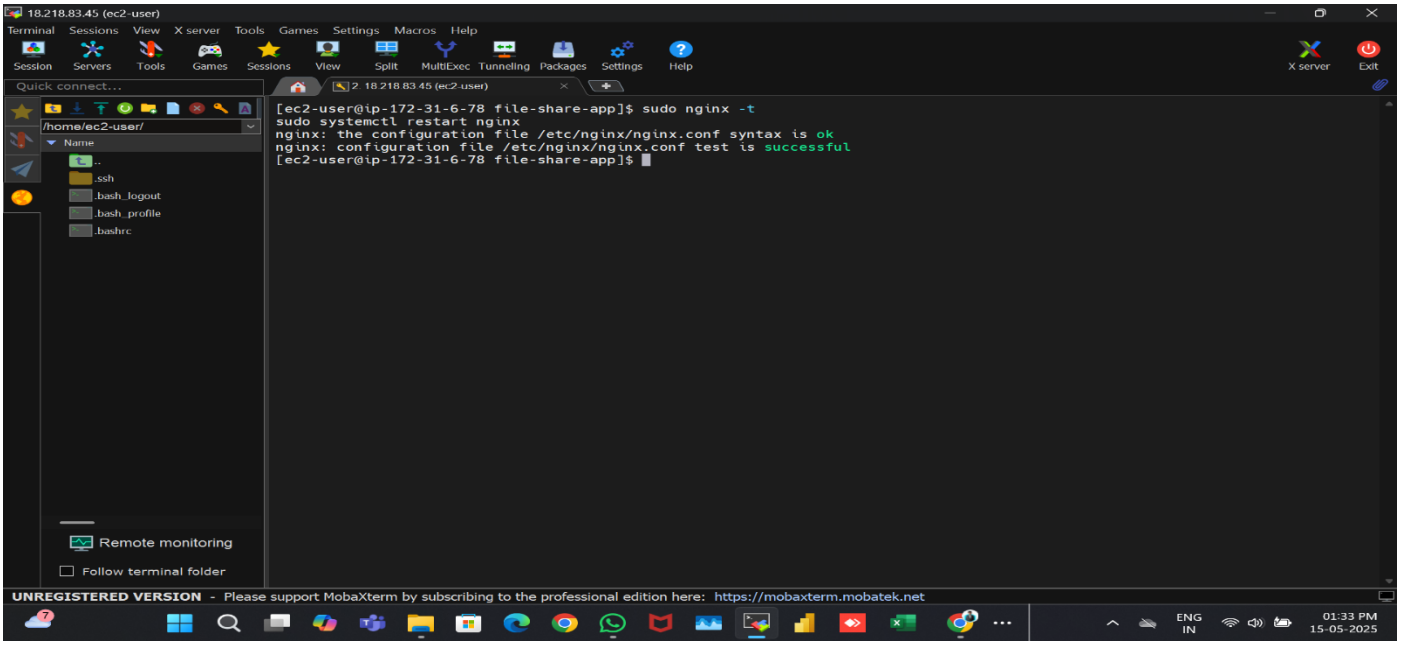
sudo systemctl restart nginx

sudo systemctl enable nginx



Purpose/Action:

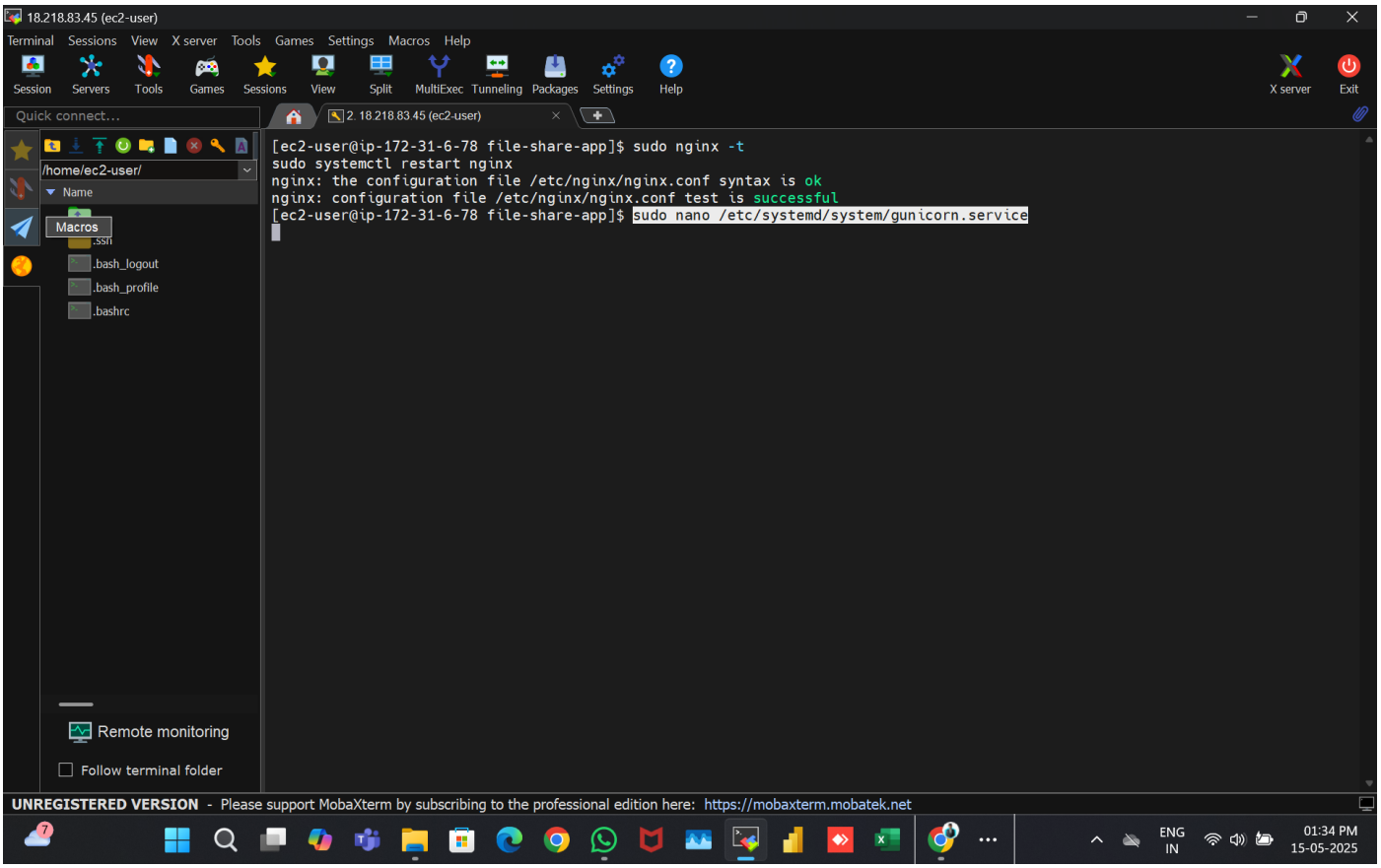
- sudo nginx -t
  - **Tests the Nginx configuration** for syntax errors.
  - If there's an error in /etc/nginx/nginx.conf, this command will tell you **before restarting** the service.
- sudo systemctl restart nginx
  - **Restarts the Nginx service**, applying any changes made to the configuration file.



use a gunicorn.service file?

- Automatically start Gunicorn on boot
- Keep the app running in the background
- Restart automatically if it crashes
- Manage with systemctl (start, stop, restart, enable, etc.)

sudo nano /etc/systemd/system/gunicorn.service



Add :

[Unit]

Description=Gunicorn instance to serve Flask app

After=network.target

[Service]

User=ec2-user

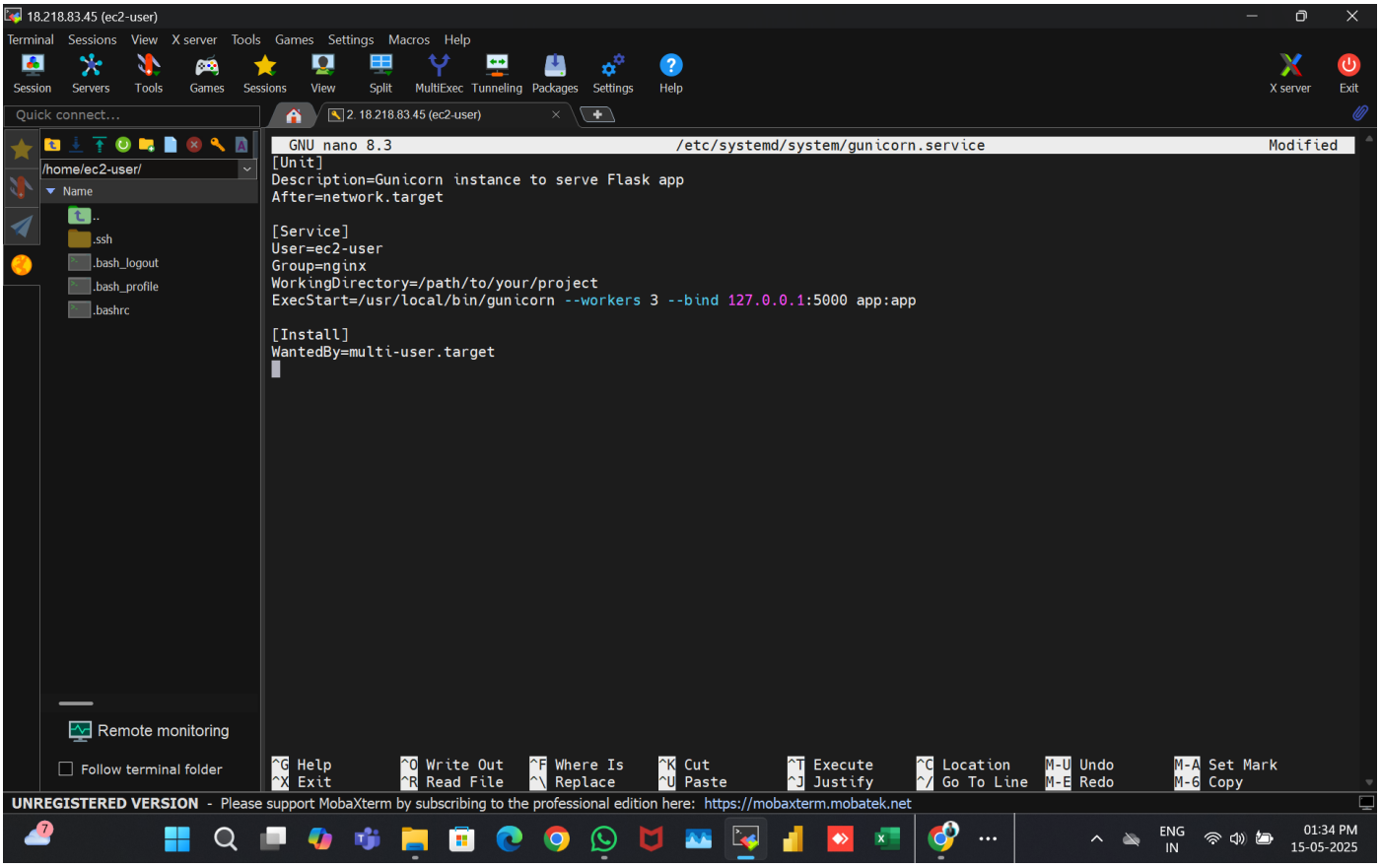
Group=nginx

WorkingDirectory=/path/to/your/project

ExecStart=/usr/local/bin/gunicorn --workers 3 --bind 127.0.0.1:5000 app:app

[Install]

WantedBy=multi-user.target



Now ,

Re-executes systemd (used after system updates).

Reloads systemd to apply changes in service files.

Starts the Gunicorn service immediately.

Enables Gunicorn to start automatically on boot.

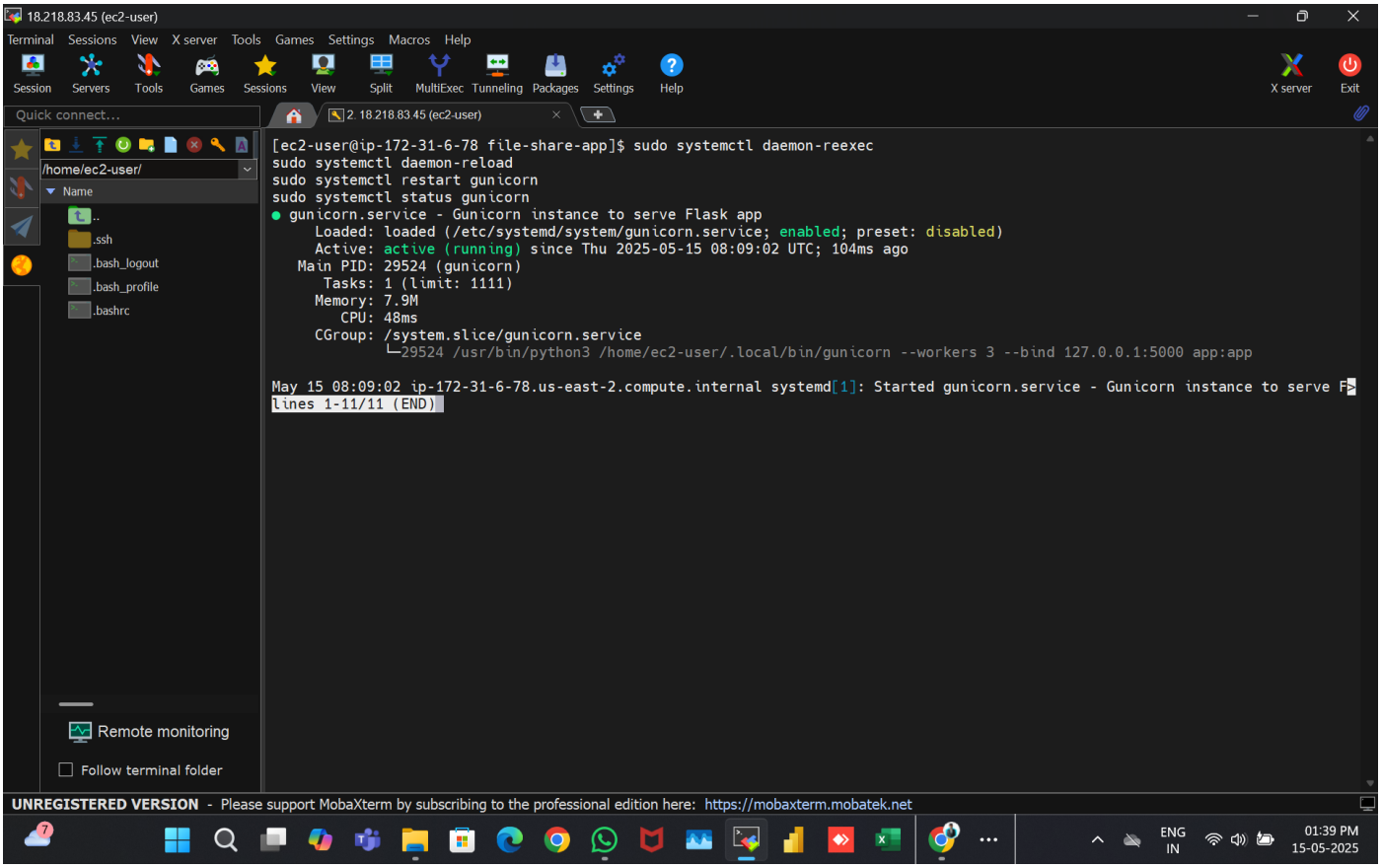
Cmd :

sudo systemctl daemon-reexec

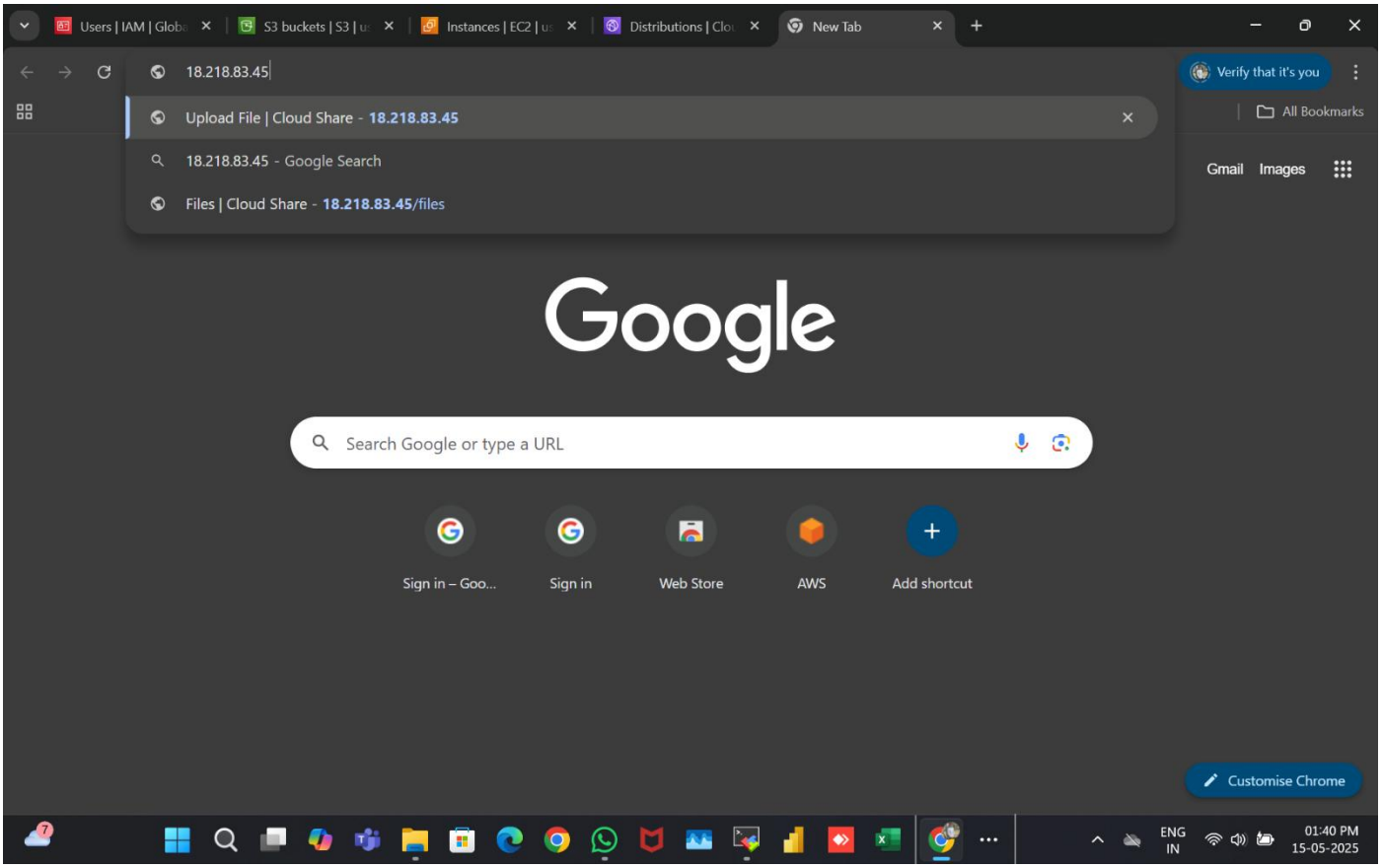
sudo systemctl daemon-reload

sudo systemctl start gunicorn

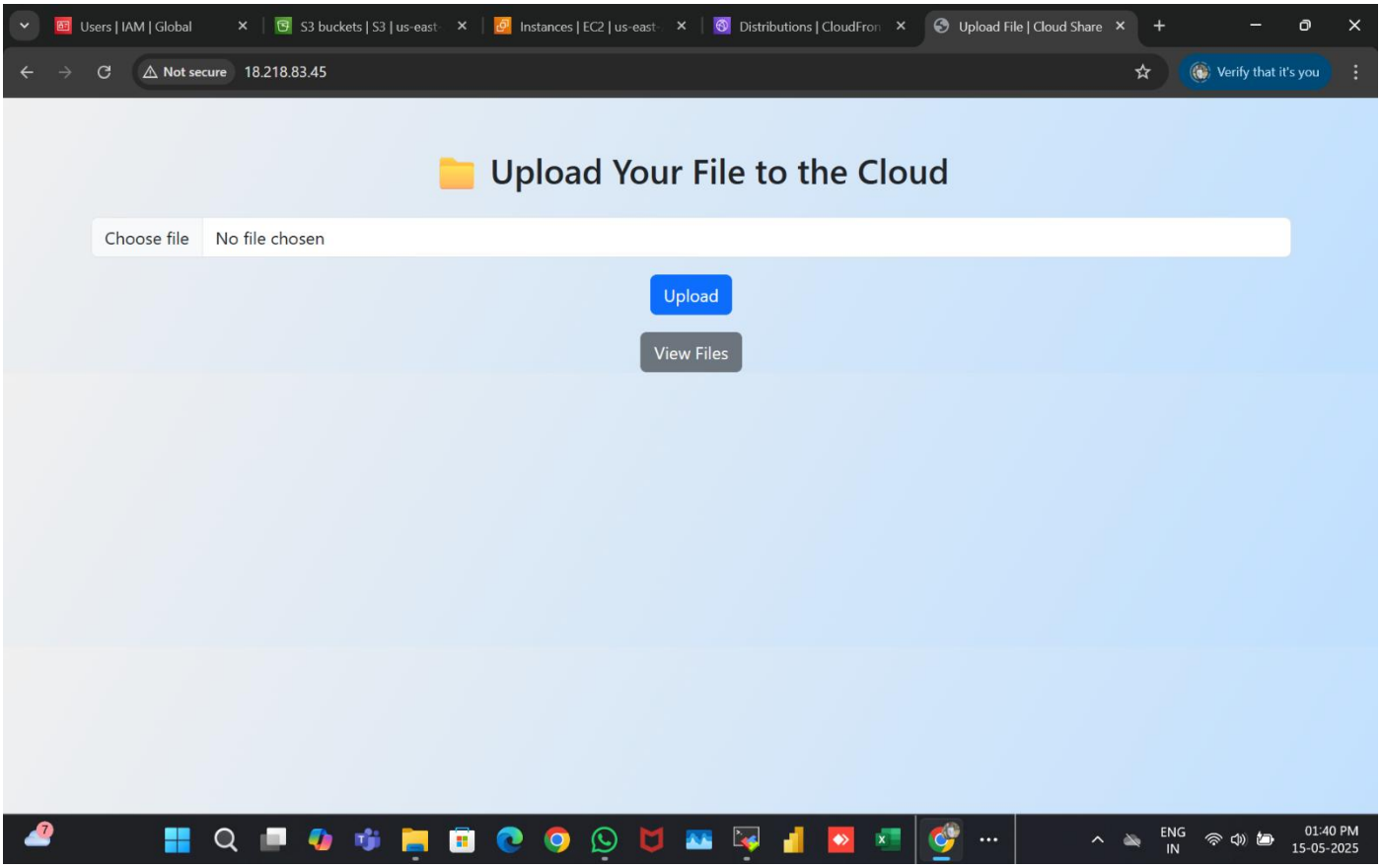
sudo systemctl enable gunicorn

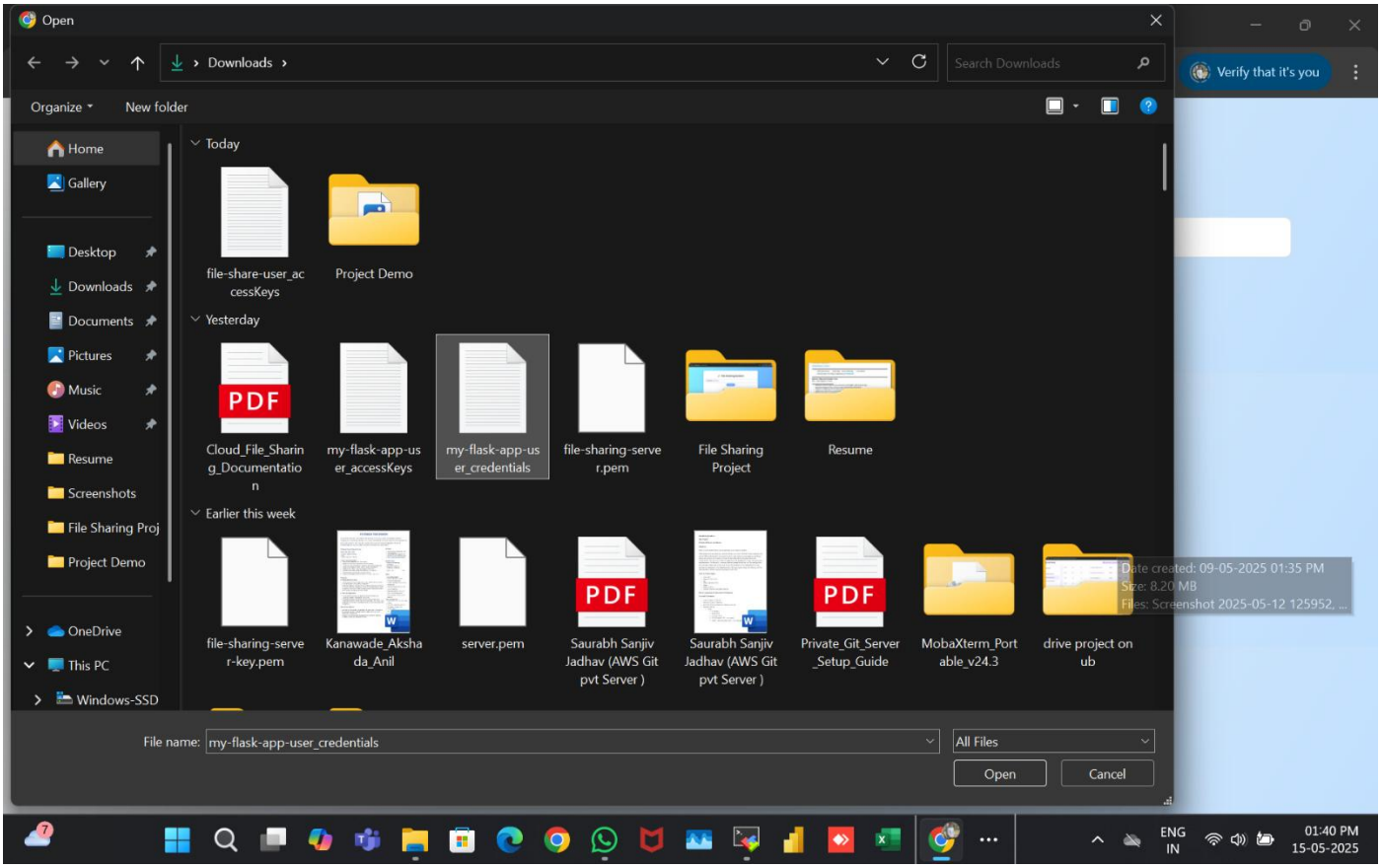


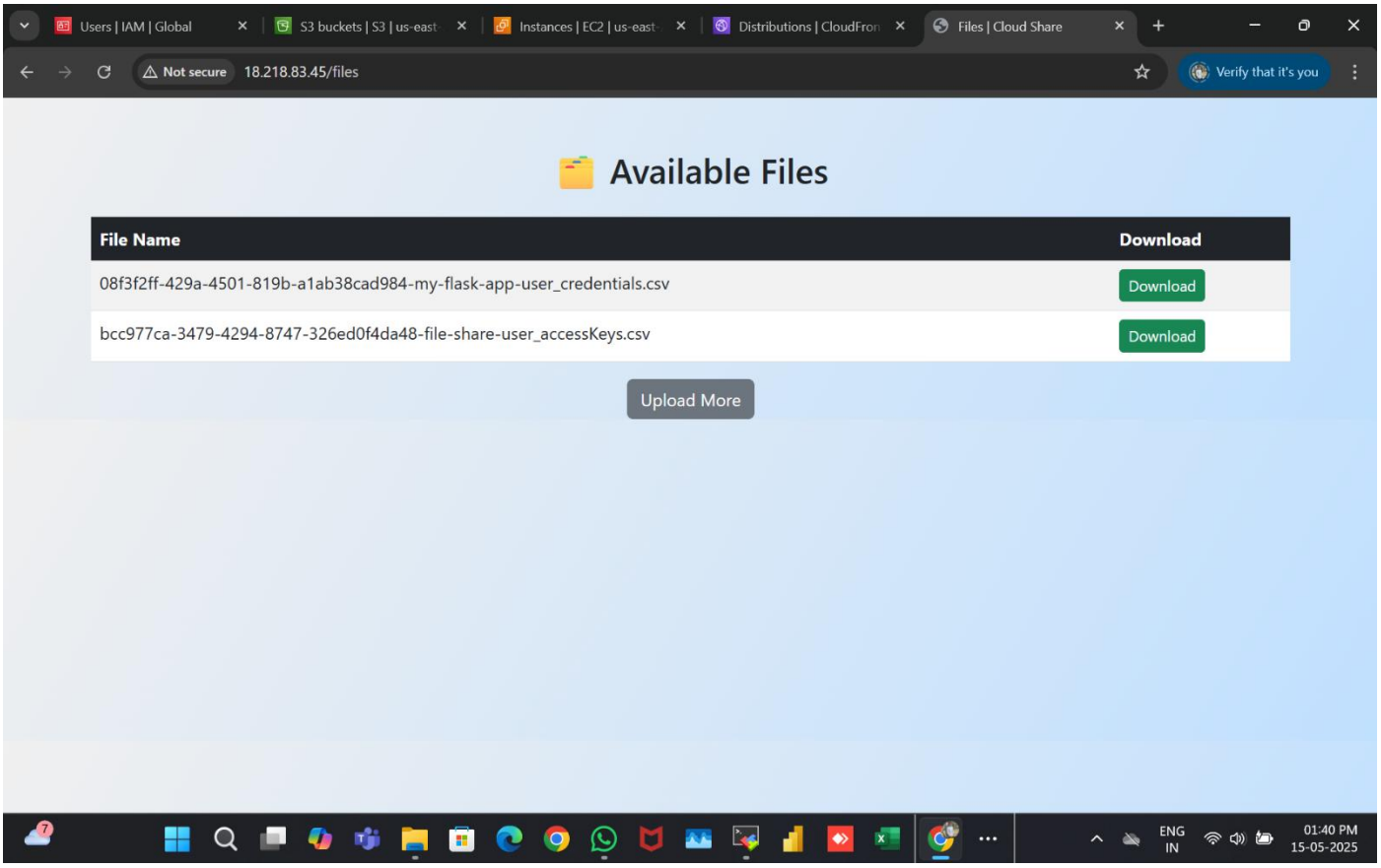
Now access your web app at:
http://18.218.83.45



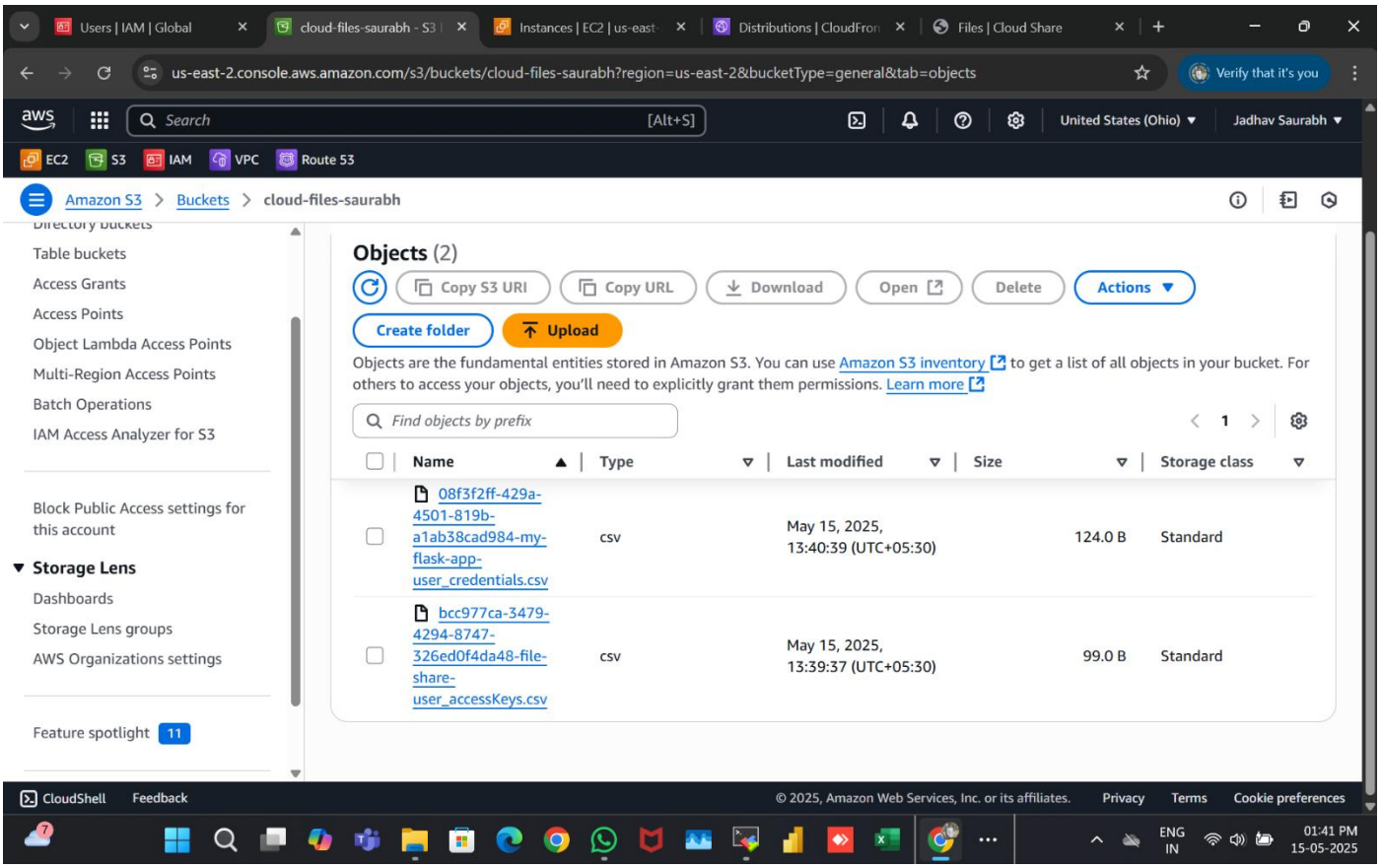
Now its Successfully working.







Uploads a file - Flask app uploads it to your S3 bucket.

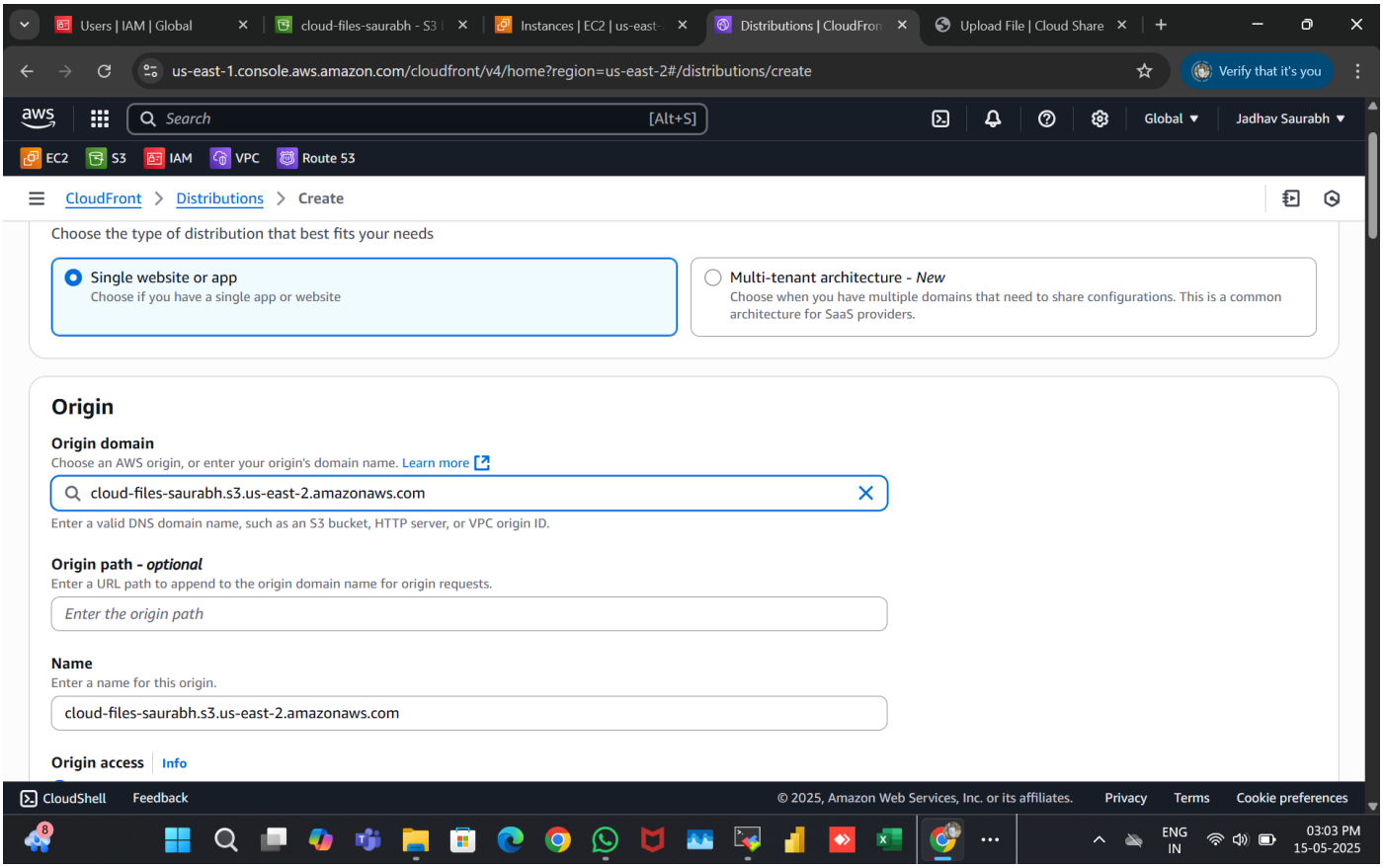


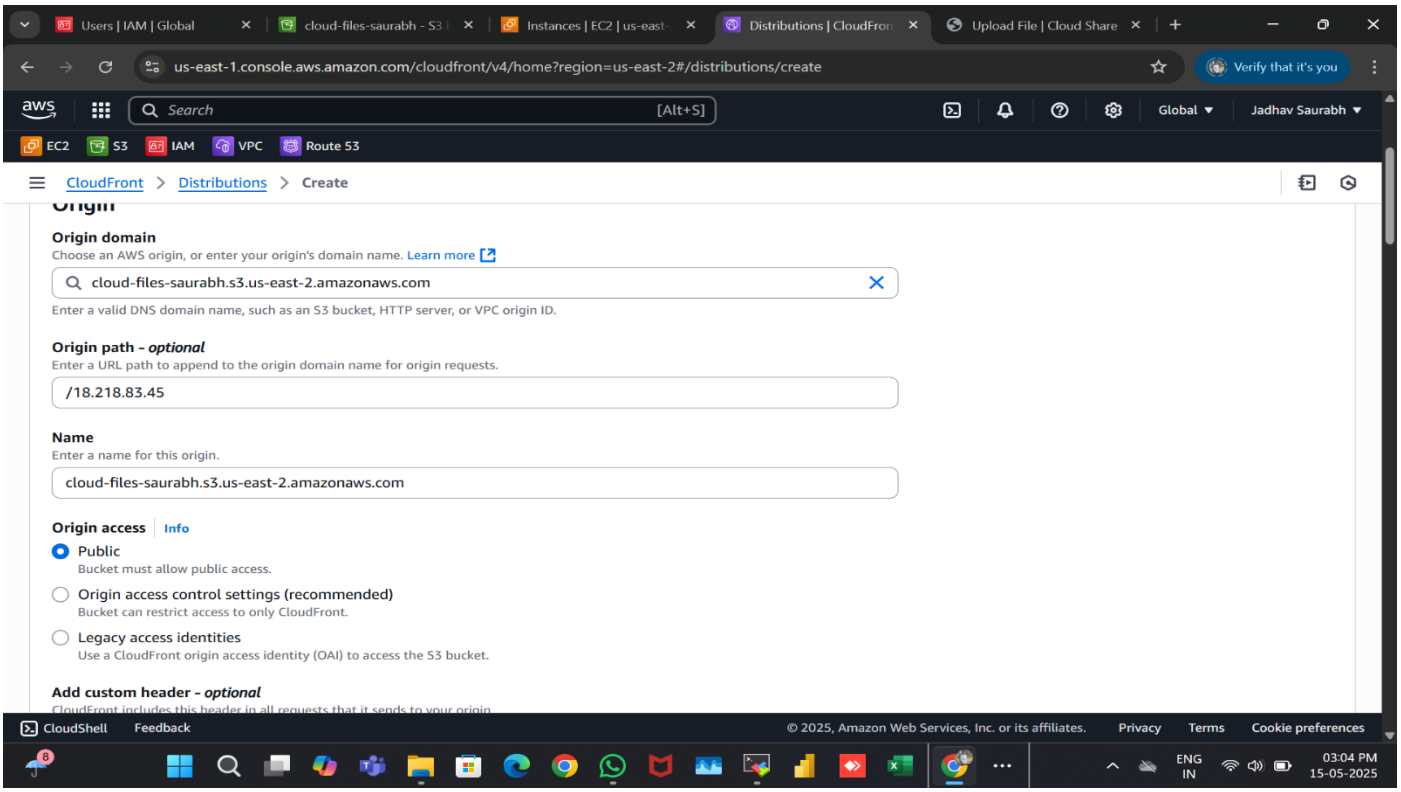
Purpose of Amazon CloudFront

Amazon CloudFront is a Content Delivery Network (CDN) service provided by AWS. Its primary purpose is to deliver content (like images, videos, websites, and files) to users with low latency and high transfer speed using a global network of edge locations.

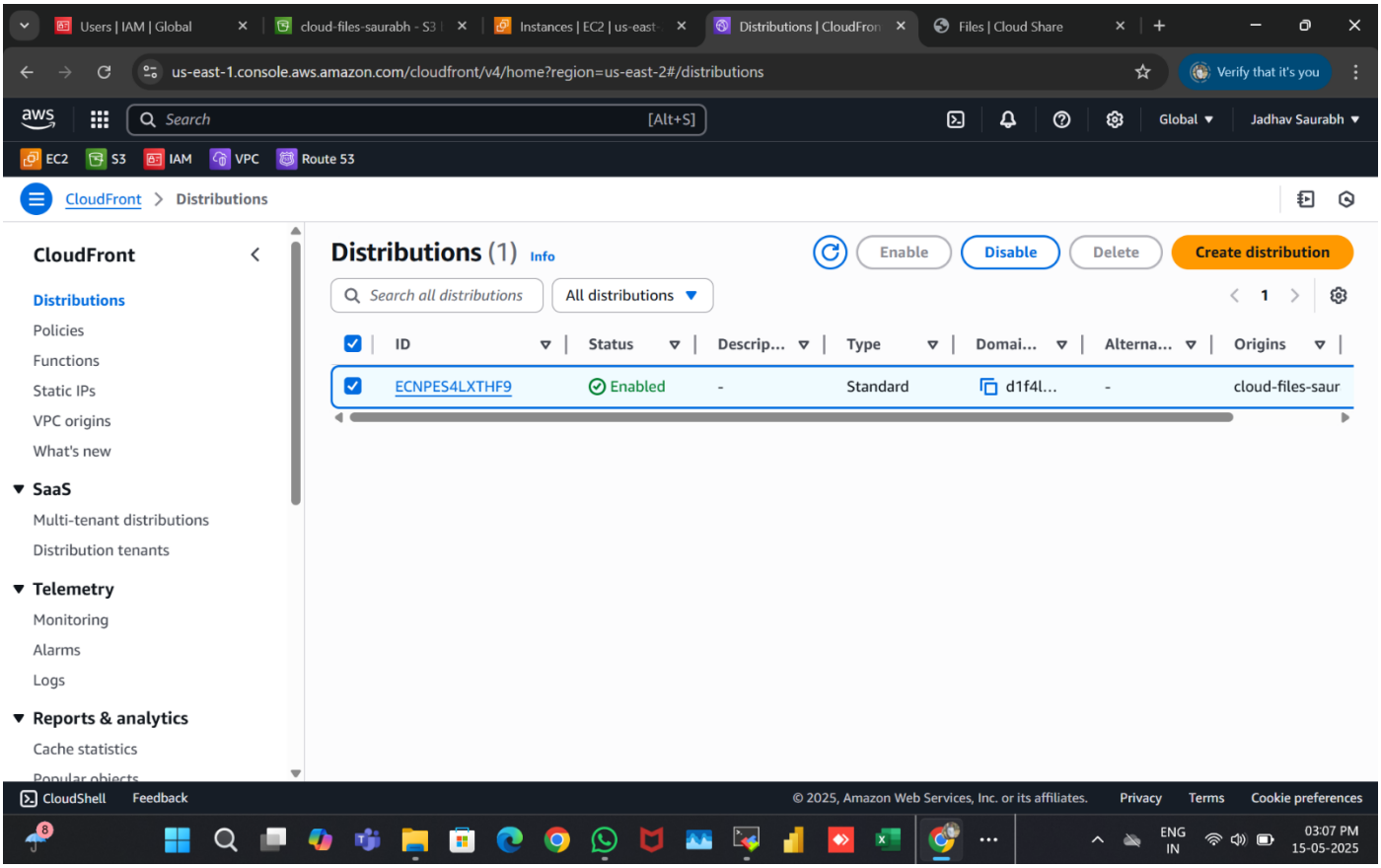
User gets fast and secure access via CloudFront (with optional caching and signed URL support .

Create distribution





Now its Working.



we successfully built a secure, scalable, and globally accessible cloud-based file sharing system using AWS S3, EC2, IAM, CloudFront, and Nginx—delivering performance, security, and reliability at its core.

Thank you.