Cloud Computing - SEMESTER II

# From Development to Deployment: Building and Implementing a Credit Risk Analysis Model on AWS

BY: Awwal Choudhari (A025), Anil Vhatkar (A004) and Om Muddebihal (A032)

**SVKM'S NMIMS Nilkamal School of Mathematics, Applied Statistics and Analytics, Vile Parle West, Mumbai- 400 056**

---

## INTRODUCTION :

In today's financial landscape, effective credit risk analysis is crucial for institutions to make informed lending decisions and mitigate potential losses. Machine learning has emerged as a powerful tool in this domain, enabling the development of sophisticated models that can analyse vast amounts of data and provide accurate risk assessments. This deployment guide aims to walk you through the process of deploying a credit risk analysis machine learning model on the Amazon Web Services (AWS) cloud platform.

AWS offers a robust and scalable infrastructure for deploying and managing machine learning models, making it an ideal choice for credit risk analysis applications. By leveraging AWS services such as Amazon SageMaker, Amazon Elastic Compute Cloud (EC2), and Amazon Simple Storage Service (S3), you can streamline the deployment process, ensure high availability, and efficiently scale your credit risk analysis model to meet changing demand.

By following this deployment guide, we will cover key concepts, best practices, and technical steps involved in deploying a credit risk analysis model on AWS. From data pre-processing and model training to deployment and integration with AWS services, each section is designed to provide you with a comprehensive understanding of the deployment process.

## Purpose:

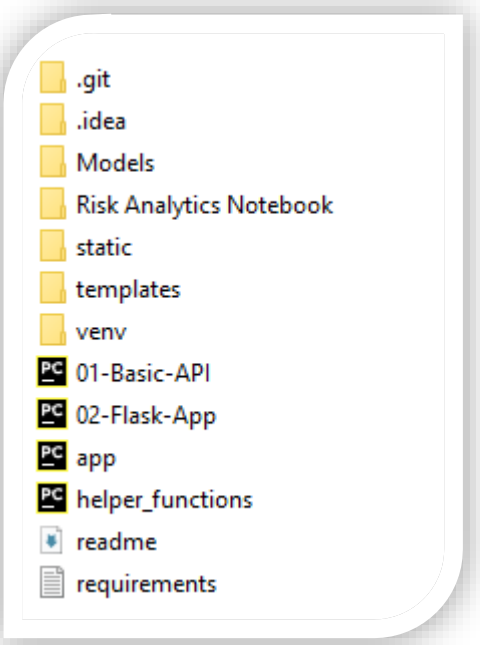
The purpose of this project is to understand the process of deploying on AWS. By following these steps, one can gain hands-on experience with AWS services, deployment procedures, and troubleshooting techniques.

# AWS Services and Other Tools Used:

- Amazon EC2: For provisioning virtual servers to host the website.
- AWS Terminal: To interact with AWS services via the command line interface.
- Putty: For SSH access to the EC2 instance.
- WinSCP: For secure file transfer between the local machine and the EC2 instance.
- Various Python packages: Including Flask, Flask-WTF, and scikit-learn for web application development.

# Implementing a Credit Risk Analysis Model on AWS

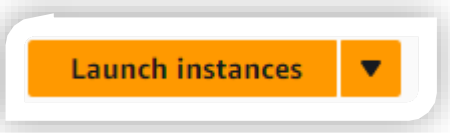
- We have One Machine Learning Project using SVM name is
  **Risk Analytics Deployment**
  Which include these folders like – Static, Templates, Models etc



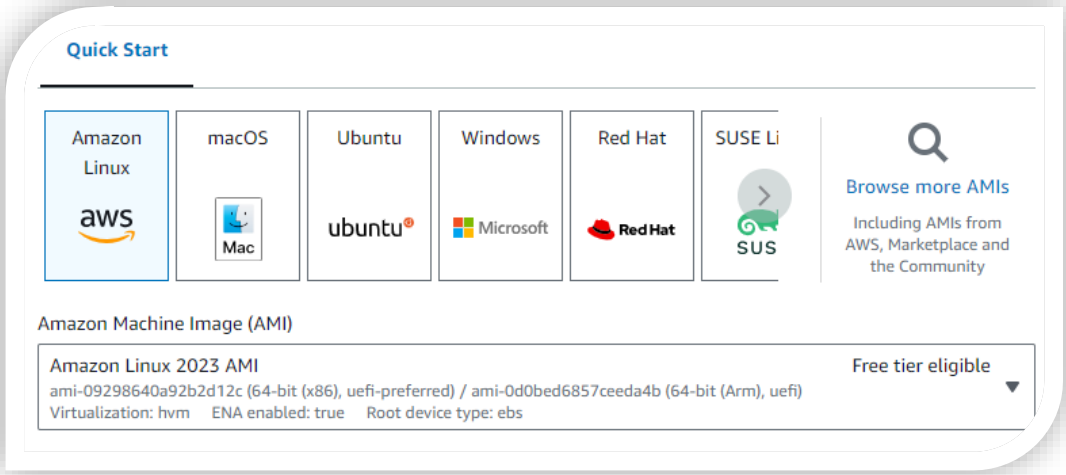

We want to deploy this on Aws Website So How We can ?

Lets Start :-

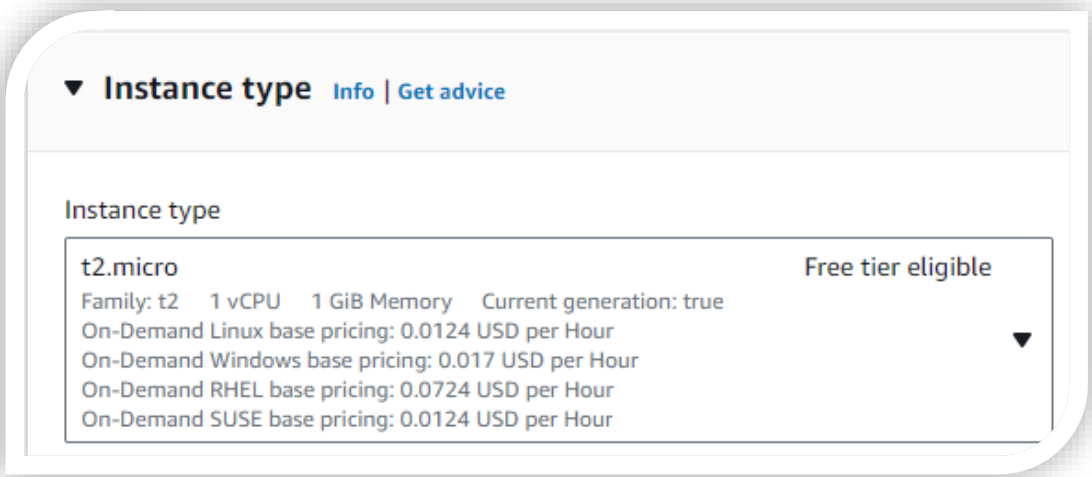
- Login Through Aws Root User (Mumbai Region) Or IAM user.
- We want **Aws terminal** where we will load or Push our ML Code on EC2.
- For Aws terminal Open **EC2** on AWS console
- First We will make New Instances and then **Create Instances**



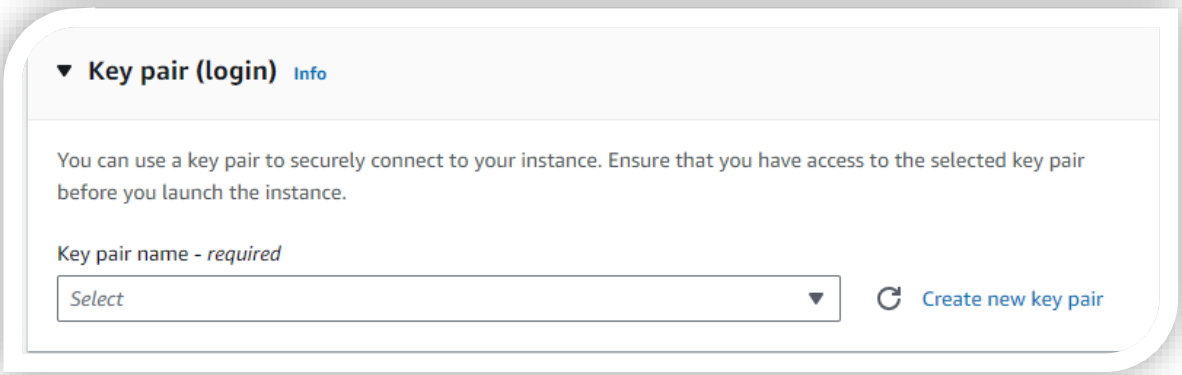

- **Do :- Name :- awwal7 , Os :- Amazon Linux, & select Free Tier Machine, Rest of all keep default and launch.**



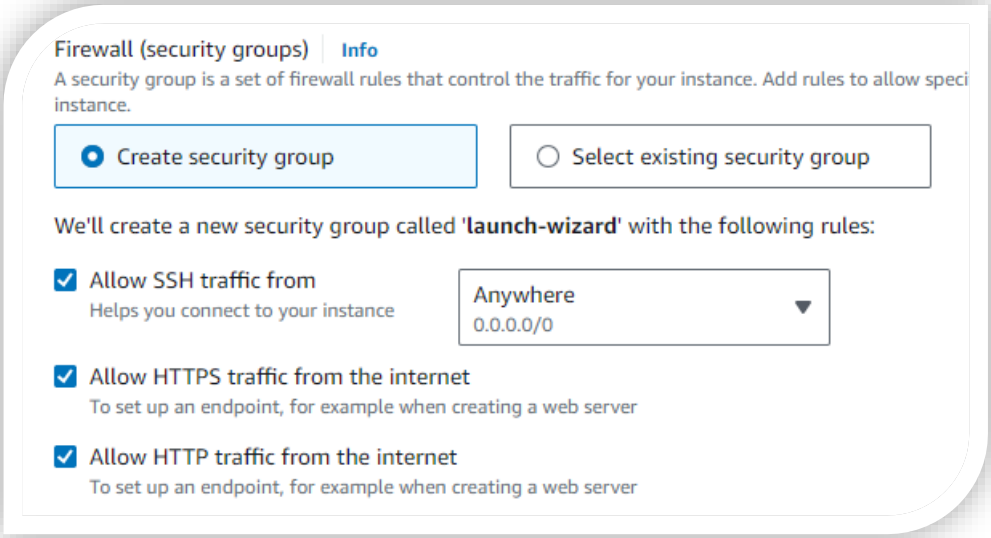

- Select Processor with one CPU (instance type free Charge)
  We can use only **750 Hrs** after this it will charge. **(t2.micro)**



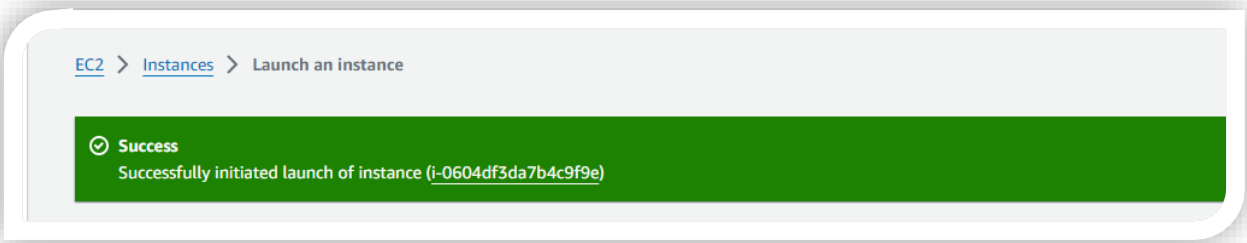

- Create **New Key Pair** for securely connect to our instance

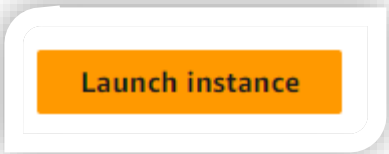



- **Give Key pair name – rest of default..**
  **→ Download – deploy.pem ( Amazon will give u strong password for safety) - Save this in folder where u want.**
  **→ Network Setting :- Select all check box it will give access to all traffics.**




**Storage by default 8 Gb (we can add more volume but that will charge).**
**All Keep default and Launch Instance.**






- Now GO and Check INSTANCE. (Running Instances)
- For connection we will download one application - **putty**

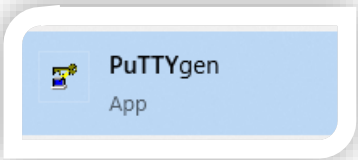
---

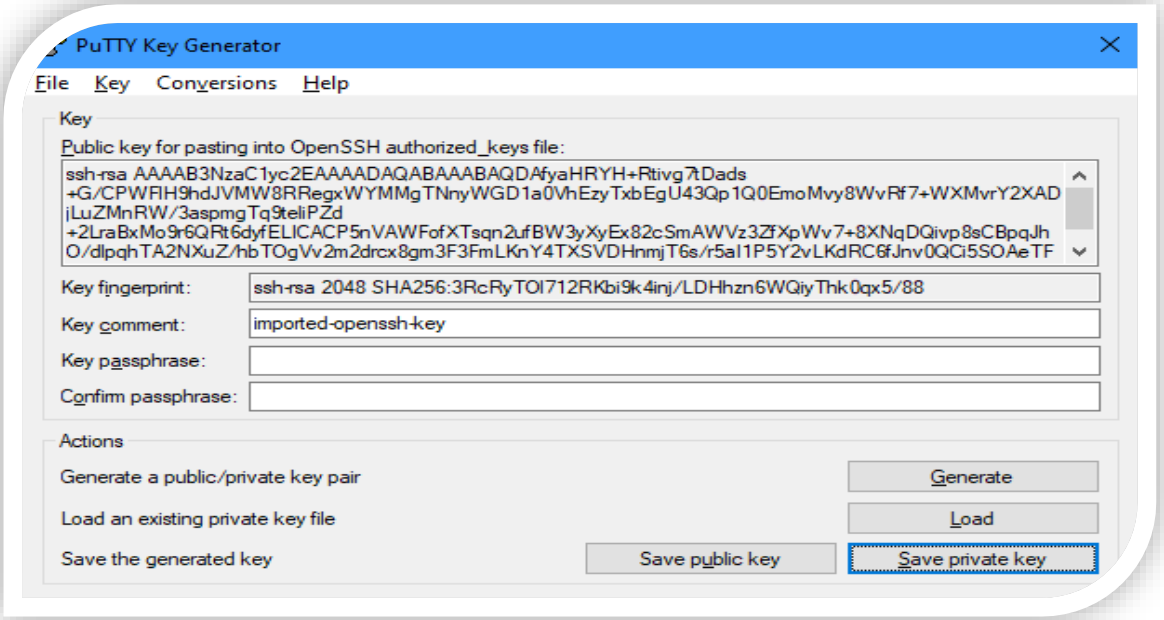
# PUTTY

- Download Putty Normally as we do.
- This Putty will connect Our Laptop and EC2 Instance through SSH path.

## Steps:-

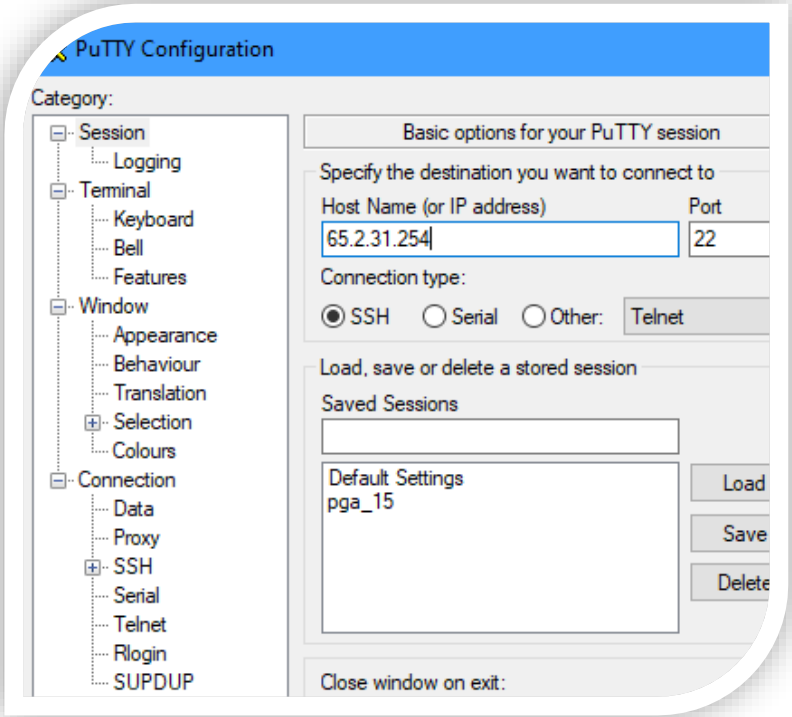
- Open **Putty gen** application
- Load – deploy .pem file – ok – Save Private Key (as rdemo)



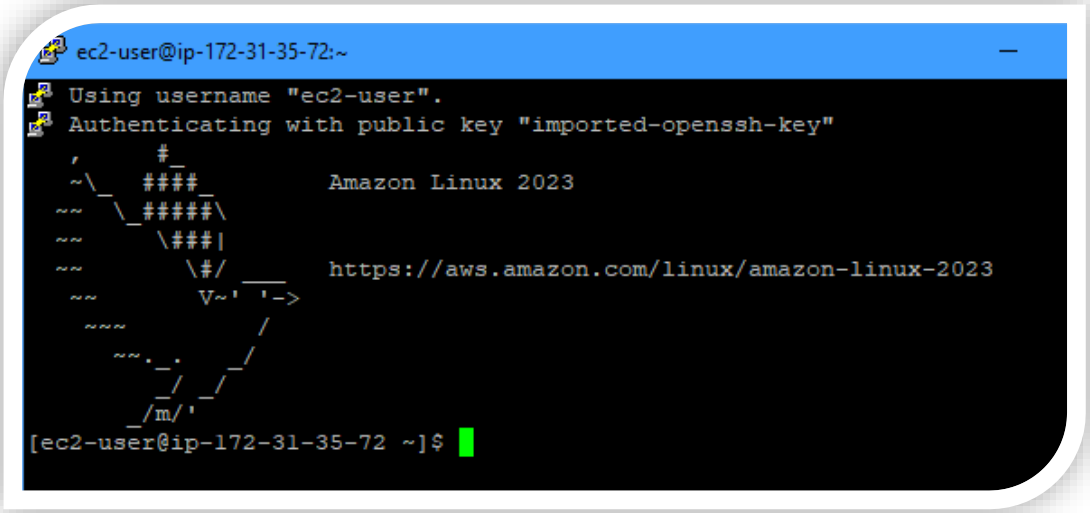



## Open Putty application :-

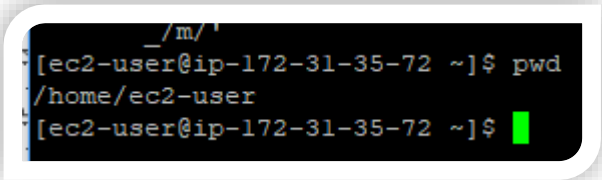
- Host name – Cloud instance (Ec2 ) Public IPv4 copy..
- Connection → SSH → Auth → Credentials (Private Key) .ppk file browse.
- Connection → Data → Auto login username → ( EC2 instance connect, User name) **ec2-user**.
- Session → Save session → type anything (Ds25) save. → Select that and open it. Accept
- **OPEN TERMINAL** after this.
  Look at once
- Local Laptop Terminal (cmd) OR AWS EC2 Terminal




- after all putty operation done.It will Open AWS EC2 Terminal (Accept).




- **In this Terminal We will Push Or Code File which will read by EC2.**
- AWS TERMINAL → write pwd (present working directory)




- Write **Ls** or anything Else it will return nothing Because We didn't upload or push any Ml model .
- Now we will download **Winscp**
- Instance Present in Cloud But our Ml Code is not present on that Aws Cloud so How We can?

---

## **- WinScp (**Windows Secure Copy)

**WinScp software download.**

- secure file transfor protocol software (SFTP) https://winscp.net/eng/index.php

- download next next button check and tick **explorer. Commander next**

WinSCP

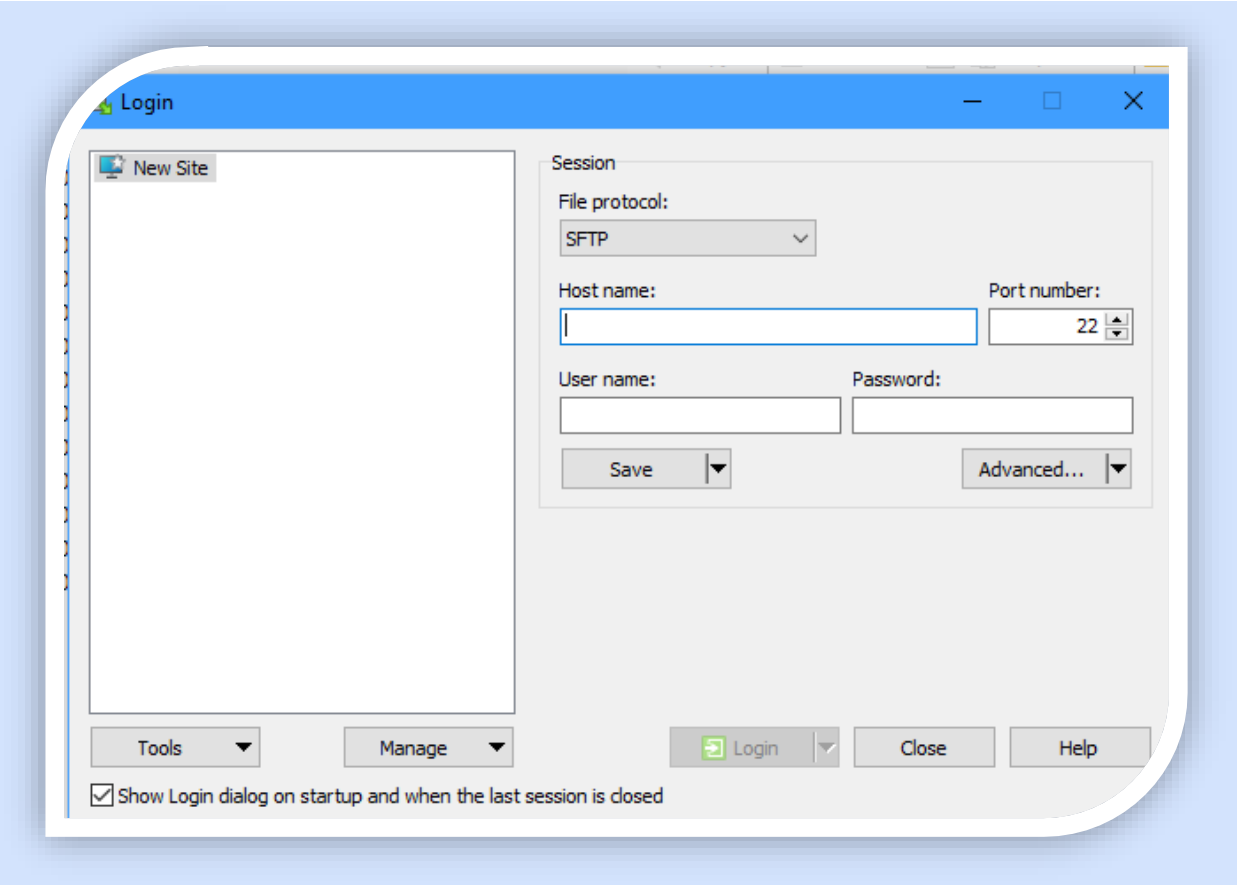
Free SFTP, SCP, S3 and FTP client for Windows

- It allows secure file transfers between the client's local computer and the remote server.

- SFTP basically kia hai ke

e.g:- kisine kuch order kia or wo illegal chiz hai or me kisi ko bhi openly nhi de sakta hu oosko pack krke dena hoga.. i.e hum apne data ko bhi cloud pe upload krege with securely jisse koi hack nhi kr sake ..

**WinScp** Open



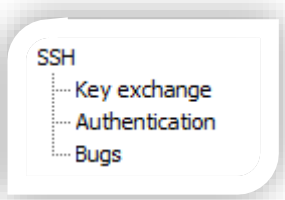

Public IPv4 address

65.2.31.254 |open address

- **Host name → ip v4 instance EC2**

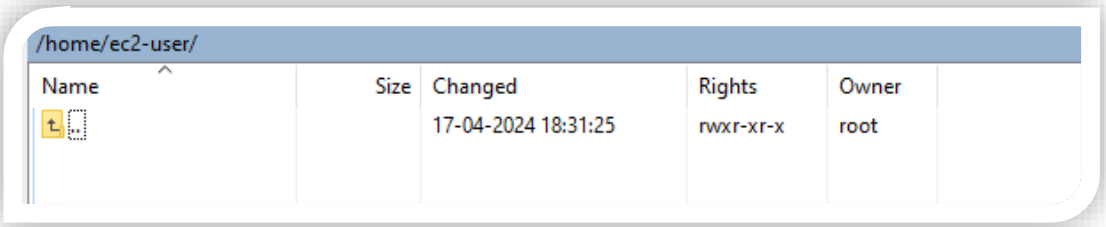
- **User name → ec2-user**

- Password → Advanced → SSH → Authentication , , private key file (upload **.ppk file)** deploy.ppk

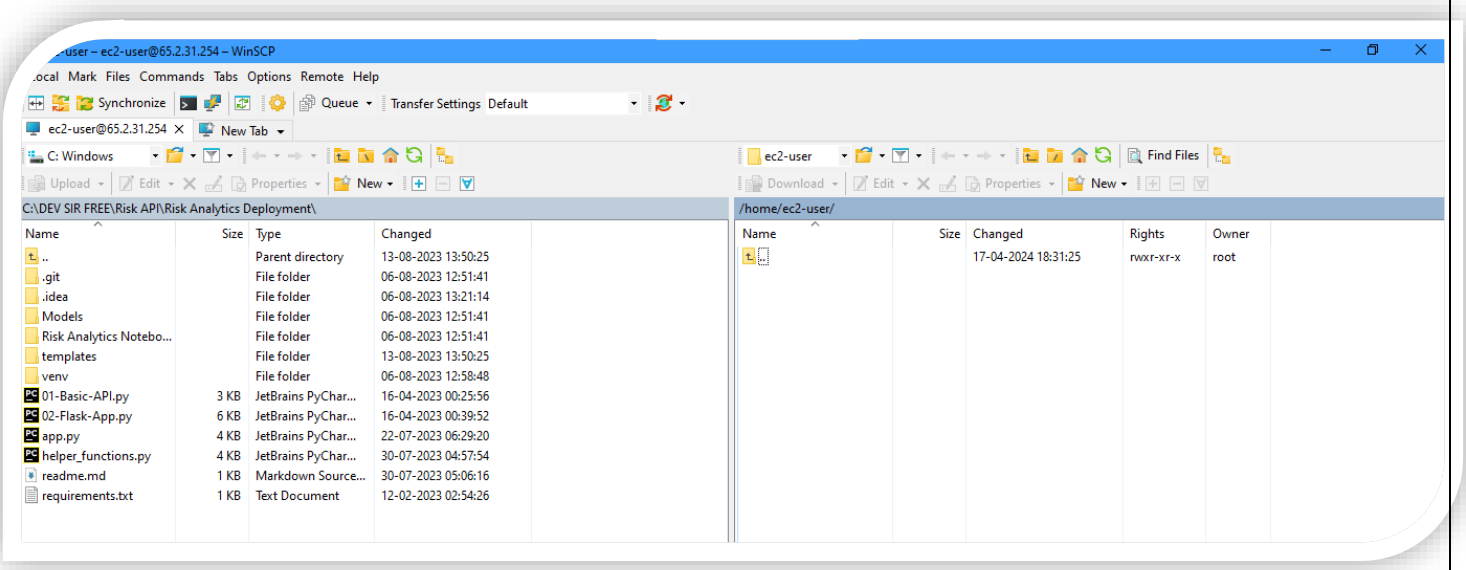



For strong security.

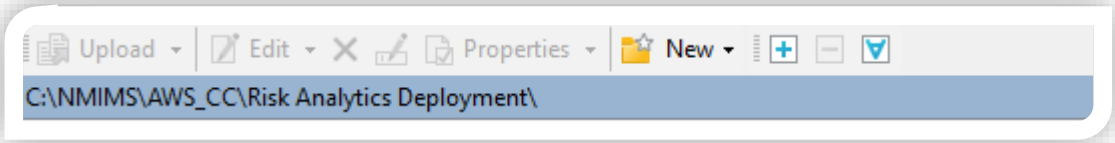
- Login click → Yes & OPEN.



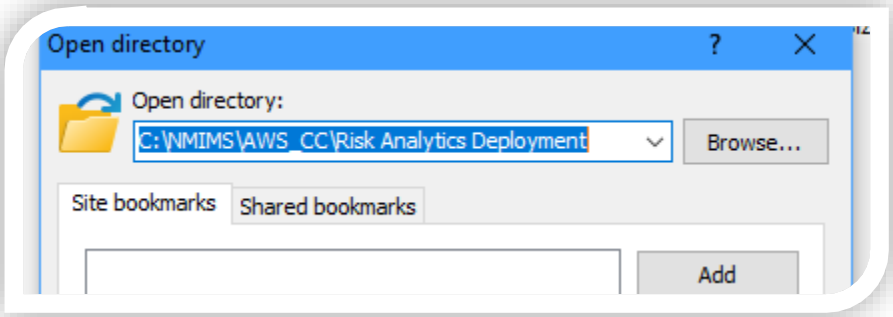

- one side laptop's window explorer another side AWS instance ka explorer.



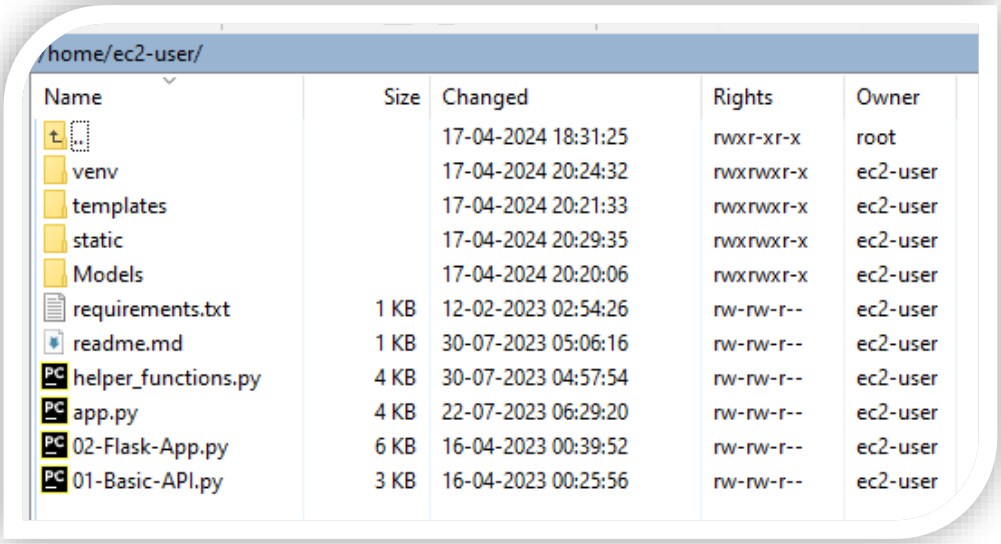
- Copy the Path of Ml model(Risk Analytics) File and paste on Blue part of WinScp.




- Now File Will upload OR u can Paste on Directory




- Load what u want just darg and drop .. These file I upload.




| Name | Size | Changed | Rights | Owner |
|---|---|---|---|---|
| .. | | 17-04-2024 18:31:25 | rwxr-xr-x | root |
| venv | | 17-04-2024 20:24:32 | rwxrwxr-x | ec2-user |
| templates | | 17-04-2024 20:21:33 | rwxrwxr-x | ec2-user |
| static | | 17-04-2024 20:29:35 | rwxrwxr-x | ec2-user |
| Models | | 17-04-2024 20:20:06 | rwxrwxr-x | ec2-user |
| requirements.txt | 1 KB | 12-02-2023 02:54:26 | rw-rw-r-- | ec2-user |
| readme.md | 1 KB | 30-07-2023 05:06:16 | rw-rw-r-- | ec2-user |
| helper_functions.py | 4 KB | 30-07-2023 04:57:54 | rw-rw-r-- | ec2-user |
| app.py | 4 KB | 22-07-2023 06:29:20 | rw-rw-r-- | ec2-user |
| 02-Flask-App.py | 6 KB | 16-04-2023 00:39:52 | rw-rw-r-- | ec2-user |
| 01-Basic-API.py | 3 KB | 16-04-2023 00:25:56 | rw-rw-r-- | ec2-user |

- Now it has been load on Cloud. (req.txt)
  - Minimize it

- OPEN AWS TERMINAL :- **clear** type krke clean terminal → **ls -l** now type this.

- **Open AWS Terminal** :- **Clear** for Clean → **ls -l** for check it's loaded or not.

```
ec2-user@ip-172-31-35-72:~
[ec2-user@ip-172-31-35-72 ~]$ ls -l
total 28
-rw-rw-r--. 1 ec2-user ec2-user 2484 Apr 15  2023 01-Basic-API.py
-rw-rw-r--. 1 ec2-user ec2-user 5168 Apr 15  2023 02-Flask-App.py
drwxrwxr-x. 2 ec2-user ec2-user   56 Apr 17 14:50 Models
-rw-rw-r--. 1 ec2-user ec2-user 3644 Jul 22  2023 app.py
-rw-rw-r--. 1 ec2-user ec2-user 3157 Jul 29  2023 helper_functions.py
-rw-rw-r--. 1 ec2-user ec2-user  845 Jul 29  2023 readme.md
-rw-rw-r--. 1 ec2-user ec2-user  428 Feb 11  2023 requirements.txt
drwxrwxr-x. 2 ec2-user ec2-user   62 Apr 17 14:59 static
drwxrwxr-x. 3 ec2-user ec2-user   96 Apr 17 14:51 templates
drwxrwxr-x. 4 ec2-user ec2-user   68 Apr 17 14:54 venv
[ec2-user@ip-172-31-35-72 ~]$
```

- Now You can see all files has been uploaded.
- Rest of things has to be done in this AWS EC2 terminal.
- **Python** and required libraries are not found so install .

HERE lot of PROBLEM I FACE… In command line

**- sudo yum install python3-pip**

```
ec2-user@ip-172-31-35-72:~
[ec2-user@ip-172-31-35-72 ~]$ python
-bash: python: command not found
[ec2-user@ip-172-31-35-72 ~]$ sudo yum install python3-pip
Last metadata expiration check: 2:22:09 ago on Wed Apr 17 13:01:42 2024.
Dependencies resolved.
================================================================================
 Package                                Architecture                    Versio
================================================================================
Installing:
 python3-pip                            noarch                          21.3.1
Installing weak dependencies:
 libxcrypt-compat                       x86_64                          4.4.33

Transaction Summary
================================================================================
Install  2 Packages

Total download size: 1.9 M
Installed size: 11 M
Is this ok [y/N]: ^COperation aborted.
[ec2-user@ip-172-31-35-72 ~]$
```

→ **click yes then it will install pip and python 3 both..**

→ **Check installation done or not**

**Python3**

```
Complete!
[ec2-user@ip-172-31-35-72 ~]$ python3
Python 3.9.16 (main, Sep  8 2023, 00:00:00)
[GCC 11.4.1 20230605 (Red Hat 11.4.1-2)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

→ **Old python version doesn't exist**

→ **exit()**

→ **pip3 install -r requirements.txt**

```
[ec2-user@ip-172-31-35-72 ~]$ pip3 install -r requirements.txt
Defaulting to user installation because normal site-packages is not writeable
Collecting astor==0.8.1
  Downloading astor-0.8.1-py2.py3-none-any.whl (27 kB)
Collecting certifi==2020.4.5.1
  Downloading certifi-2020.4.5.1-py2.py3-none-any.whl (157 kB)
     |████████████████████████████████| 157 kB 9.4 MB/s
Collecting click==7.1.1
  Downloading click-7.1.1-py2.py3-none-any.whl (82 kB)
     |████████████████████████████████| 82 kB 2.0 MB/s
Collecting Flask==1.1.2
  Downloading Flask-1.1.2-py2.py3-none-any.whl (94 kB)
     |████████████████████████████████| 94 kB 6.3 MB/s
Collecting Flask-WTF==0.14.3
  Downloading Flask_WTF-0.14.3-py2.py3-none-any.whl (13 kB)
```

→ `ERROR: No matching distribution found for grpcio==1.28.1`

**Grapcio version edit u can.**

- **Red warning bcos of version. (WinScp Requirement click and edit version or remove == )**

**→ Run again in terminal requirement code.**

```
ERROR: No matching distribution found for numpy==1.18.2
```

**Same error for Numpy.**

**Same for this also..**

```
ERROR: No matching distribution found for scipy==1.4.1
```

**Ctrl + S** **after editing save**

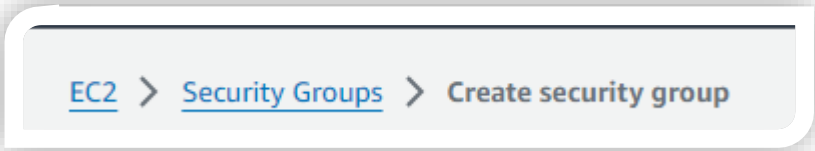
- **Our Ml code run on python 3.7 version but this AWS Terminal using version 3.9.**

---

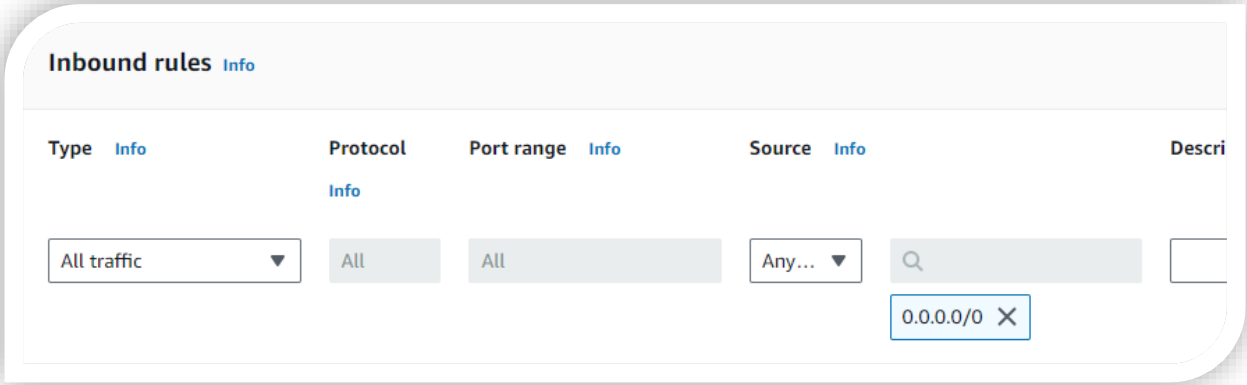
## - How to solve these error just we have seen above.

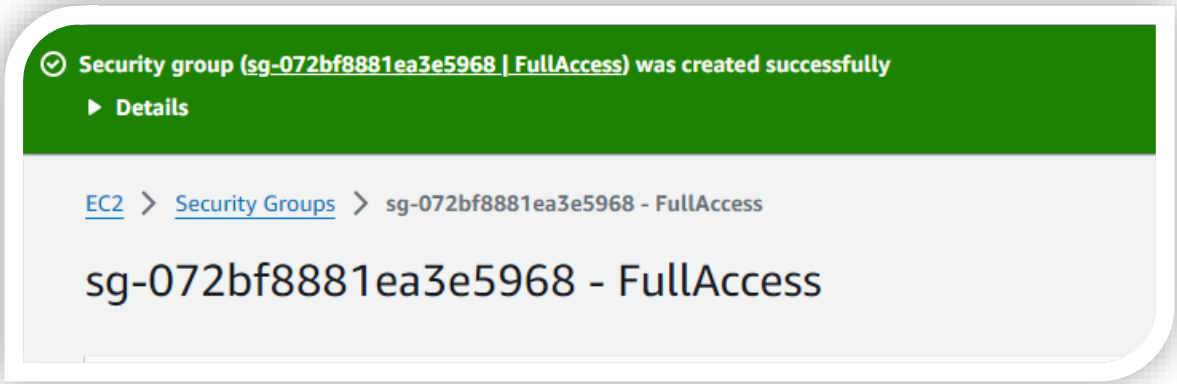
## - Solution : Security group


**→ Open EC2 Instances → Open Security group**




→ **FullAccess → Inbound rules (All traffics, Anywhere ipv4)**

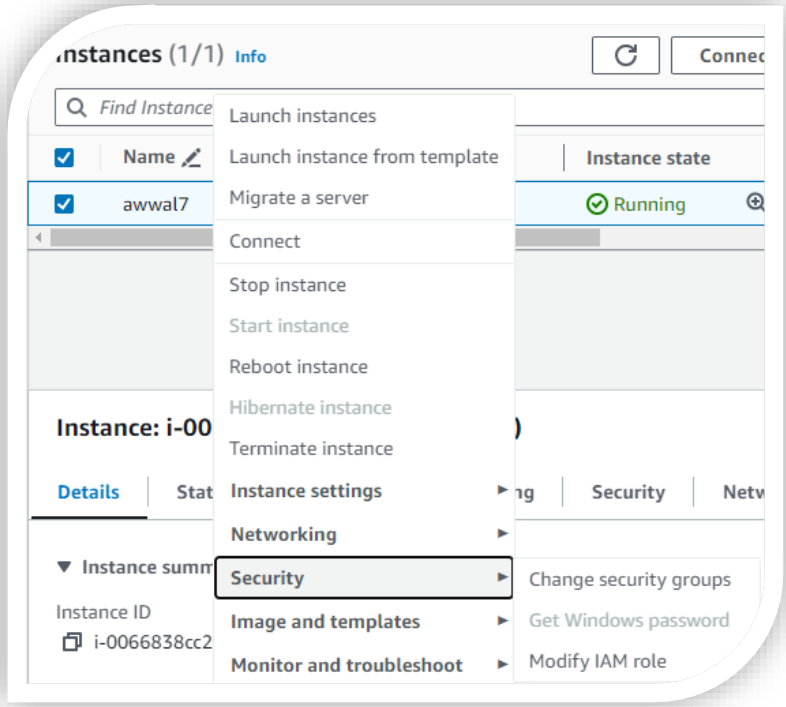



- It will decide which Network can access our website.

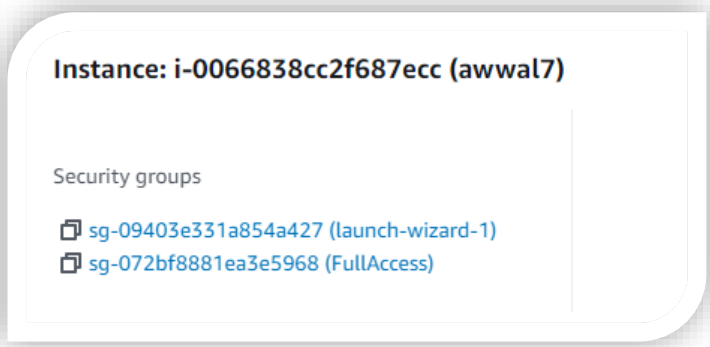
→ Now SECURITY group link with EC2 instances.

→ right click on instance → Change security group




→ **FullAccess which we create – Load and save.**

→ **Select Instance and check Security has loaded or not security FullAccess ?**




## Error Solve

→ **Go on TERMINAL → Error was coming bcos Requirements file was some issue or wrong.**

→ **python3 app.py** → error aayega- **pip3 install flask_wtf**

→ **run again python3 app.py**

→ **pip3 install scikit-learn**

→ Running fast bcos this running on AWS platform

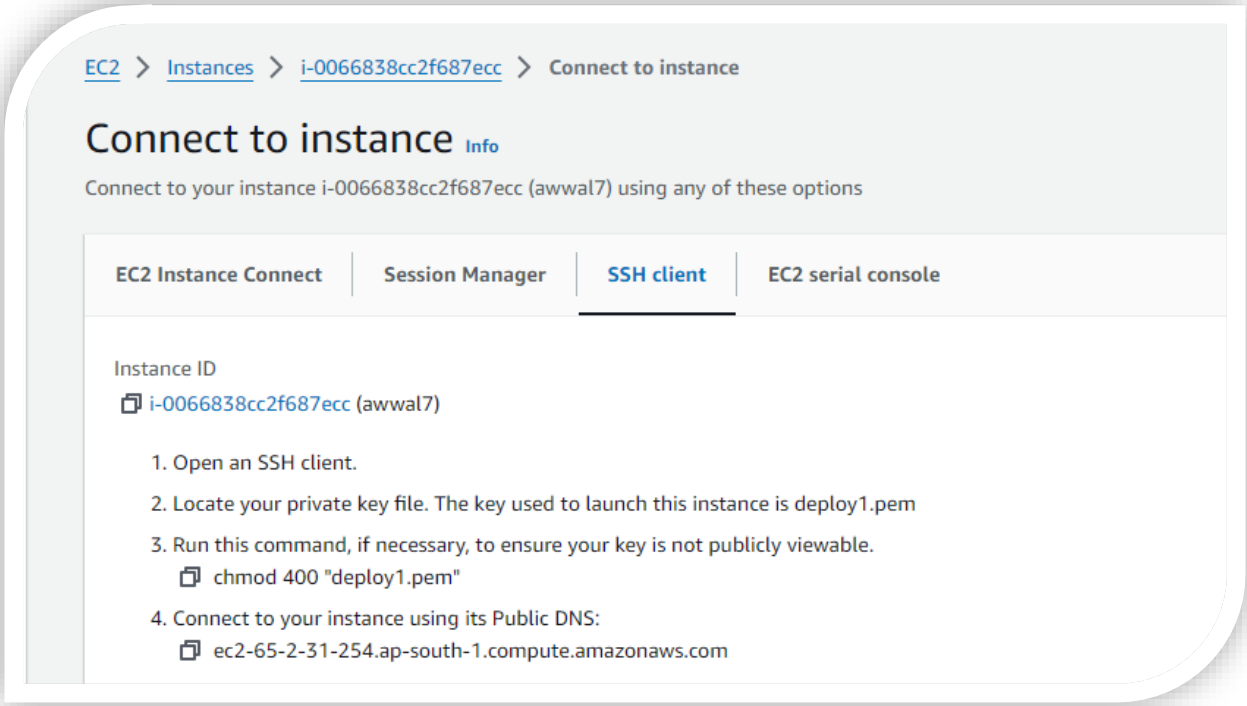
→ python3 app.py  again run

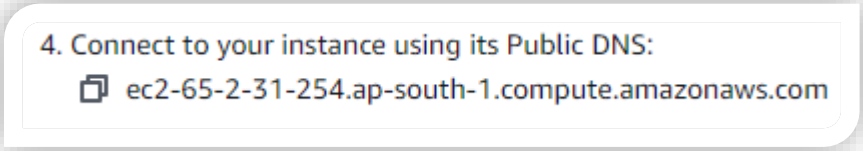
**Now server has been RUN.**

→

```
 warnings.warn(
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

→ EC2 instance connect for deployment



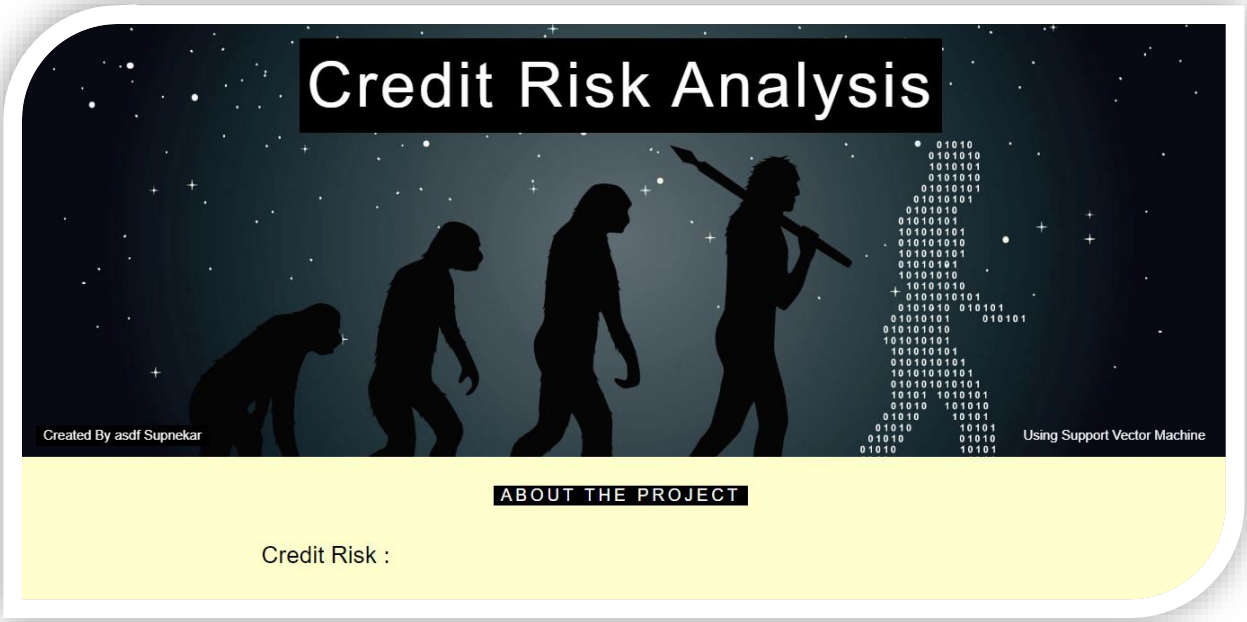

→

**Copy and run on browser :8080 enter**

http://ec2-65-2-31-254.ap-south-1.compute.amazonaws.com:8080/

**→ We can change our domain but it will charge.**

**→ for Close Close Server terminal Ctrl + C stop.**




## AWS Terminal Commands:

- pwd: Print the current working directory.
- clear: Clear the terminal screen.
- ls -l: List files in the current directory.
- sudo yum install python3-pip: Install pip for Python3.
- python3: Check if Python3 is installed.
- exit(): Exit Python3 interpreter.
- pip3 install -r requirements.txt: Install necessary libraries from requirements file.
- python3 app.py: Run the Flask application.
- Ctrl + C: Terminate the server.

# Problems Faced:

During the deployment process, several challenges were encountered, including:

- Dependency issues with Python packages.

```
ERROR: No matching distribution found for numpy==1.18.2
ERROR: No matching distribution found for grpcio==1.28.1
ERROR: No matching distribution found for scipy==1.4.1
```

- Configuration errors in security groups.
- Compatibility issues with Python versions.
- Troubleshooting SSH connections and file transfers.

# Feasibility:

Deploying a website on Amazon Web Services (AWS) is not only feasible but also highly practical. The technical steps outlined, including provisioning EC2 instances, transferring files, resolving dependencies, and running the website application, align perfectly with AWS services such as EC2, S3, and RDS. This indicates our technical requirements can be effectively met within the AWS ecosystem. Additionally, while our initial focus may be on deploying the website on a single EC2 instance, AWS's scalability features, such as auto-scaling groups and load balancers, offer the flexibility to accommodate future growth in website traffic, ensuring scalability and optimal performance.

Furthermore, this project emphasizes cost optimization by leveraging AWS's free tier resources and pay-as-you-go pricing model. By effectively managing costs and scaling resources as needed, we ensure that our deployment on AWS remains cost-effective. Moreover, while our project primarily focuses on the technical aspects of deployment, AWS offers robust security features and compliance certifications that can be integrated into our deployment process to ensure adherence to industry standards for security and compliance. By leveraging the flexibility, support resources, and security features provided by AWS, our project lays a solid foundation for successful website deployment on the platform.

# Future Work:

In order to further optimize the deployment process and mitigate any challenges encountered during deployment, there are several avenues for future work:

- **Automation with AWS CloudFormation or Elastic Beanstalk:** Automating deployment tasks using AWS CloudFormation or Elastic Beanstalk can streamline the deployment process, reduce manual errors, and ensure consistency across environments. By defining infrastructure as code, future deployments can be executed more efficiently and with greater reliability.

- **Integration of Continuous Deployment Pipelines with AWS CodePipeline:** Implementing continuous deployment pipelines using AWS CodePipeline enables automated testing, build, and deployment of code changes to the website. This approach facilitates rapid iteration and deployment of updates, ensuring a seamless and agile development process.

- **Implementing Monitoring and Scaling Solutions:** Enhancing monitoring and scaling solutions for the deployed website is essential for maintaining performance, reliability, and cost efficiency. By implementing monitoring tools such as Amazon CloudWatch and integrating auto-scaling policies, the website can dynamically adjust resources based on demand, ensuring optimal performance during peak traffic periods and cost optimization during idle times.

# Conclusion:

Deploying a website on AWS necessitates thorough planning, meticulous configuration, and adept troubleshooting skills. By meticulously following the steps and proactively addressing encountered challenges, users can gain invaluable insights into cloud deployment practices and harness the full potential of AWS services. With a commitment to continuous learning and improvement, deploying websites on AWS can evolve into a more efficient and scalable process, catering to diverse use cases and driving business success in the dynamic landscape of cloud computing.

# References:


- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

- https://winscp.net/eng/index.php

- https://docs.aws.amazon.com/whitepapers/latest/overview-deployment-options/aws-deployment-services.html