

# 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

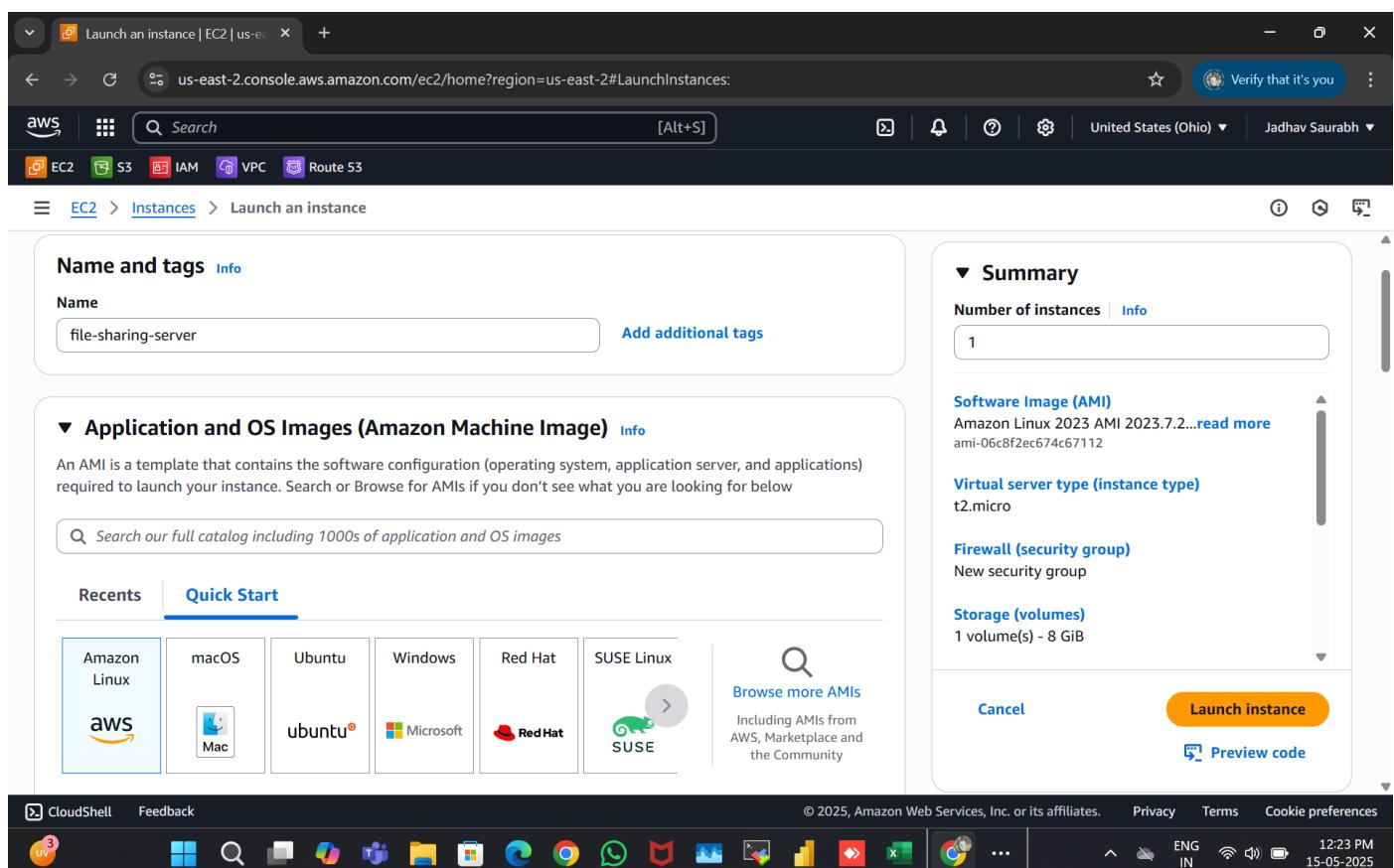
## Security Implementation :

- 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 :

### Configure as follows :

- **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)



The screenshot shows the 'Launch an instance' wizard in the AWS EC2 console. The first step, 'Instance type', is selected. A 't2.micro' instance is chosen, which is marked as 'Free tier eligible'. Other options like 'On-Demand Ubuntu Pro base pricing' and 'On-Demand Linux base pricing' are listed. A toggle switch allows selecting 'All generations'. A link to 'Compare instance types' is available. The 'Summary' section on the right shows 1 instance being launched with the 'Amazon Linux 2023 AMI 2023.7.2...' selected. The 'Virtual server type (instance type)' is set to 't2.micro'. The 'Network settings' section includes a key pair named 'file-sharing-server'. The 'Storage (volumes)' section shows 1 volume(s) - 8 GiB. Buttons for 'Cancel', 'Launch instance', and 'Preview code' are at the bottom.

## Assign Security groups.

The screenshot shows the 'Launch an instance' wizard in the AWS EC2 console, with the 'Security groups' step selected. Under 'Firewall (security groups)', there are two options: 'Create security group' (radio button) and 'Select existing security group' (radio button). The 'Select existing security group' option is selected, and a dropdown menu shows 'Primary-SG sg-0cf1ea1415240ffb8 VPC: vpc-02ccf3089f48a3687'. A link to 'Compare security group rules' is provided. The 'Summary' section on the right shows 1 instance being launched with the 'Amazon Linux 2023 AMI 2023.7.2...' selected. The 'Virtual server type (instance type)' is set to 't2.micro'. The 'Storage (volumes)' section shows 1 volume(s) - 8 GiB. Buttons for 'Cancel', 'Launch instance', and 'Preview code' are at the bottom.

## Ec2 Instance is Successfully Launch. And Copy IP Address

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with 'EC2' selected under 'Instances'. The main area displays 'Instances (1/1)' with a single item: 'file-sharing-se...' (Instance ID: i-05e69dd59acb867ee). The instance is listed as 'Running' with an 't2.micro' instance type. A tooltip over the Public IPv4 address '18.218.83.45' indicates it has been copied. Below the instance details, there are tabs for 'Details', 'Status and alarms', 'Monitoring', 'Security', 'Networking', 'Storage', and 'Tags'. At the bottom of the page, there's a footer with various AWS services and system status indicators.

## 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**

Screenshot of the AWS S3 'Create bucket' wizard.

**Bucket type**

- General purpose  
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
- Directory  
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

**Bucket name** [Info](#)  
cloud-files-saurabh

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn More](#)

**Copy settings from existing bucket - optional**  
Only the bucket settings in the following configuration are copied.  
[Choose bucket](#)

Format: s3://bucket/prefix

**Object Ownership** [Info](#)

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

- ACLs disabled (recommended)  
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ACLs enabled  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 12:29 PM 15-05-2025

**Bucket is ready.**

Screenshot of the AWS S3 'Buckets' page.

**Account snapshot - updated every 24 hours** [All AWS Regions](#) [View Storage Lens dashboard](#)

Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets. [Learn more](#)

[General purpose buckets](#) [Directory buckets](#)

**General purpose buckets (1/1)** [Info](#) [All AWS Regions](#)

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
cloud-files-saurabh	US East (Ohio) us-east-2	<a href="#">View analyzer for us-east-2</a>	May 15, 2025, 12:29:42 (UTC+05:30)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 12:29 PM 15-05-2025

## 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:
  - o AmazonS3FullAccess (*or custom policy for bucket*)
5. Save **Access Key ID** and **Secret Access Key**

The screenshot shows the 'Create user' wizard in the AWS IAM console. The current step is 'Step 3: Set permissions'. On the left, a sidebar lists steps: 'Set permissions' (selected), 'Step 3', 'Review and create', 'Step 4', and 'Retrieve password'. The main area is titled 'User details' with a 'User name' field containing 'file-share-user'. Below it, a note says: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)'. A checked checkbox says 'Provide user access to the AWS Management Console - optional'. A tooltip for this checkbox states: 'If you're providing console access to a person, it's a best practice [link] to manage their access in IAM Identity Center.' A large blue-bordered box contains the question 'Are you providing console access to a person?'. It has two options: 'Specify a user in Identity Center - Recommended' (selected) and 'I want to create an IAM user'. A note for the second option says: 'We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keypairs, or a backup credential for emergency account access.' Another blue-bordered box at the bottom contains the note: 'If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keypairs, you can generate them after you create this IAM user. [Learn more](#)'.

The screenshot shows the AWS IAM 'Create user' wizard at Step 4: Set permissions policies. The user is selecting the 'AmazonS3FullAccess' policy from a list of AWS managed policies. The policy is highlighted with a blue border.

**Permissions policies (1/1348)**

Choose one or more policies to attach to your new user.

Policy name	Type	Attached entities
AmazonDMSRedshiftS3Role	AWS managed	0
<b>AmazonS3FullAccess</b>	AWS managed	0
AmazonS3ObjectLambdaE...	AWS managed	0
AmazonS3OutpostsFullAc...	AWS managed	0
AmazonS3OutpostsReadQ...	AWS managed	0
AmazonS3ReadOnlyAccess	AWS managed	0

## User Created.

The screenshot shows the AWS IAM 'Users' page. A single user, 'file-share-user', is listed. The user was created 15-05-2025 at 12:31 PM. The user has no groups, no MFA, and no password age.

**Identity and Access Management (IAM)**

Users (1/1) [Info](#)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Group:	Last activity	MFA	Password age
file-share-user	/	0	-	-	-

## Create a Access key .

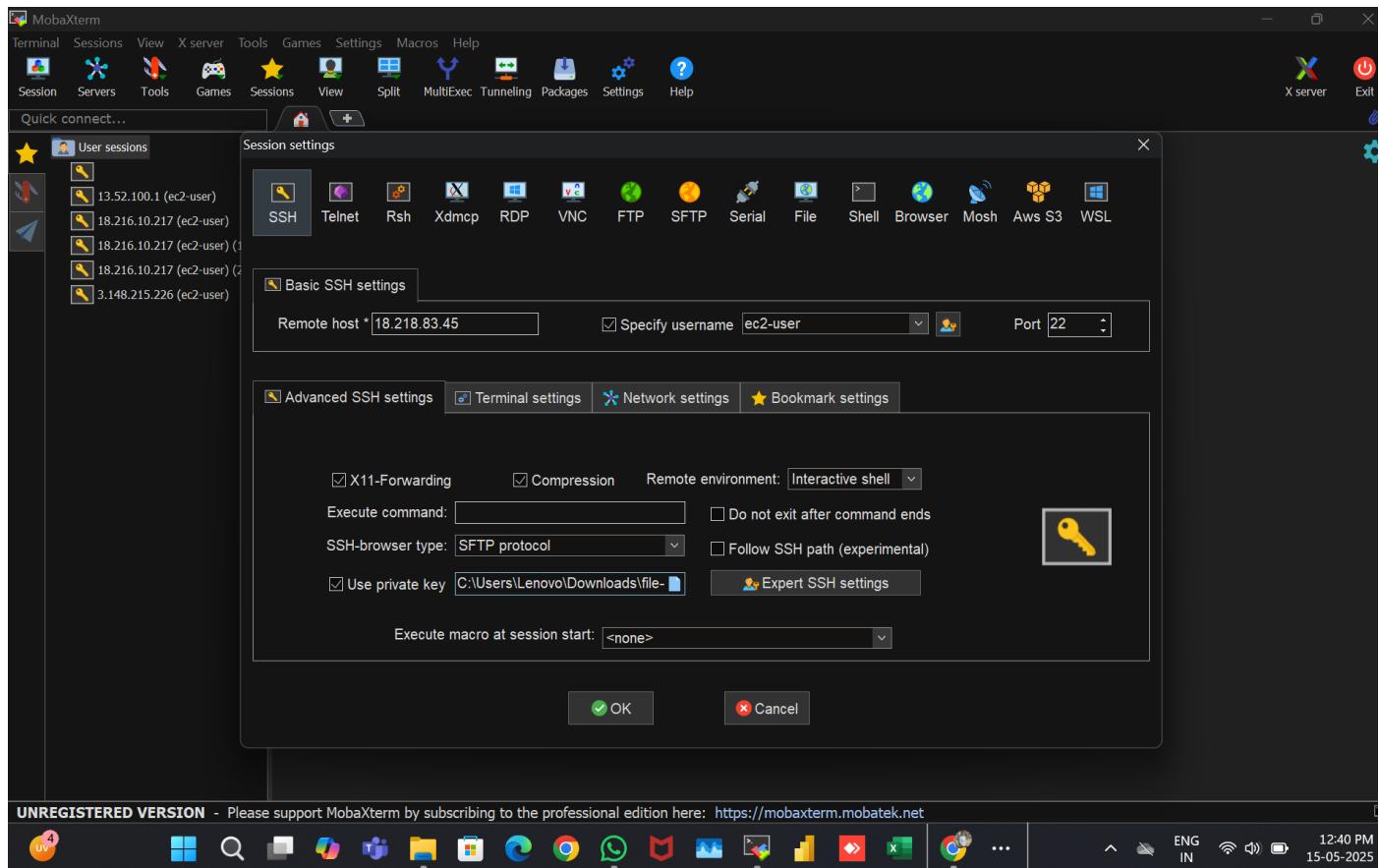
The screenshot shows the 'Create access key' wizard in the AWS IAM console. The title bar says 'Create access key | IAM | Global'. The main content area is titled 'Access key best practices & alternatives' with a sub-section 'Use case'. Under 'Use case', the 'Command Line Interface (CLI)' option is selected, with a note: 'You plan to use this access key to enable the AWS CLI to access your AWS account.' Other options include 'Local code', 'Application running on an AWS compute service', and 'Third-party service'. On the left sidebar, 'Access key best practices & alternatives' is highlighted. Below it are 'Step 2 - optional' (Set description tag) and 'Step 3' (Retrieve access keys). The bottom of the screen shows the Windows taskbar with various pinned icons.

## Download .CSV file.

The screenshot shows the 'Create access key' wizard in the AWS IAM console, step 3. A green success message box says 'Access key created' and 'This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' Below this, there's a note: 'If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.' A table shows the 'Access key' (AKIA54WIF2NK5RPF1674) and 'Secret access key' (redacted). At the bottom, a section titled 'Access key best practices' lists: 'Never store your access key in plain text, in a code repository, or in code.', 'Disable or delete access key when no longer needed.', 'Enable least-privilege permissions.', and 'Rotate access keys regularly.' A link 'For more details about managing access keys, see the best practices for managing AWS access keys.' is provided. At the bottom right are 'Download .csv file' and 'Done' buttons. The bottom of the screen shows the Windows taskbar.

## 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
```

```
• 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
~~ \|/ V~' '--> https://aws.amazon.com/linux/amazon-linux-2023
~~
~~ .-'/ \
~~ /m' \
[ec2-user@ip-172-31-6-78 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-6-78 ~]$
```

160 kB/s | 16 kB 00:00

Remote monitoring

Follow terminal folder

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

```
• 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.

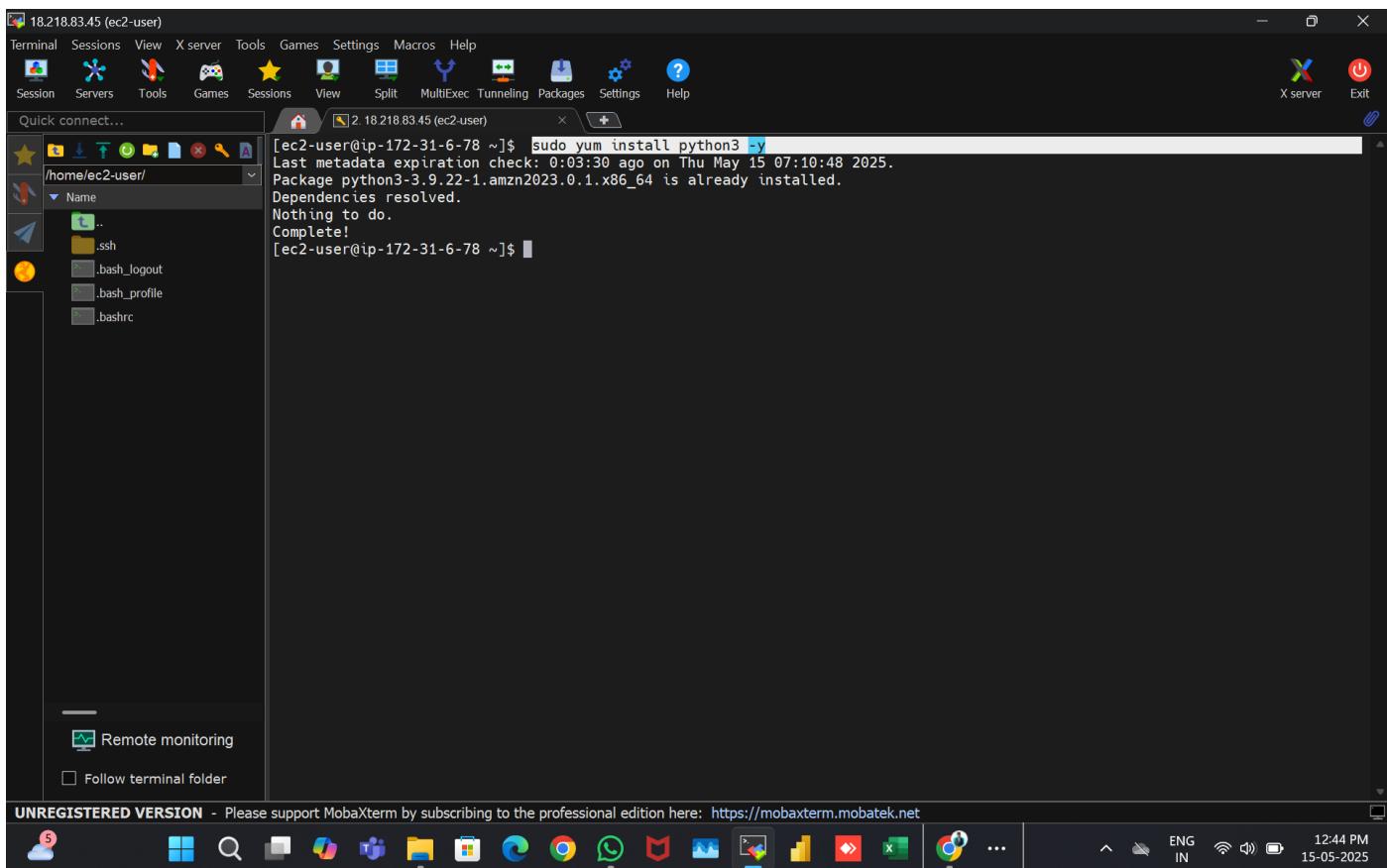
,
~\_#####
~~ \#####
~~ \###| https://aws.amazon.com/linux/amazon-linux-2023
~~
~~ .-'/ \
~~ /m' \
[ec2-user@ip-172-31-6-78 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-6-78 ~]$ sudo yum install python3 git nginx -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-httd 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-mime types 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
```

160 kB/s | 16 kB 00:00

Remote monitoring

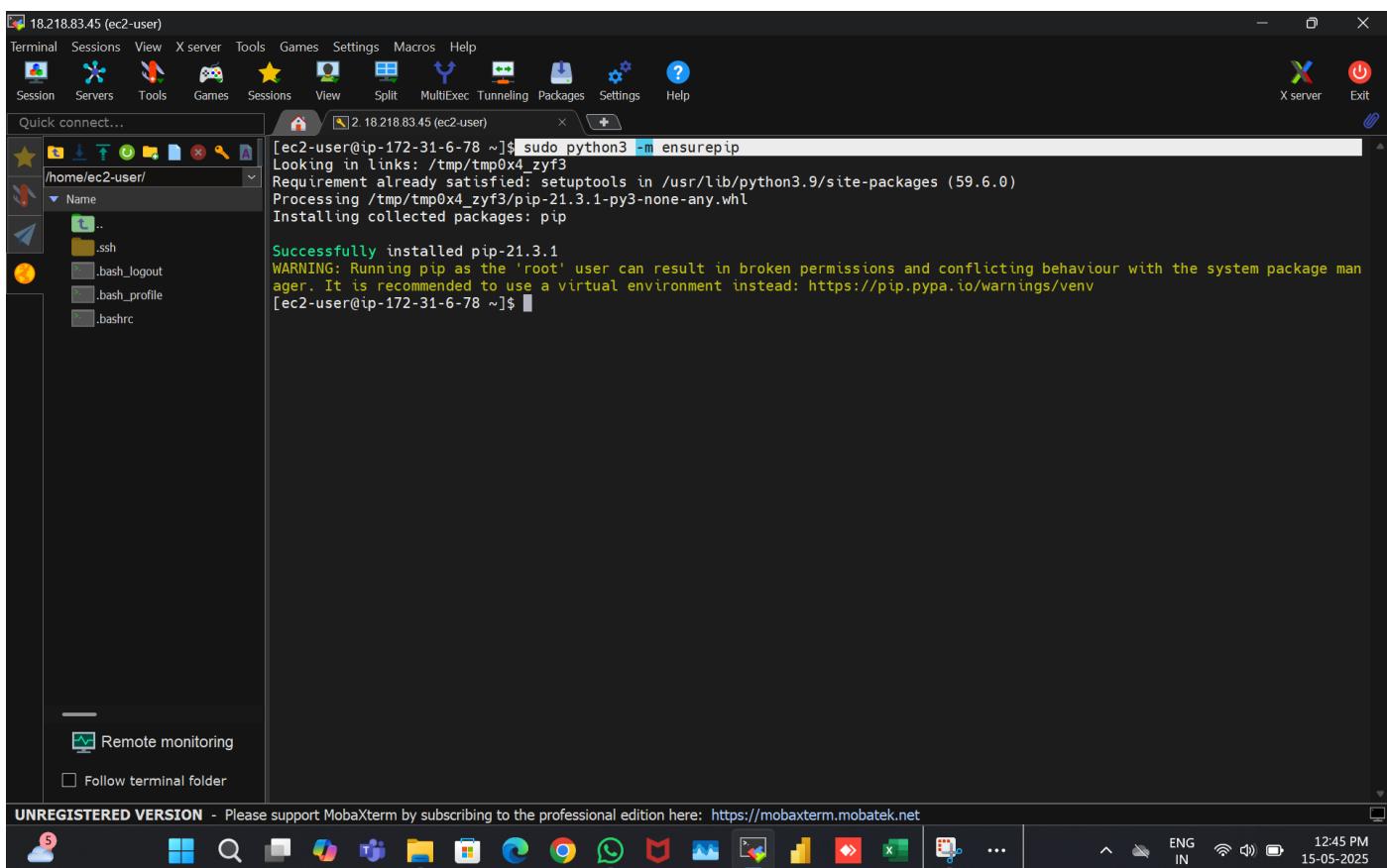
Follow terminal folder

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>



```
[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>



```
[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

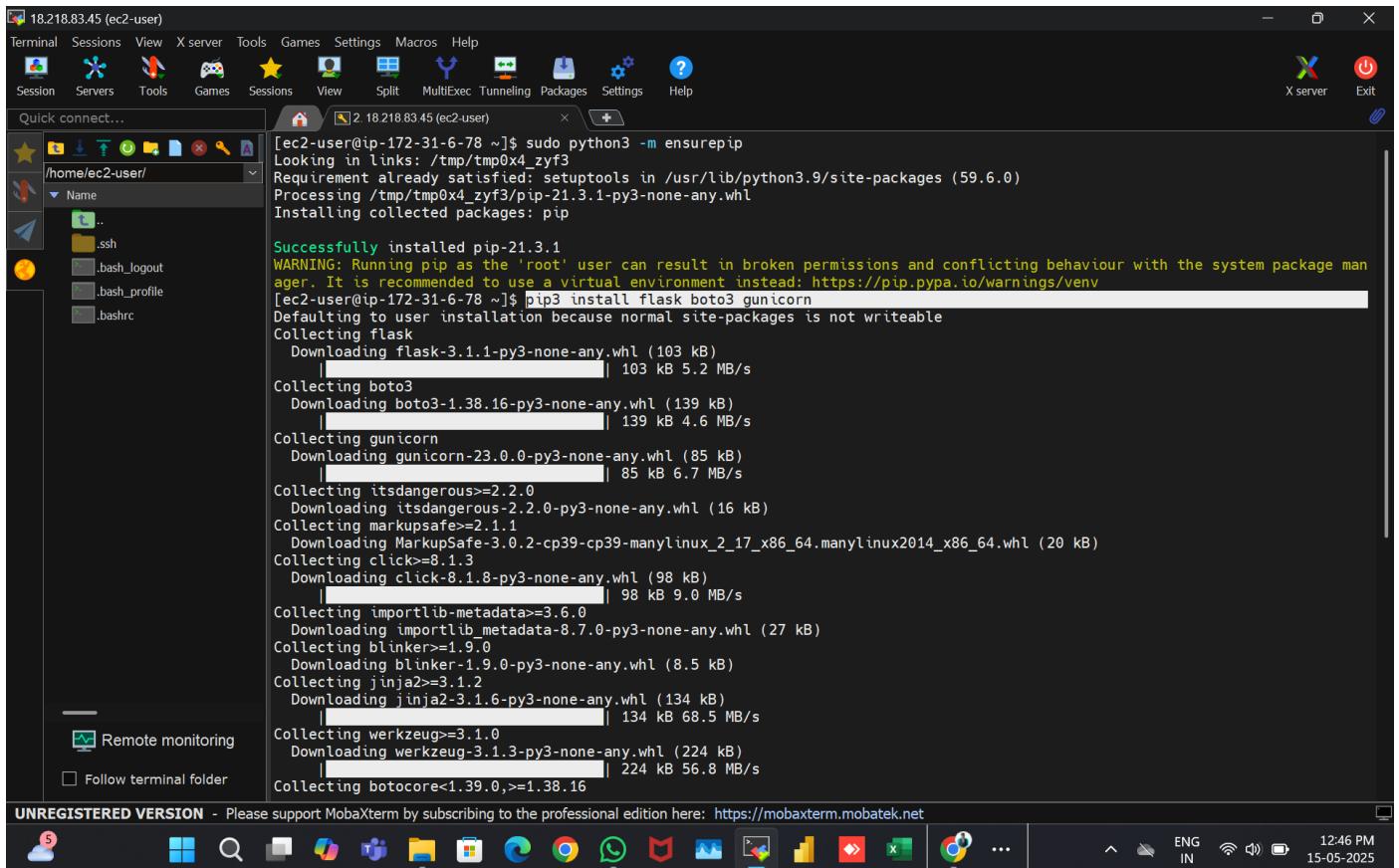
Successfully installed pip-21.3.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-172-31-6-78 ~]$
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

## Install Python Dependencies

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

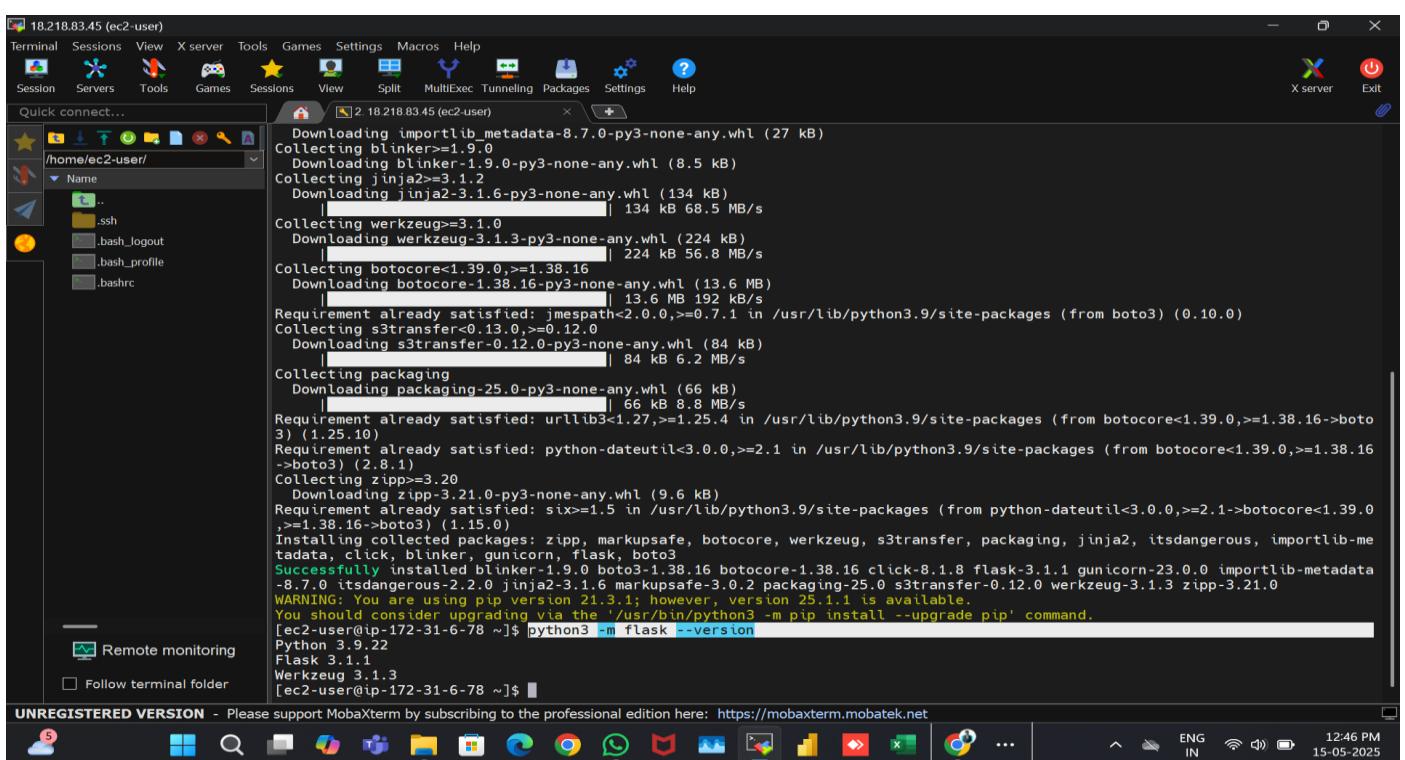
pip install flask boto3 gunicorn



```
[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 ~]$ pip3 install flask boto3 gunicorn
Defaulting to user installation because normal site-packages is not writeable
Collecting flask
  Downloading flask-3.1.1-py3-none-any.whl (103 kB)
    ━━━━━━━━| 103 kB 5.2 MB/s
Collecting boto3
  Downloading boto3-1.38.16-py3-none-any.whl (139 kB)
    ━━━━━━| 139 kB 4.6 MB/s
Collecting gunicorn
  Downloading gunicorn-23.0.0-py3-none-any.whl (85 kB)
    ━━━━| 85 kB 6.7 MB/s
Collecting itsdangerous>=2.2.0
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting markupsafe>=2.1.1
  Downloading MarkupSafe-3.0.2-cp39-cp39-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (20 kB)
Collecting click>=8.1.3
  Downloading click-8.1.8-py3-none-any.whl (98 kB)
    ━━━━| 98 kB 9.0 MB/s
Collecting importlib-metadata>=3.6.0
  Downloading importlib_metadata-8.7.0-py3-none-any.whl (27 kB)
Collecting blinker>=1.9.0
  Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Collecting jinja2>=3.1.2
  Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
    ━━━━| 134 kB 68.5 MB/s
Collecting werkzeug>=3.1.0
  Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
    ━━━━| 224 kB 56.8 MB/s
Collecting botocore<1.39.0,>=1.38.16
```

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



```
[ec2-user@ip-172-31-6-78 ~]$ python3 -m flask --version
Python 3.9.22
Flask 3.1.1
Werkzeug 3.1.3
[ec2-user@ip-172-31-6-78 ~]$
```

## Create Project Directory and Navigate

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

The screenshot shows a MobaXterm window titled "18.218.83.45 (ec2-user)". The terminal session is running on a Linux system. The user has run the command "mkdir file-share-app && cd file-share-app" in the terminal. The terminal window also displays the message "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net". The desktop environment includes a taskbar at the bottom with various application icons.

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

nano app.py

The screenshot shows a MobaXterm window titled "18.218.83.45 (ec2-user)". The terminal session is running on a Linux system. The user has run the command "nano app.py" in the terminal. The terminal window also displays the message "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net". The desktop environment includes a taskbar at the bottom with various application icons.

```
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()
```

```

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
Quick connect...
2.18.218.83.45 (ec2-user) app.py
GNU nano 8.3
from flask import Flask, render_template, request, redirect
import boto3
import uuid

app = Flask(__name__, static_url_path='/static')

BUCKET_NAME = 'cloud-files-saurabh'
ACCESS_KEY = 'AKIA54WI2NK5RPFI674'
SECRET_KEY = 'pKbvZ3PN8jTXQl/MHcxk205/8GbmzbksZoYcTGf7'

s3 = boto3.client('s3',
                  aws_access_key_id=ACCESS_KEY,
                  aws_secret_access_key=SECRET_KEY
                  )

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload():
    file = request.files['file']
    filename = str(uuid.uuid4()) + "-" + file.filename
    s3.upload_fileobj(file, BUCKET_NAME, filename)
    return redirect('/files')

@app.route('/files')
def list_files():
    objects = s3.list_objects_v2(Bucket=BUCKET_NAME).get('Contents', [])
    urls = []
    for obj in objects:
        url = s3.generate_presigned_url('get_object',
                                         Params={'Bucket': BUCKET_NAME, 'Key': obj['Key']},
                                         ExpiresIn=3600)
        urls.append(url)
    return render_template('files.html', urls=urls)

[Read 39 lines]
^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^W Replace ^U Paste ^J Justify ^V Go To Line M-E Redo
M-A Set Mark M-G Copy

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

## 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
```

```

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
Quick connect...
2.18.218.83.45 (ec2-user)
[ec2-user@ip-172-31-6-78 ~]$ mkdir file-share-app && cd file-share-app
[ec2-user@ip-172-31-6-78 file-share-app]$ nano app.py
[ec2-user@ip-172-31-6-78 file-share-app]$ nano app.py
[ec2-user@ip-172-31-6-78 file-share-app]$ mkdir templates
nano templates/index.html

```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

```
<!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>
```

```
GNU nano 8.3
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Upload File | Cloud Share</title>
    <link rel="stylesheet" href="/static/css/style.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <div class="container mt-5 text-center fade-in">
        <h2 class="mb-4">Upload Your File to the Cloud</h2>
        <form method="POST" action="/upload" enctype="multipart/form-data">
            <input class="form-control mb-3" type="file" name="file" required>
            <button class="btn btn-primary">Upload</button>
        </form>
        <a href="/files" class="btn btn-secondary mt-3">View Files</a>
    </div>
</body>
</html>
```

**Create or Edit the File and add following code .**

nano templates/files.html

```
[ec2-user@ip-172-31-6-78 ~]$ mkdir file-share-app && cd file-share-app
[ec2-user@ip-172-31-6-78 file-share-app]$ nano app.py
[ec2-user@ip-172-31-6-78 file-share-app]$ nano app.py
[ec2-user@ip-172-31-6-78 file-share-app]$ mkdir templates
nano templates/index.html
[ec2-user@ip-172-31-6-78 file-share-app]$ 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>

</body>
```

```

</table>

<div class="text-center mt-3">
    <a href="/" class="btn btn-secondary">Upload More</a>
</div>

</div>

</body>

</html>

```

```

18.218.83.45 (ec2-user)
Terminal Sessions View Xserver Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect... 2.18.218.83.45 (ec2-user) Modified
Name /home/ec2-user/
.. .ssh .bash_logout .bash_profile .bashrc
GNU nano 8.3
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Files | Cloud Share</title>
    <link rel="stylesheet" href="/static/css/style.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</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>
</body>
</html>

```

Remote monitoring

Follow terminal folder

Help Exit Read File Replace Paste Undo Redo Set Mark

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

ENG IN 01:09 PM 15-05-2025

## 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;
```

```
}
```

The screenshot shows a terminal window titled "18.218.83.45 (ec2-user)". The window contains a file named "static/css/style.css" being edited in the "GNU nano 8.3" editor. The code in the editor is:

```
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; }  
}  
  

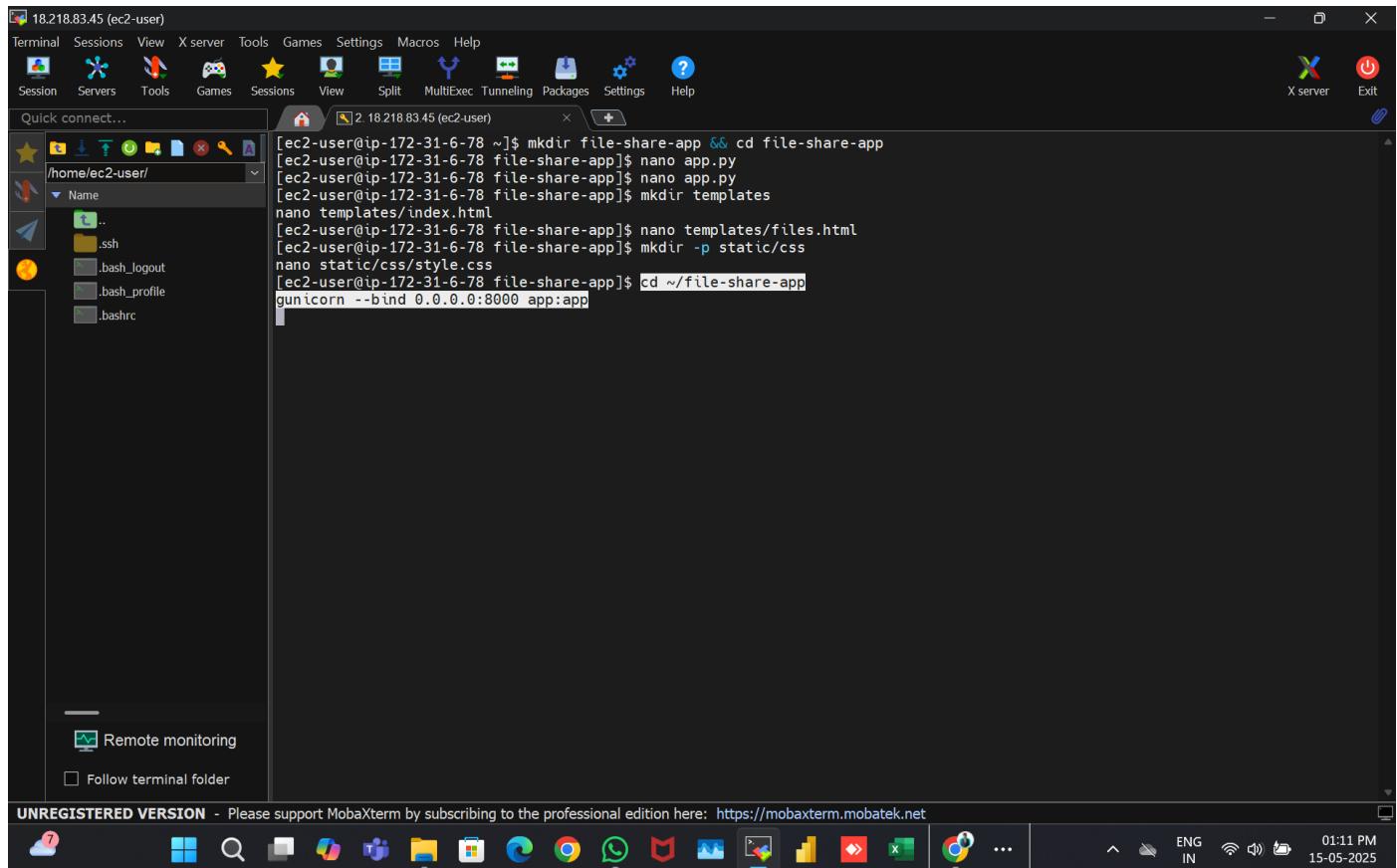
```

The terminal interface includes a menu bar with options like Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, Help, and a toolbar with icons for Session, Servers, Tools, Games, Sessions, Split, MultiExec, Tunneling, Packages, Settings, and Help. The bottom status bar shows "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net". The system tray at the bottom right shows various icons for network, battery, and system status.

**Run the following command :**

Cd ~file-share-app

gunicorn --bind 0.0.0.0:8000 app:app



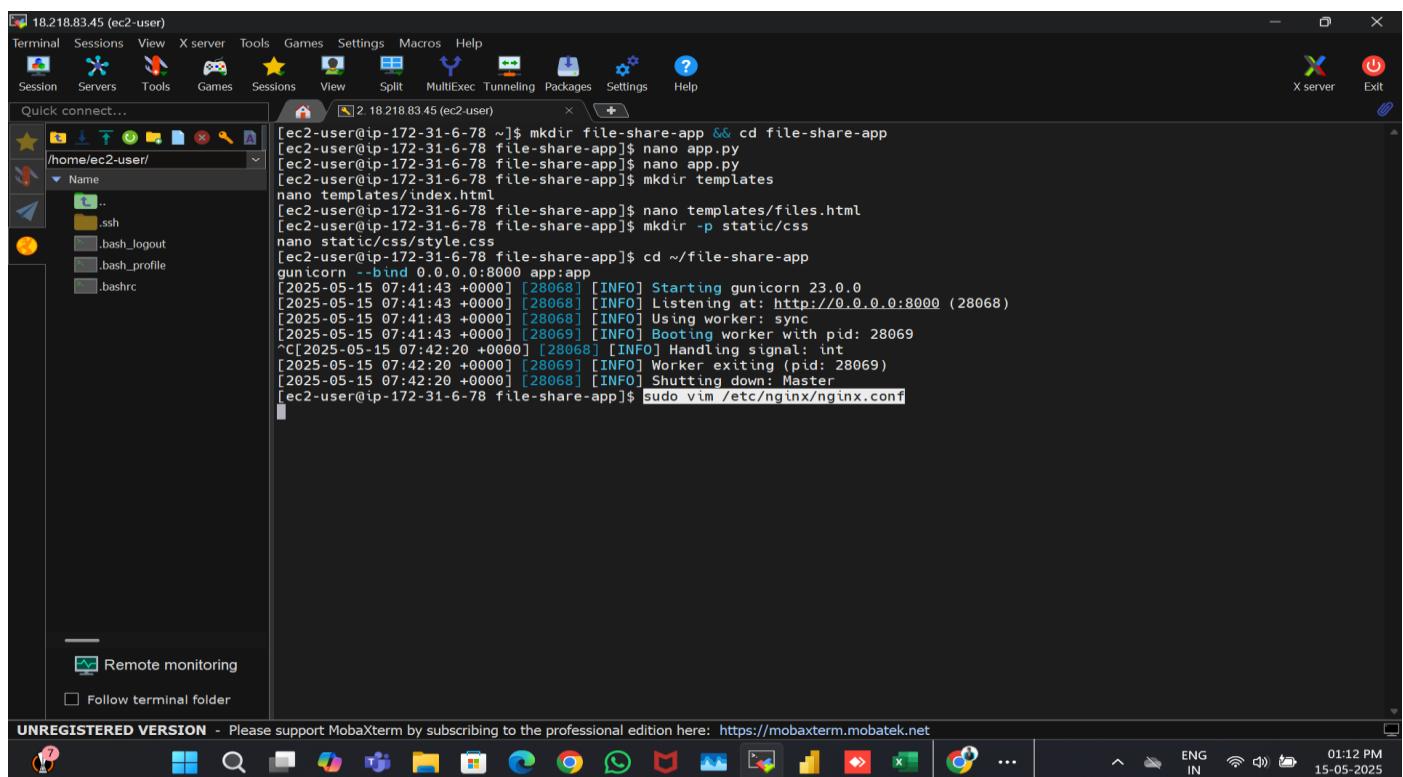
The screenshot shows a terminal window titled "18.218.83.45 (ec2-user)". The terminal session is running the command "gunicorn --bind 0.0.0.0:8000 app:app". The output of the command shows the creation of directory "file-share-app", nano editing of "app.py", "index.html", and "style.css", and the execution of "gunicorn". The terminal interface includes a sidebar with session management, a status bar at the bottom, and a taskbar at the very bottom.

```
[ec2-user@ip-172-31-6-78 ~]$ mkdir file-share-app && cd file-share-app
[ec2-user@ip-172-31-6-78 file-share-app]$ nano app.py
[ec2-user@ip-172-31-6-78 file-share-app]$ nano index.html
[ec2-user@ip-172-31-6-78 file-share-app]$ nano static/css/style.css
[ec2-user@ip-172-31-6-78 file-share-app]$ 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

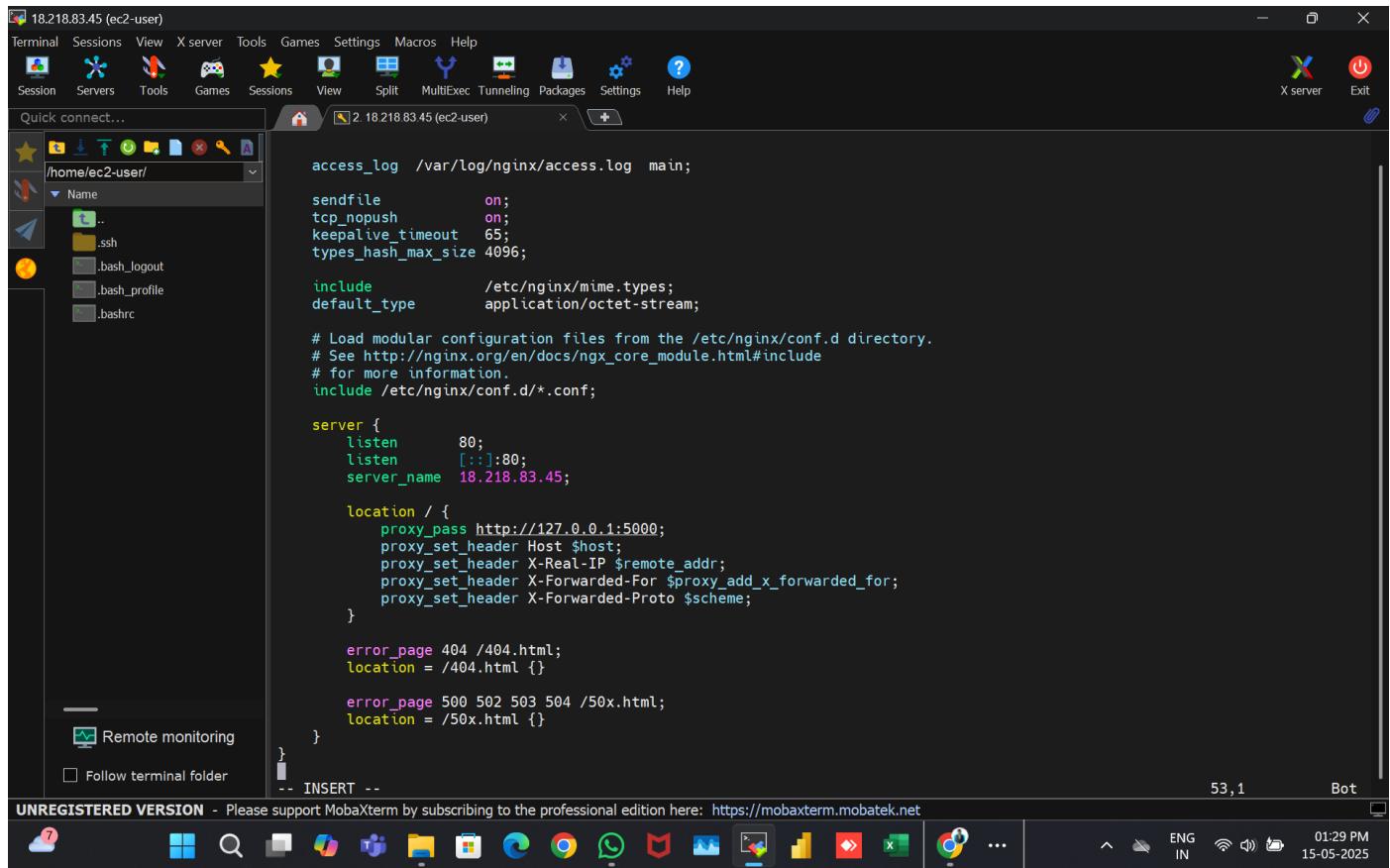


The screenshot shows a terminal window titled "18.218.83.45 (ec2-user)". The terminal session is running the command "sudo vim /etc/nginx/conf.d/file-share-app.conf". The output shows the configuration file being edited, with log messages from gunicorn indicating its startup and shutdown. The terminal interface includes a sidebar with session management, a status bar at the bottom, and a taskbar at the very bottom.

```
[ec2-user@ip-172-31-6-78 ~]$ sudo vim /etc/nginx/conf.d/file-share-app.conf
[2025-05-15 07:41:43 +0000] [28068] [INFO] Starting gunicorn 23.0.0
[2025-05-15 07:41:43 +0000] [28068] [INFO] Listening at: http://0.0.0.0:8000 (28068)
[2025-05-15 07:41:43 +0000] [28068] [INFO] Using worker: sync
[2025-05-15 07:41:43 +0000] [28069] [INFO] Booting worker with pid: 28069
^C[2025-05-15 07:42:20 +0000] [28069] [INFO] Handling signal: int
[2025-05-15 07:42:20 +0000] [28069] [INFO] Worker exiting (pid: 28069)
[2025-05-15 07:42:20 +0000] [28068] [INFO] Shutting down: Master
[ec2-user@ip-172-31-6-78 file-share-app]$ sudo vim /etc/nginx/nginx.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;  
    }  
}
```



The screenshot shows the MobaXterm interface on a Windows host. The terminal window displays the Nginx configuration file located at `/home/ec2-user/nginx.conf`. The configuration includes standard Nginx directives like `access_log`, `sendfile`, and `tcp_nopush`. It also includes a `server` block with a `location /` block that proxies requests to `http://127.0.0.1:8000` and sets various headers. The terminal window has a dark theme and shows the current working directory as `/home/ec2-user/`. The left sidebar shows a file tree with files like `..`, `ssh`, `.bash_logout`, `.bash_profile`, and `.bashrc`. The bottom status bar shows the terminal has 53 lines and is connected to a session named 'Bot'.

```
access_log /var/log/nginx/access.log main;  
sendfile on;  
tcp_nopush on;  
keepalive_timeout 65;  
types_hash_max_size 4096;  
  
include /etc/nginx/mime.types;  
default_type application/octet-stream;  
  
# Load modular configuration files from the /etc/nginx/conf.d directory.  
# See http://nginx.org/en/docs/ngx_core_module.html#include  
# for more information.  
include /etc/nginx/conf.d/*.conf;  
  
server {  
    listen 80;  
    listen [::]:80;  
    server_name 18.218.83.45;  
  
    location / {  
        proxy_pass http://127.0.0.1:5000;  
        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;  
    }  
  
    error_page 404 /404.html;  
    location = /404.html {}  
  
    error_page 500 502 503 504 /50x.html;  
    location = /50x.html {}  
}  
-- INSERT --
```

#### 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
```

```
[ec2-user@ip-172-31-6-78 file-share-app]$ sudo vim /etc/nginx/nginx.conf
[ec2-user@ip-172-31-6-78 file-share-app]$ sudo vim /etc/nginx/nginx.conf
[ec2-user@ip-172-31-6-78 file-share-app]$ sudo vim /etc/nginx/nginx.conf
[ec2-user@ip-172-31-6-78 file-share-app]$ sudo systemctl restart nginx
sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[ec2-user@ip-172-31-6-78 file-share-app]$
```

### Purpose/Action:

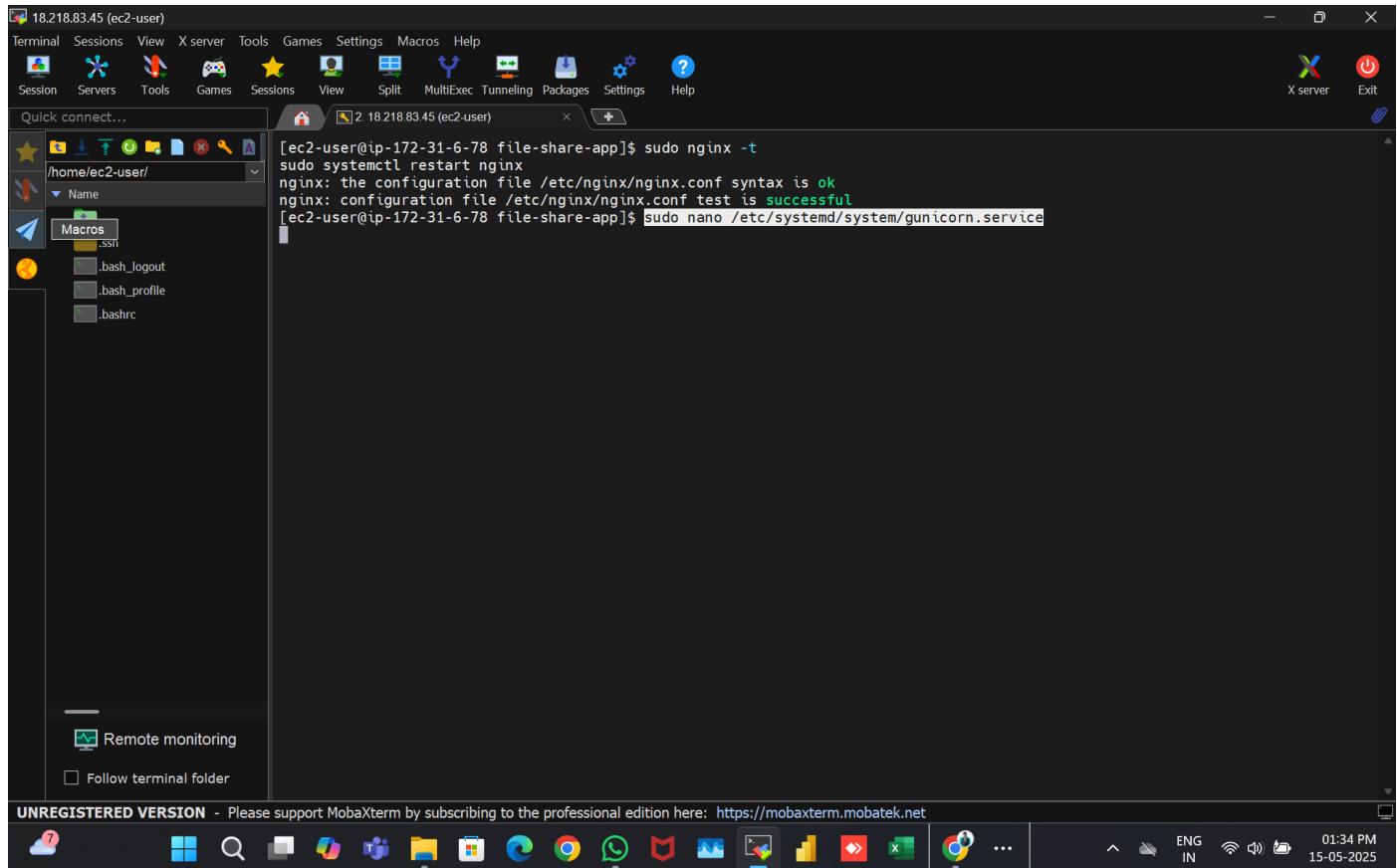
- sudo nginx -t
  - **Tests the Nginx configuration** for syntax errors.
  - If there's an error in /etc/nginx/nginx.conf, this command will tell you **before restarting** the service.
- sudo systemctl restart nginx
  - **Restarts the Nginx service**, applying any changes made to the configuration file.

```
[ec2-user@ip-172-31-6-78 file-share-app]$ sudo nginx -t
sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-172-31-6-78 file-share-app]$
```

## 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

```
GNU nano 8.3 /etc/systemd/system/gunicorn.service Modified
[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
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

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
```

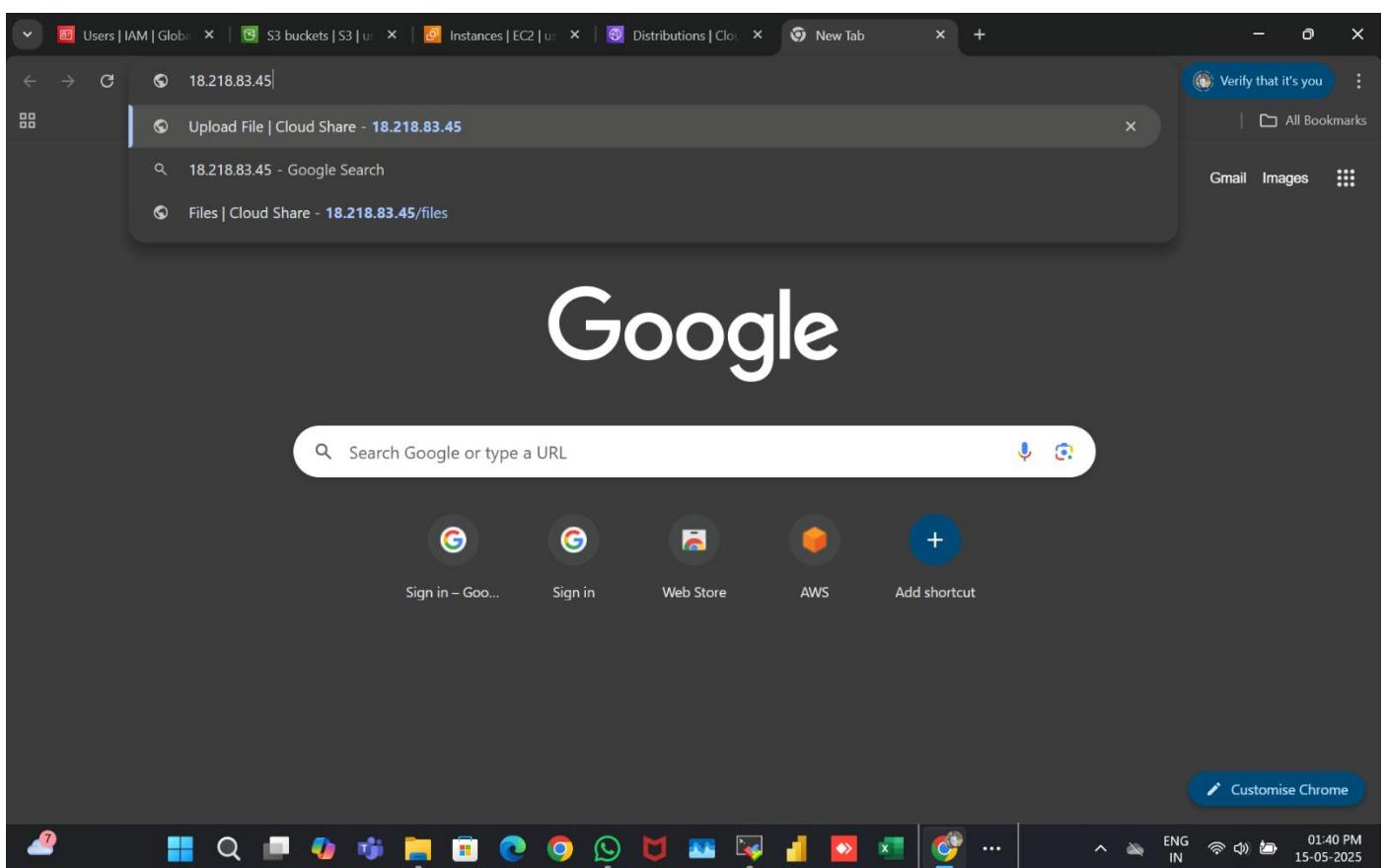
The screenshot shows a MobaXterm interface with the following details:

- Top Bar:** Includes tabs for Terminal, Sessions, View, X server, Tools, Games, Settings, Macros, and Help.
- Toolbar:** Contains icons for Session, Servers, Tools, Games, Sessions, View, Split, MultiExec, Tunneling, Packages, Settings, and Help.
- Right Side:** Includes an X server icon and an Exit button.
- Left Sidebar:** A file browser showing the directory structure of /home/ec2-user/. The contents include .., .ssh, .bash\_logout, .bash\_profile, and .bashrc.
- Terminal Window:** Titled "2 18.218.83.45 (ec2-user)". It displays the following command output:

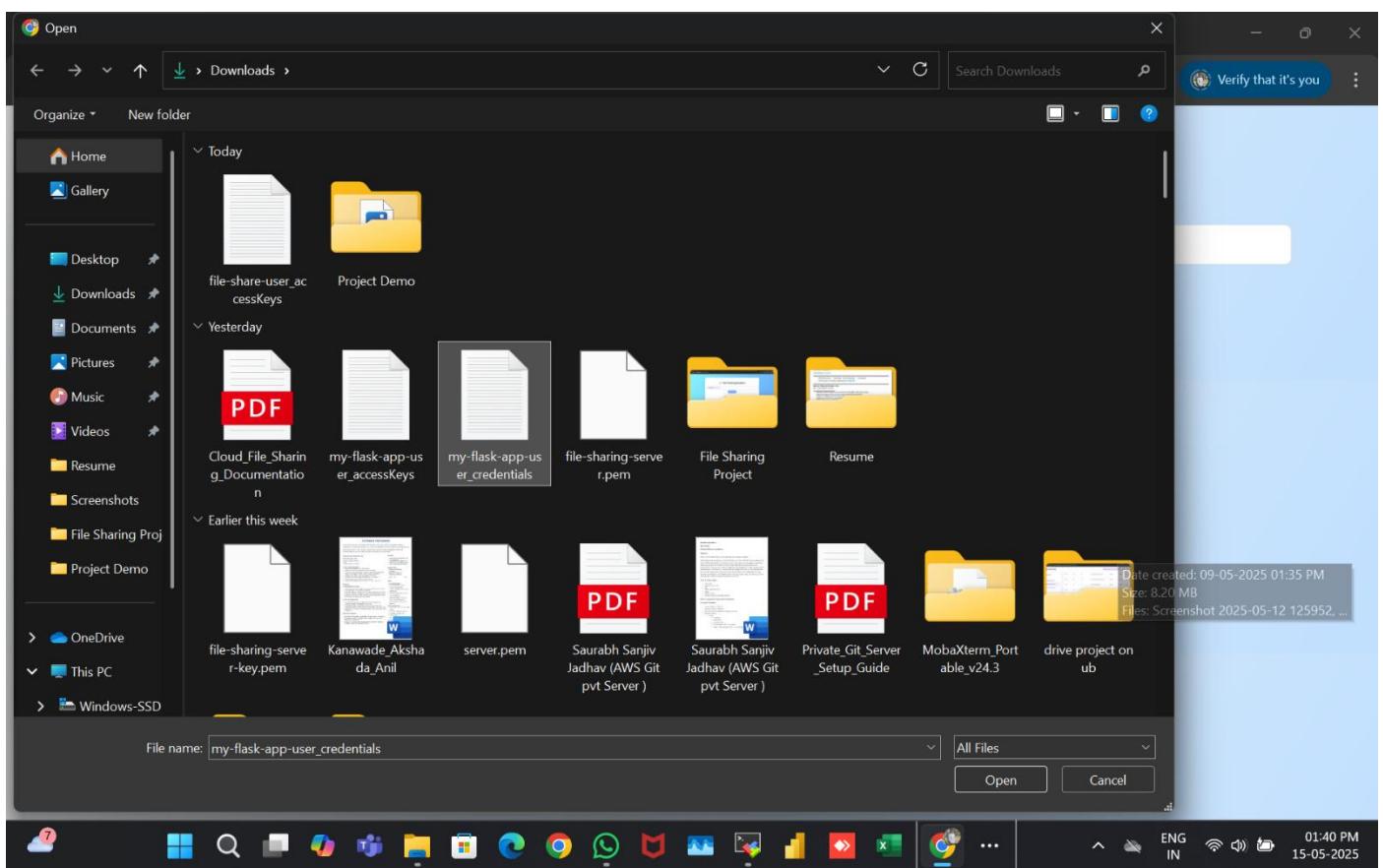
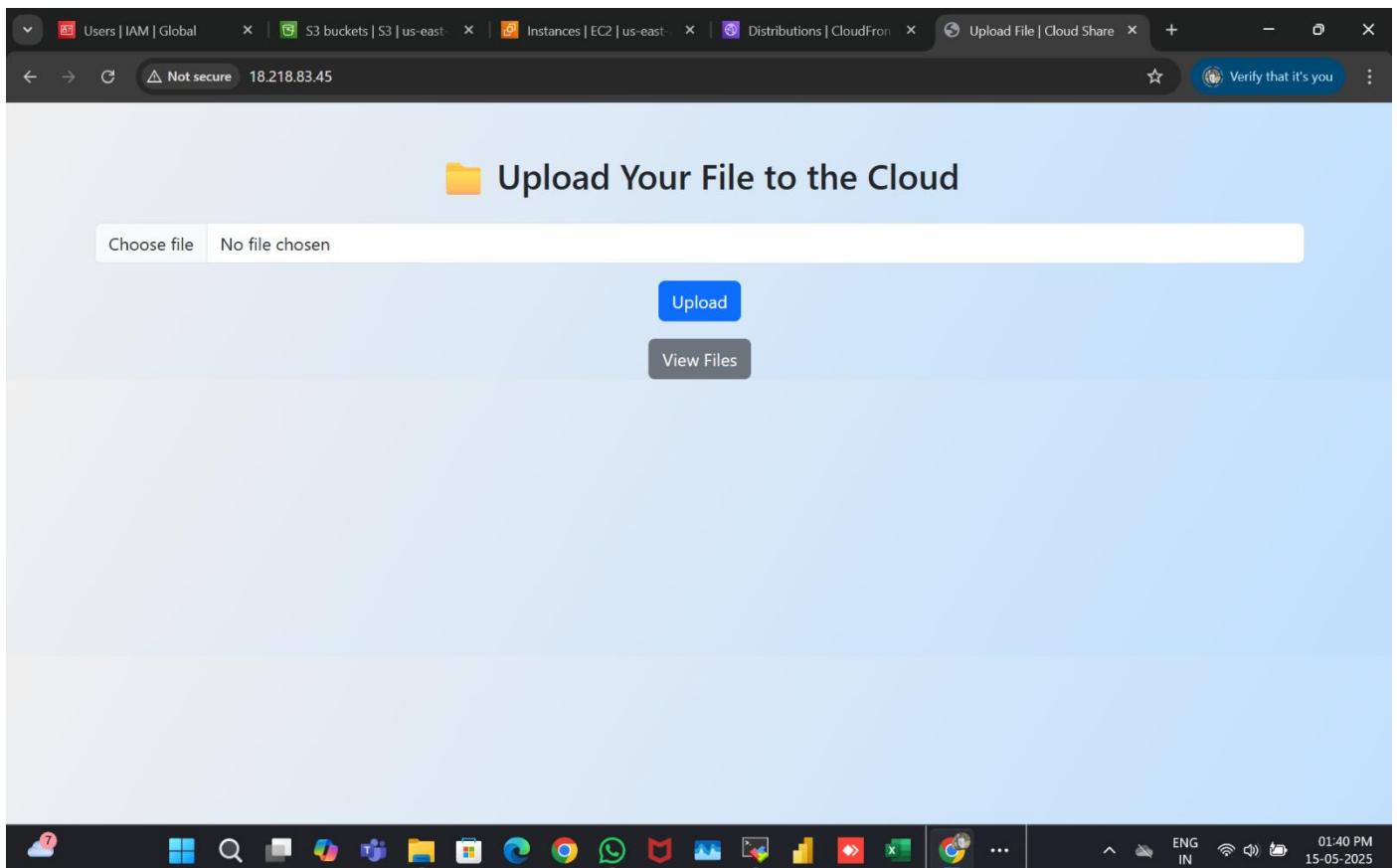
```
[ec2-user@ip-172-31-6-78 file-share-app]$ sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl restart gunicorn
sudo systemctl status gunicorn
● gunicorn.service - Gunicorn instance to serve Flask app
    Loaded: loaded (/etc/systemd/system/gunicorn.service; enabled; preset: disabled)
    Active: active (running) since Thu 2025-05-15 08:09:02 UTC; 104ms ago
      Main PID: 29524 (gunicorn)
        Tasks: 1 (limit: 1111)
       Memory: 7.9M
          CPU: 48ms
        CGroup: /system.slice/gunicorn.service
                └─29524 /usr/bin/python3 /home/ec2-user/.local/bin/gunicorn --workers 3 --bind 127.0.0.1:5000 app:app
```
- Bottom Status Bar:** Shows the date and time (May 15 08:09:02), the IP address (ip-172-31-6-78.us-east-2.compute.internal), the terminal number (system[1]), and the message "Started gunicorn.service - Gunicorn instance to serve F".
- Bottom Icons:** Includes icons for Remote monitoring, Follow terminal folder, and other system status indicators.

Now access your web app at:

<http://18.218.83.45>



Now its Successfully working.



The screenshot shows a web interface titled "Available Files". It has a header with tabs: "Users | IAM | Global", "S3 buckets | S3 | us-east-", "Instances | EC2 | us-east-", "Distributions | CloudFront", and "Files | Cloud Share". The main content area displays two files:

File Name	Download
08f3f2ff-429a-4501-819b-a1ab38cad984-my-flask-app-user_credentials.csv	<button>Download</button>
bcc977ca-3479-4294-8747-326ed0f4da48-file-share-user_accessKeys.csv	<button>Download</button>

Below the table is a button labeled "Upload More". The bottom of the screen shows a Windows taskbar with various icons and system status.

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

The screenshot shows the AWS S3 console at the URL `us-east-2.console.aws.amazon.com/s3/buckets/cloud-files-saurabh?region=us-east-2&bucketType=general&tab=objects`. The left sidebar includes links for EC2, S3, IAM, VPC, Route 53, and Amazon S3. The main area shows the "cloud-files-saurabh" bucket with two objects:

Name	Type	Last modified	Size	Storage class
08f3f2ff-429a-4501-819b-a1ab38cad984-my-flask-app-user_credentials.csv	csv	May 15, 2025, 13:40:39 (UTC+05:30)	124.0 B	Standard
bcc977ca-3479-4294-8747-326ed0f4da48-file-share-user_accessKeys.csv	csv	May 15, 2025, 13:39:37 (UTC+05:30)	99.0 B	Standard

The bottom of the screen shows a Windows taskbar with various icons and system status.

## 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

The screenshot shows the 'Create distribution' wizard on the AWS CloudFront console. The first step, 'Choose the type of distribution that best fits your needs', is displayed. It offers two options: 'Single website or app' (selected) and 'Multi-tenant architecture - New'. The 'Single website or app' option is described as choosing if you have a single app or website. The 'Multi-tenant architecture' option is described as being for multiple domains sharing configurations, common for SaaS providers. Below this, the 'Origin' section is visible, containing fields for 'Origin domain' (set to 'cloud-files-saurabh.s3.us-east-2.amazonaws.com'), 'Origin path - optional' (empty), and 'Name' (also set to 'cloud-files-saurabh.s3.us-east-2.amazonaws.com'). The 'Origin access' tab is selected at the bottom. The browser's address bar shows the URL 'us-east-1.console.aws.amazon.com/cloudfront/v4/home?region=us-east-2#/distributions/create'. The status bar at the bottom right indicates the date and time as '15-05-2025 03:03 PM'.

This screenshot shows the 'Create distribution' wizard on the AWS CloudFront console, identical to the one above but with different input values. The 'Origin domain' field now contains 'cloud-files-saurabh.s3.us-east-2.amazonaws.com'. The 'Origin path - optional' field contains '/18.218.83.45'. The 'Name' field remains 'cloud-files-saurabh.s3.us-east-2.amazonaws.com'. The 'Origin access' tab is still selected. The browser's address bar shows the URL 'us-east-1.console.aws.amazon.com/cloudfront/v4/home?region=us-east-2#/distributions/create'. The status bar at the bottom right indicates the date and time as '15-05-2025 03:04 PM'.

## Now its Working.

The screenshot shows the AWS CloudFront Distributions page. On the left, there's a sidebar with navigation links for CloudFront (selected), Policies, Functions, Static IPs, VPC origins, What's new, SaaS (Multi-tenant distributions, Distribution tenants), Telemetry (Monitoring, Alarms, Logs), and Reports & analytics (Cache statistics). The main content area has a title "Distributions (1) Info" with a "Create distribution" button. It includes a search bar and a dropdown menu for "All distributions". A table lists one distribution: "ECNPES4LXTHF9" (Enabled, Standard Type, d1f4l... Origin, cloud-files-saur). The bottom of the screen shows the AWS navigation bar with CloudShell, Feedback, and various icons, along with copyright information and system status.

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.