Saurabh Sanjiv Jadhav

AWS Project.

A Private Git Server on Ubuntu

Objective:

Deploy a self-hosted Git server using Gitea on an Ubuntu machine.

This project involves setting up a private Git server on an AWS EC2 Linux instance with secure SSH authentication, user access control, and repository management, enabling a secure and private environment for team-based software development and code collaboration. The server will feature a graphical user interface (GUI) for ease of use and administration. Additionally, a custom domain (saurabhcloud.fun) will be configured to provide easy access, and an optional Gitea web interface will be integrated for visual repository management. For added security, SSL encryption using Let's Encrypt will be implemented to ensure encrypted access over HTTPS.

Tools & Technologies:

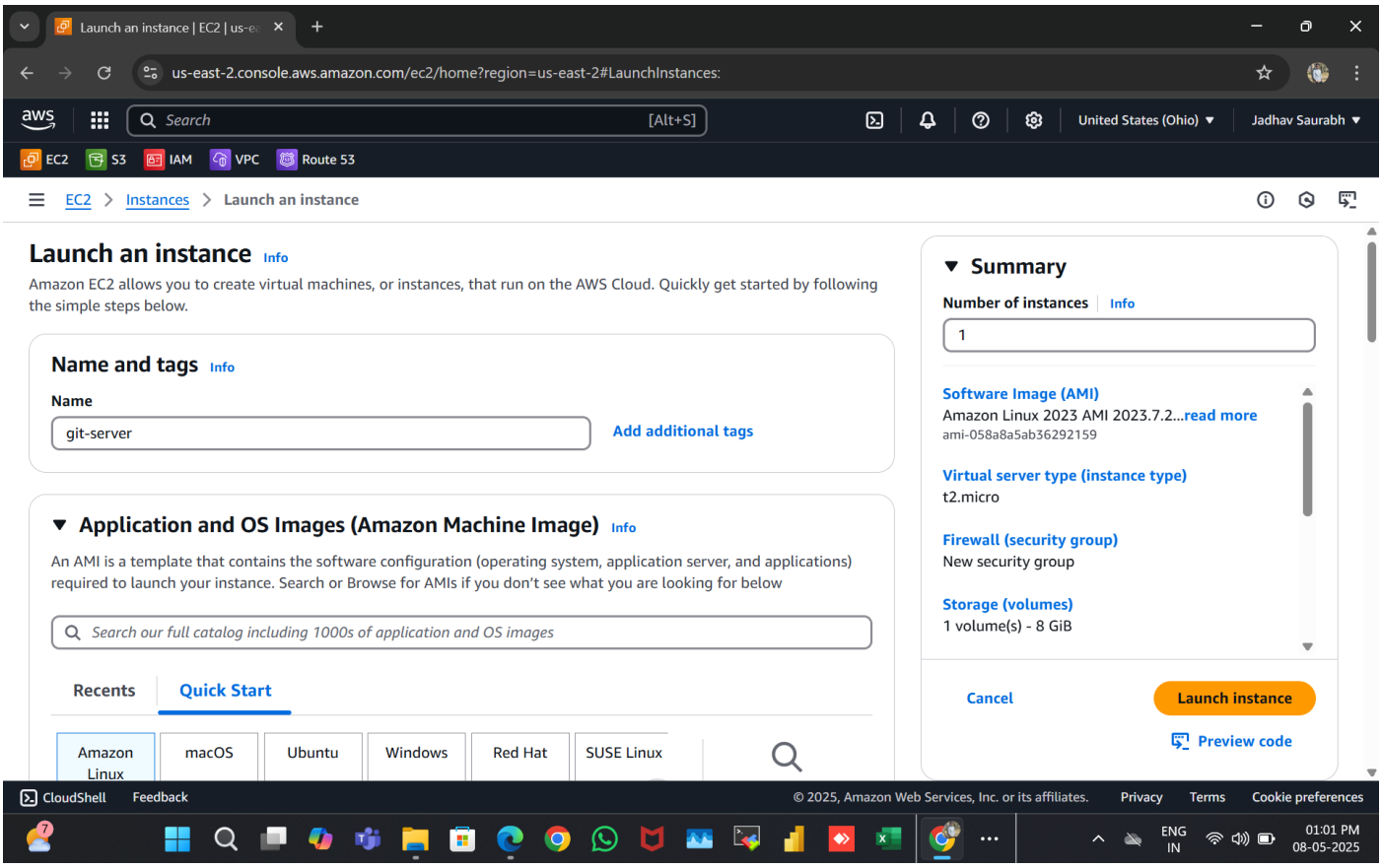
- AWS EC2
- Ubuntu 22.04 LTS
- Git
- Gitea (Optional GUI)
- Nginx
- Certbot (SSL)
- Custom Domain: saurabhcloud.fun

Part 1: Launch and Connect to EC2 Instance

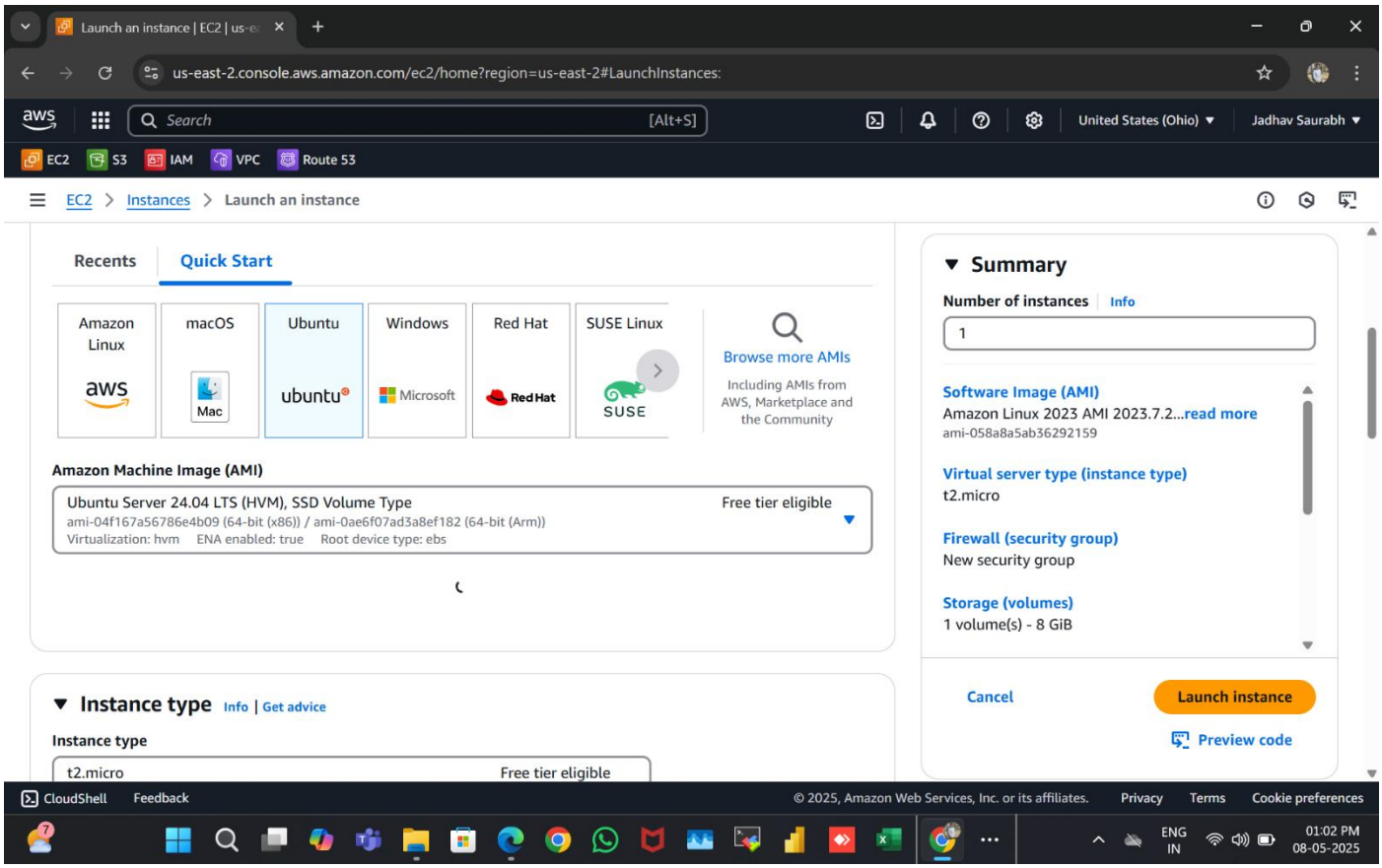
1. Launch EC2 Instance:

- AMI: Ubuntu 22.04 LTS
- Instance Type: t2.medium
- Key Pair: Use an existing or create a new one
- Security Group:
    - Allow:
        - 22 (SSH)
        - 80 (HTTP)
        - 443 (HTTPS)
        - 587 (Custom TCP – for email)
        - 3000–11000 (Custom TCP – for web apps like Gitea)

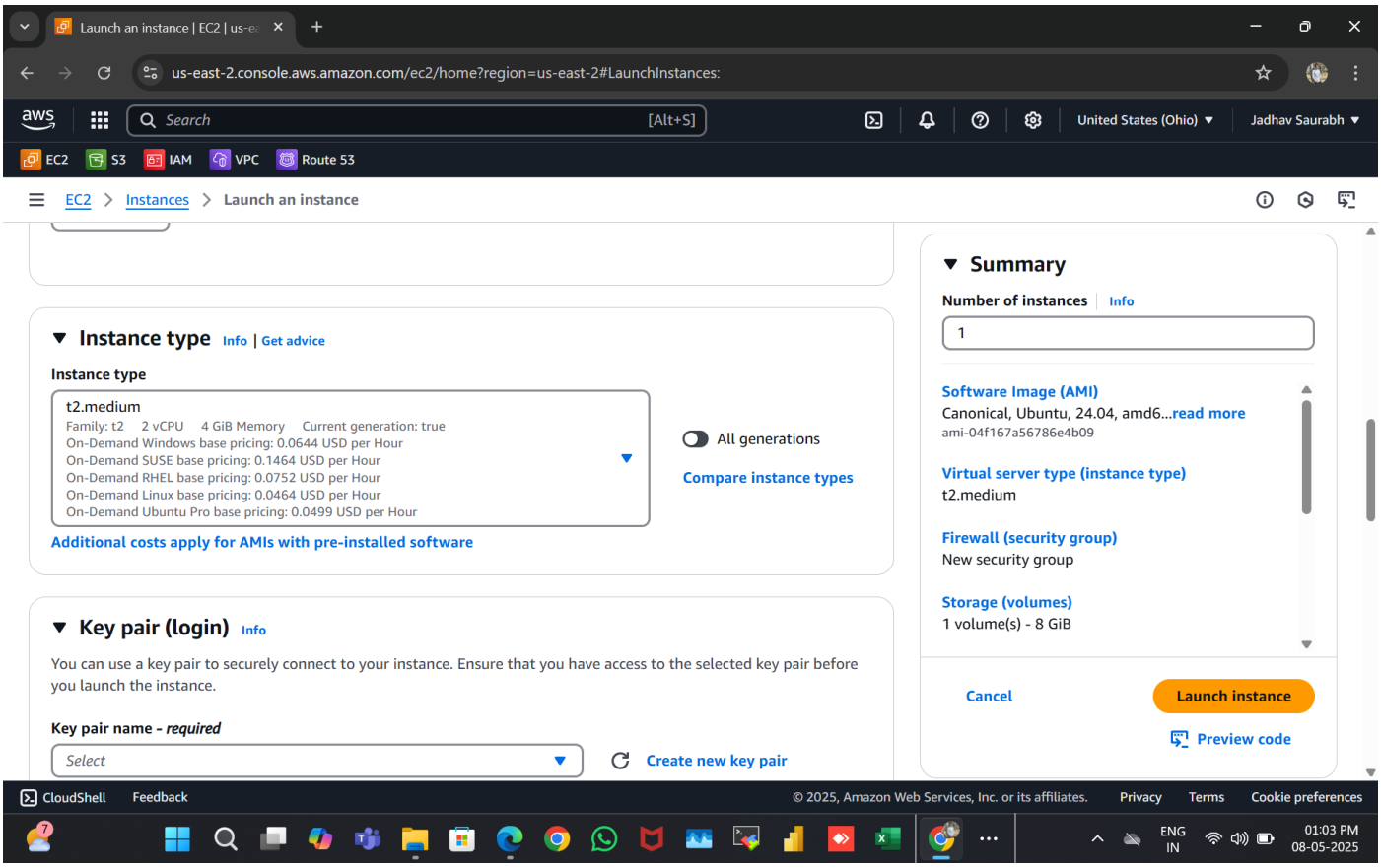
Assign EC2 Instance Name



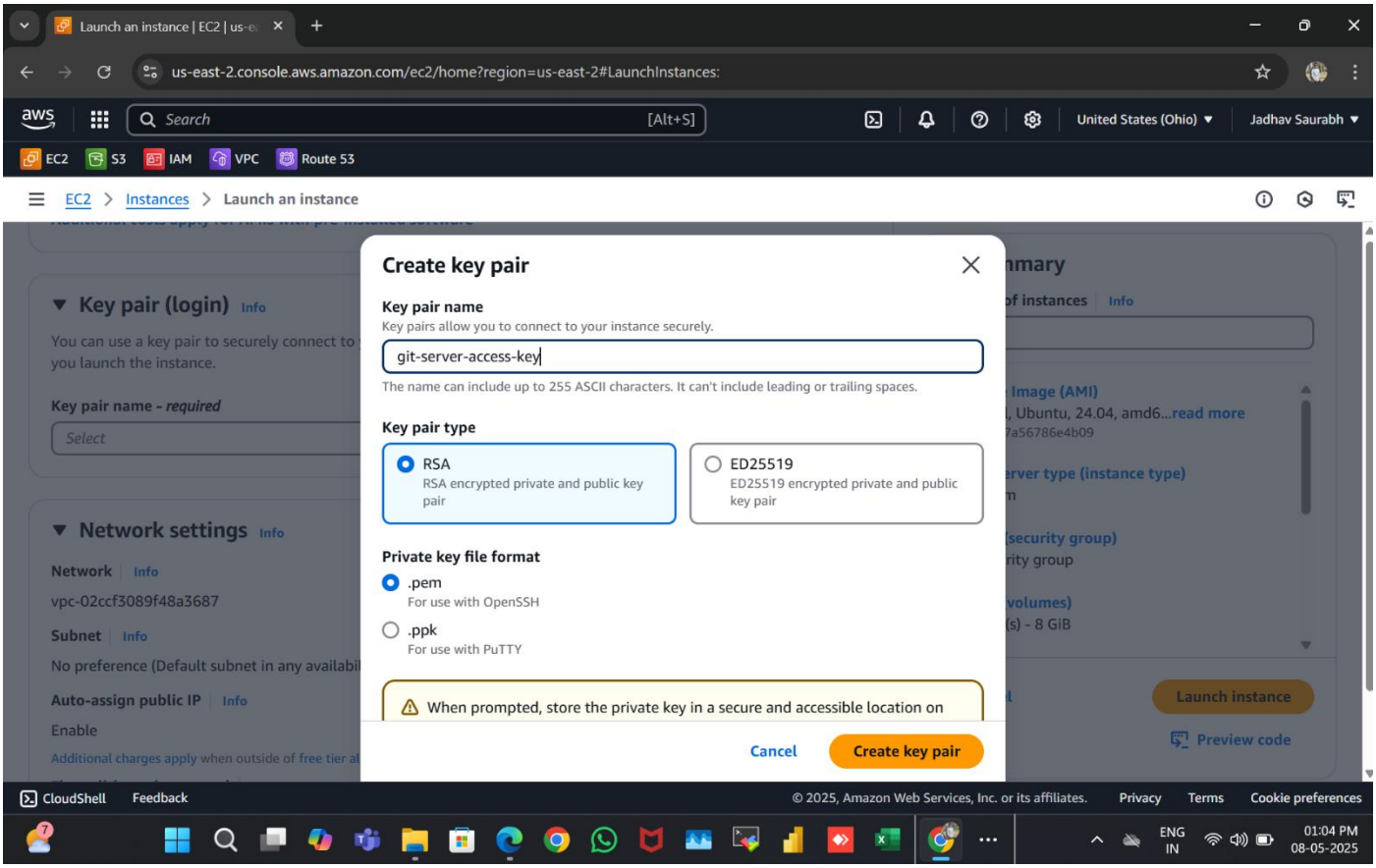
Select AMI According to Your configuration .



Select the Instance Type According to Your configuration .

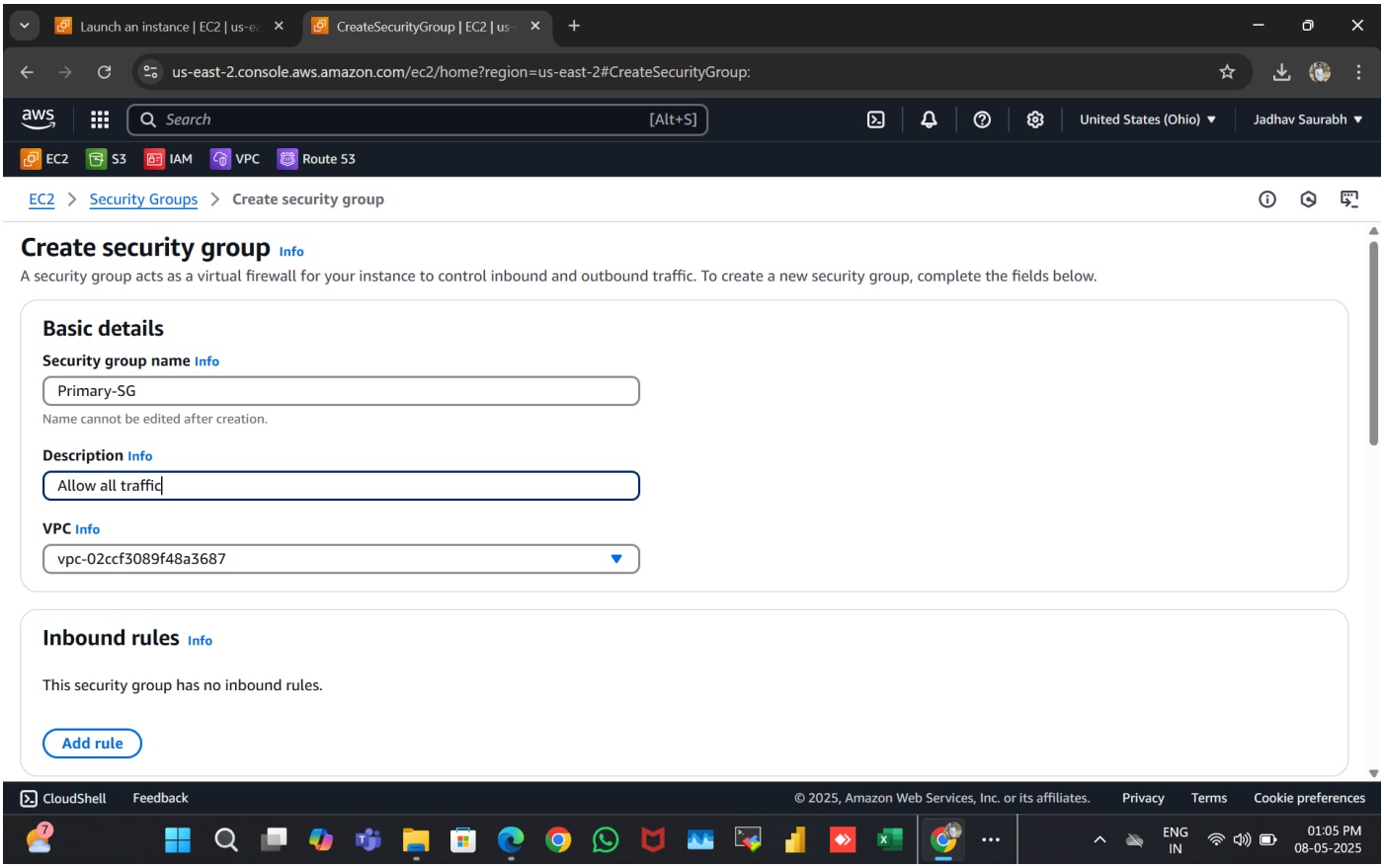


Create key Pair .

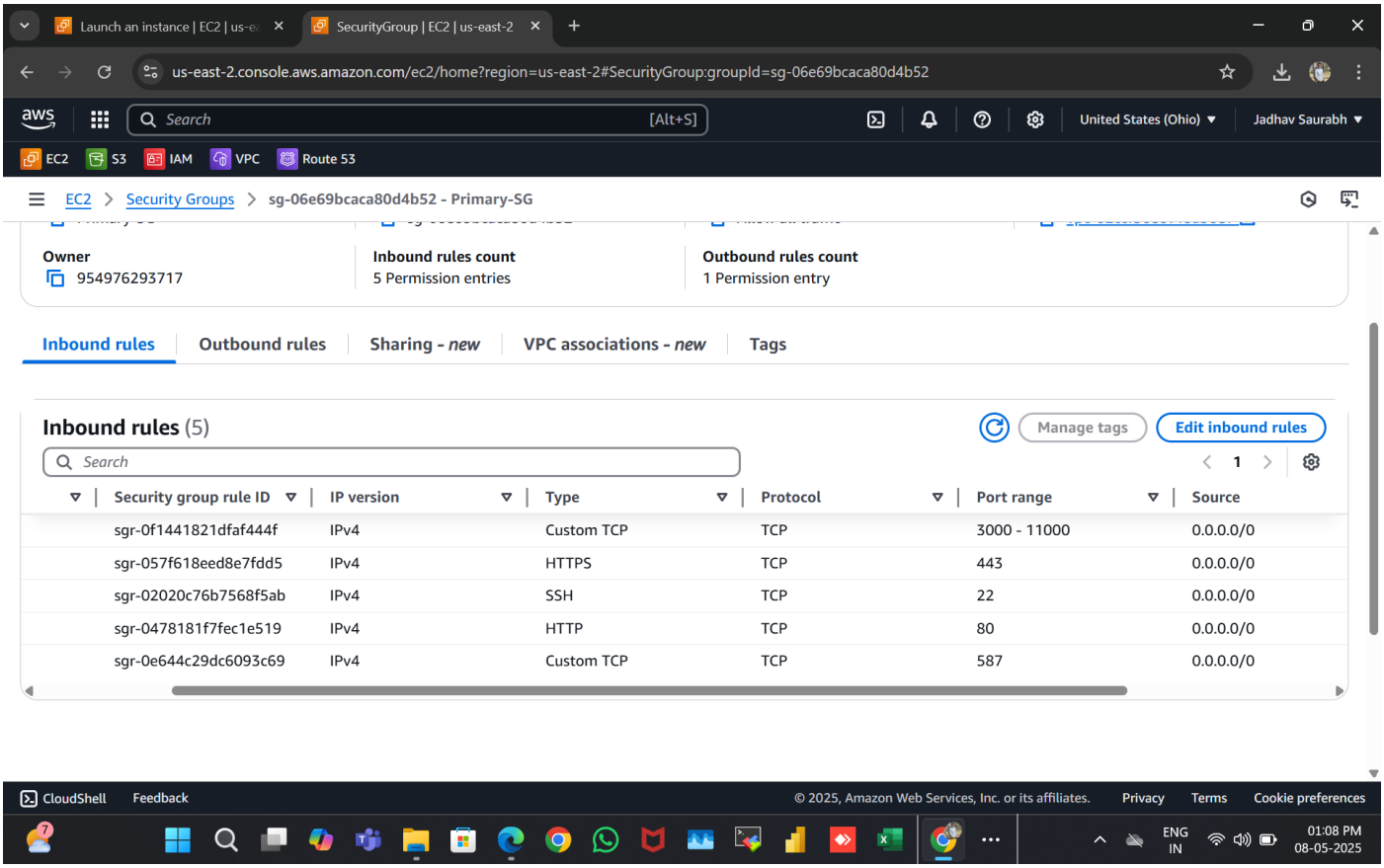


Create Security Group or Use existing Security Group

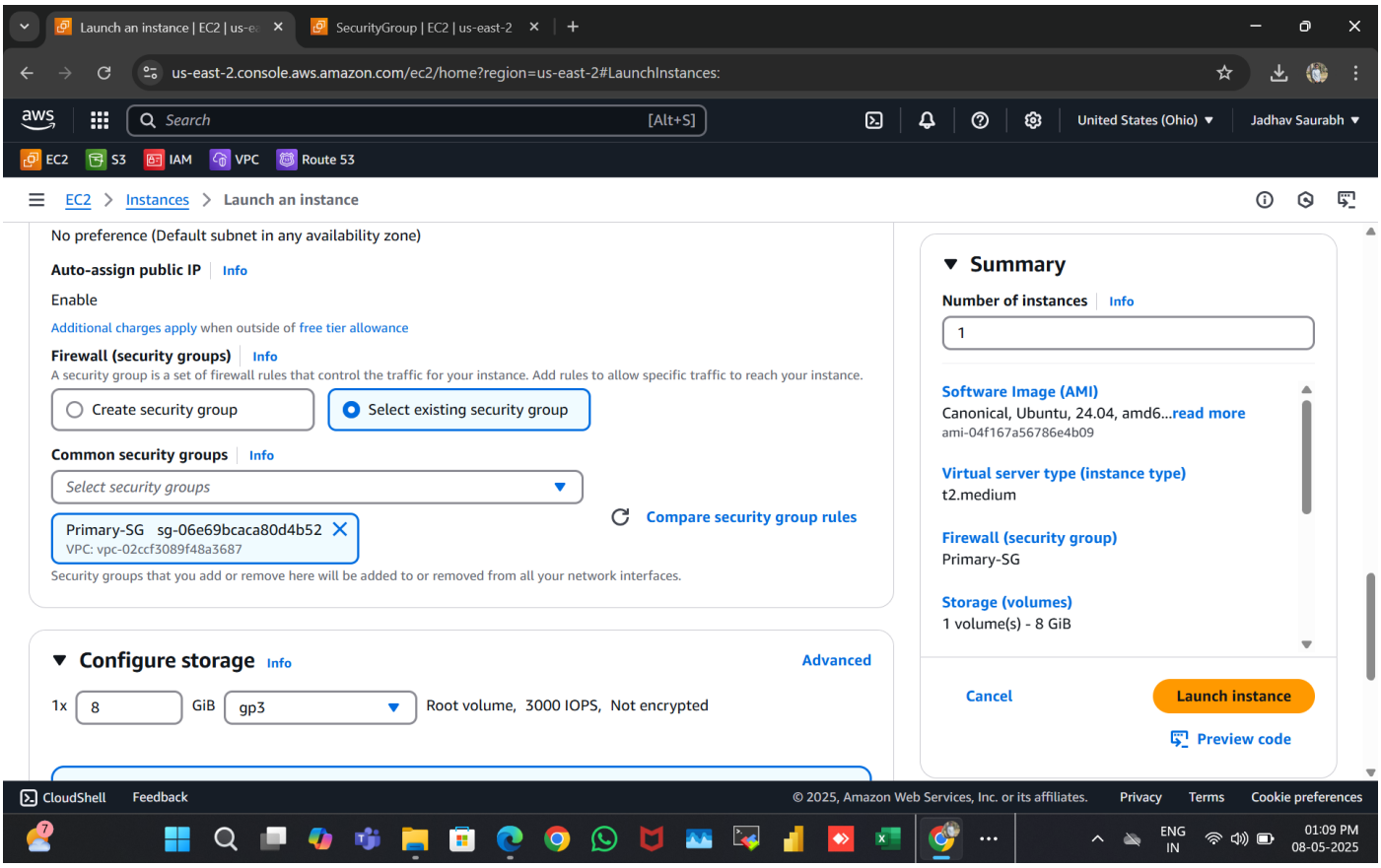
(But only check it which port is allocated if already exist Security Group .)



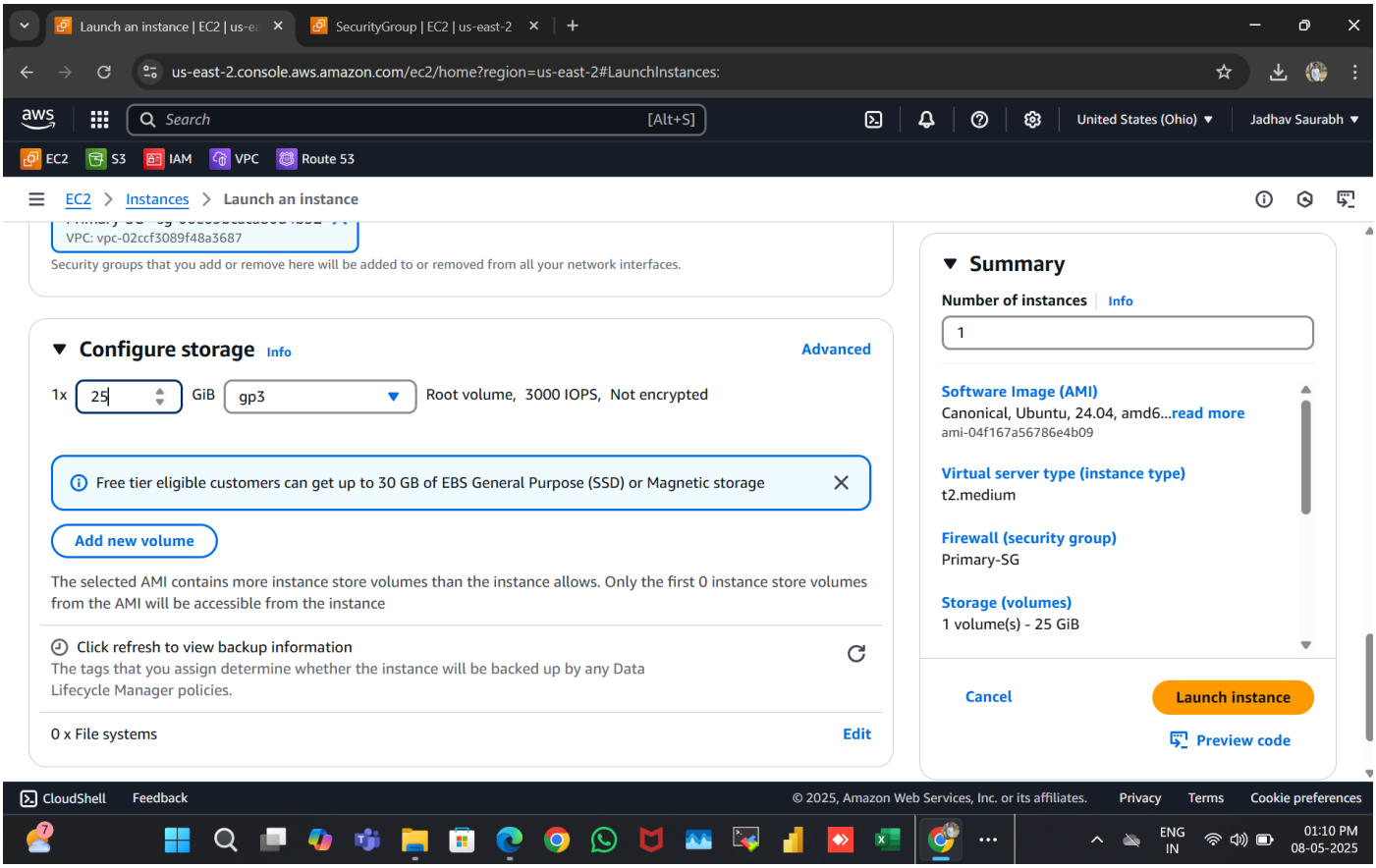
If it is not allocated. Then allocate this ports .



Now select your Security Group .

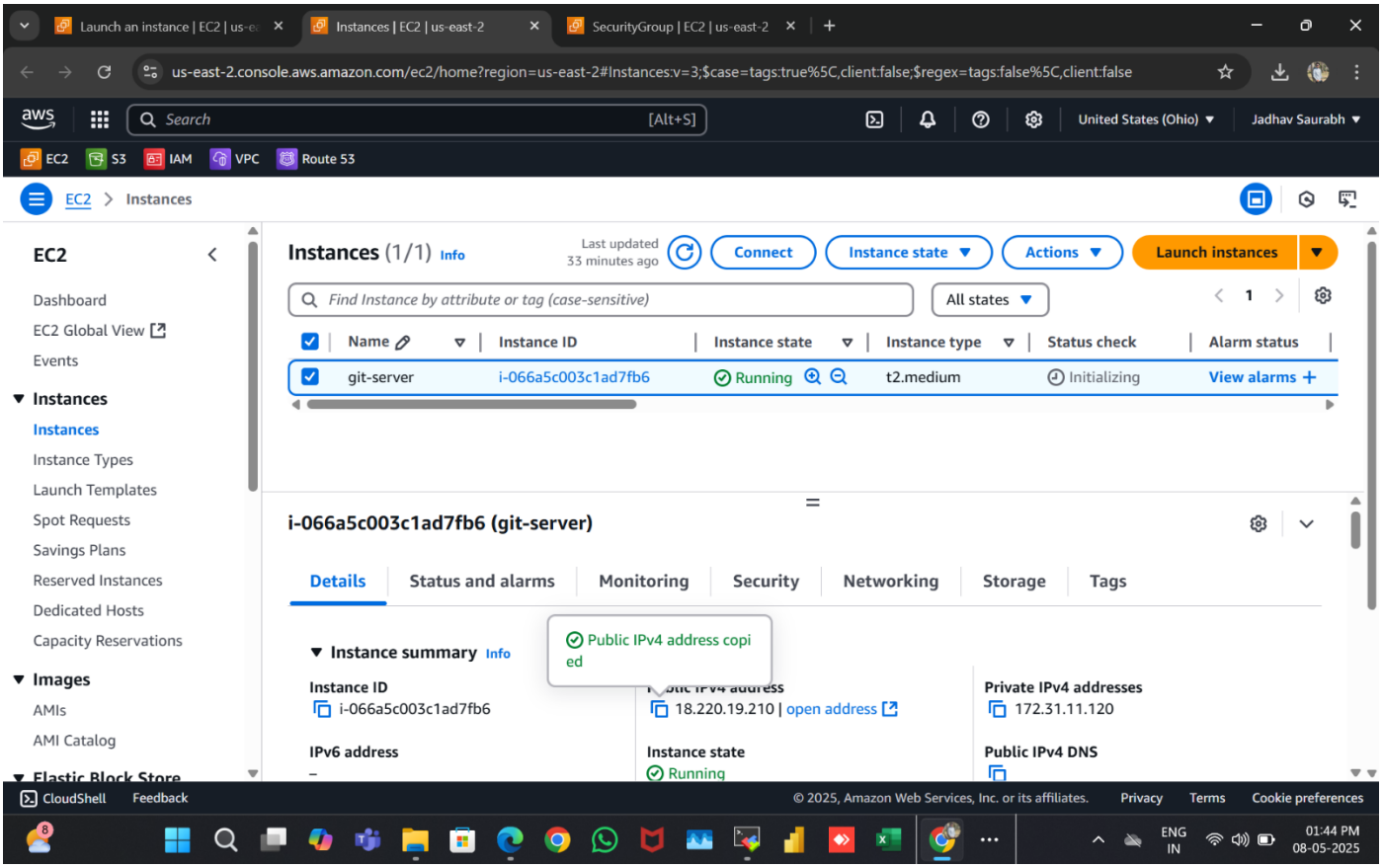


Configure the storage of your Machine .

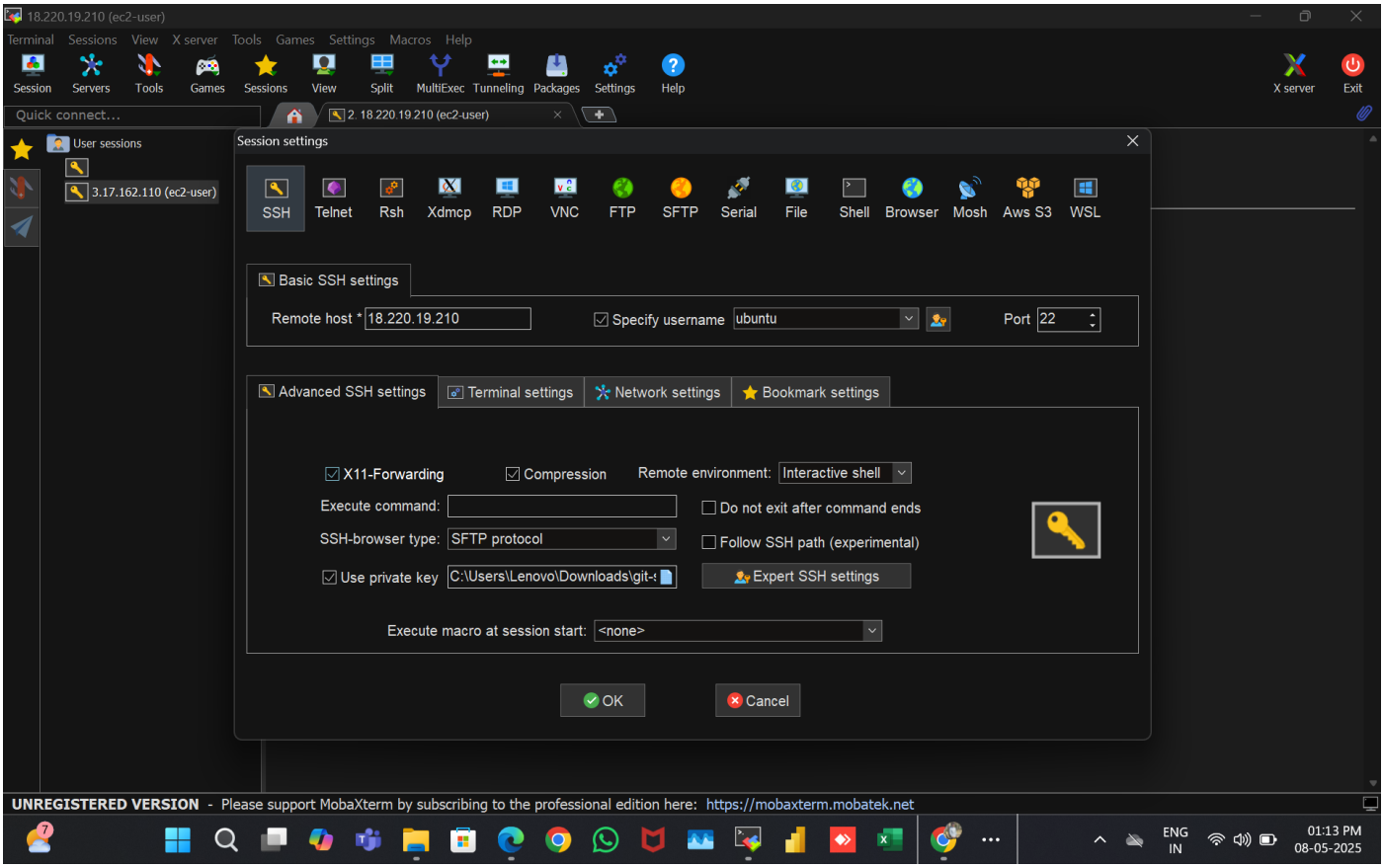


Launch Your Ec2 Instance and Now its Started Successfully.

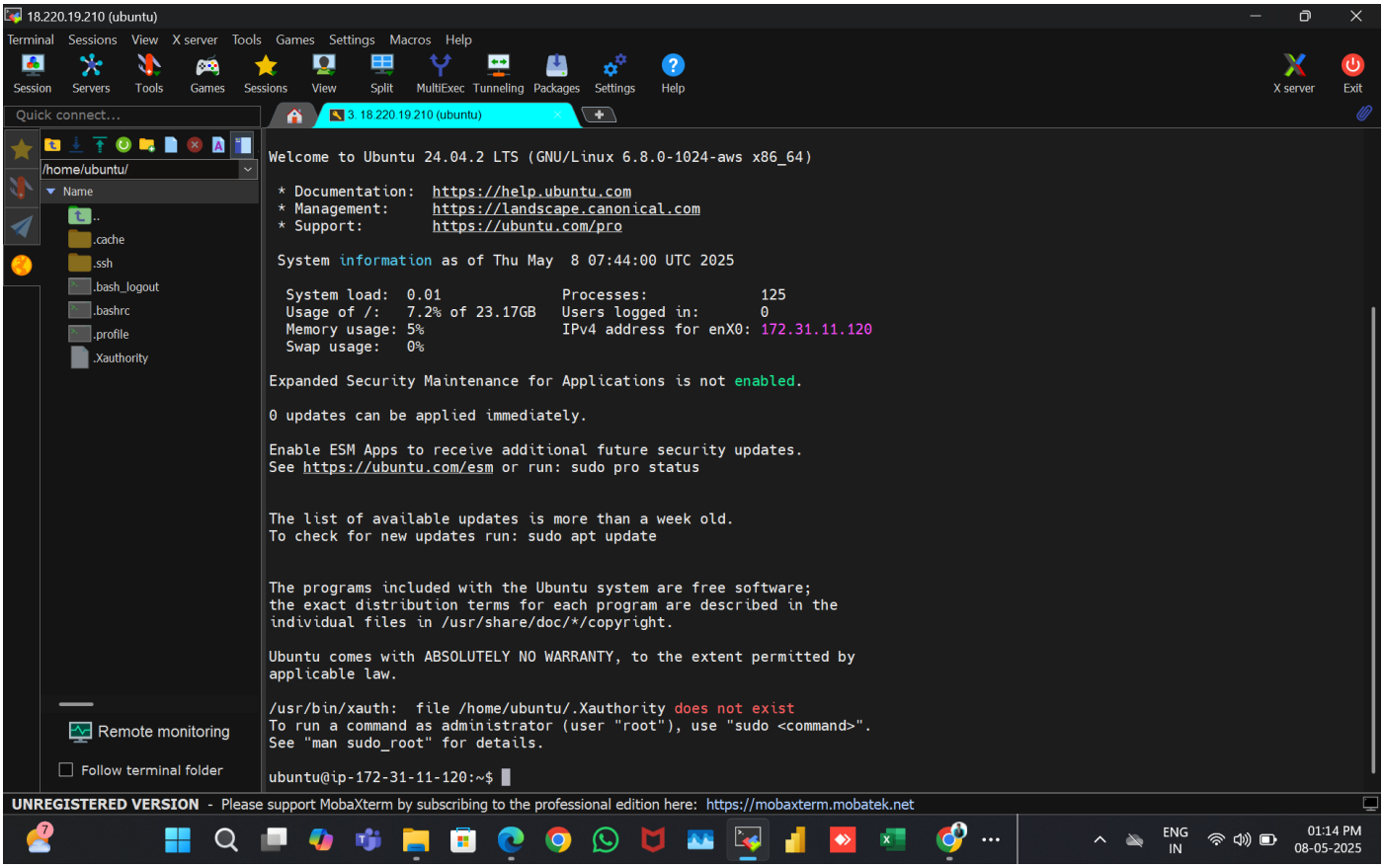
And Copy Public Ip for Remote Access.



Connect to EC2 via SSH and Upload your Private Key.



Now Successfully getting Connect to EC2 via SSH.



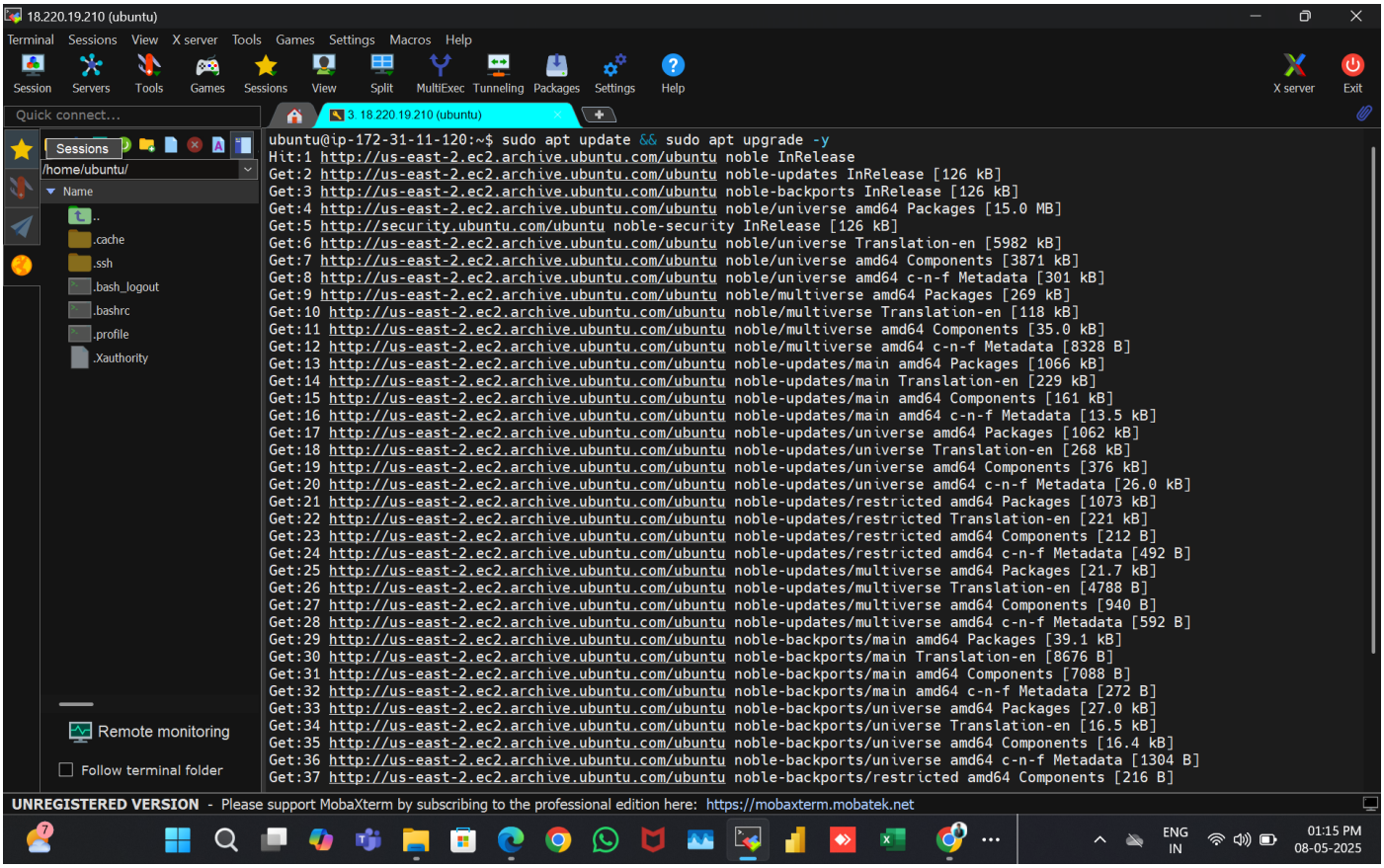
Part 2: Install GUI (Desktop Environment)

Install the Ubuntu desktop environment and enable RDP access using XRDP to manage the server with a graphical interface.

Step-by-Step Implementation and Following this Commands .

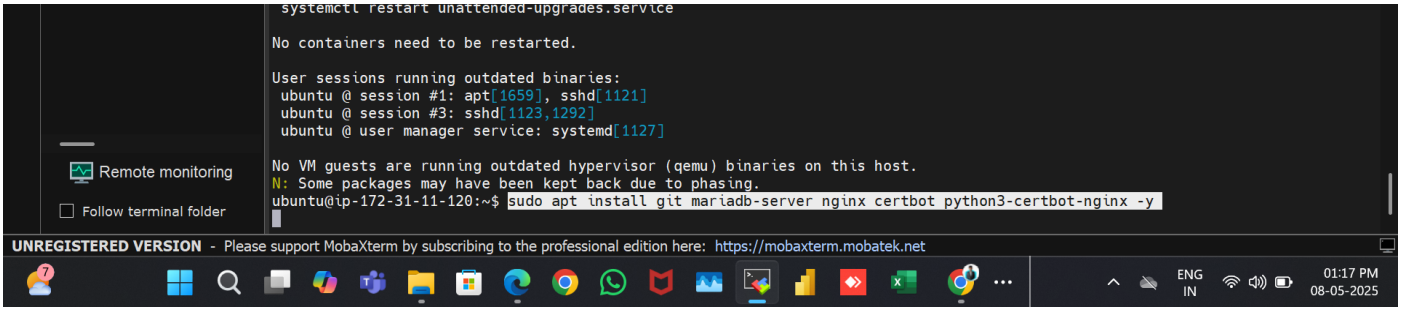
1 . Install Ubuntu & Update System

sudo apt update && sudo apt upgrade -y



2. Install Required Packages

sudo apt install git mariadb-server nginx certbot python3-certbot-nginx -y

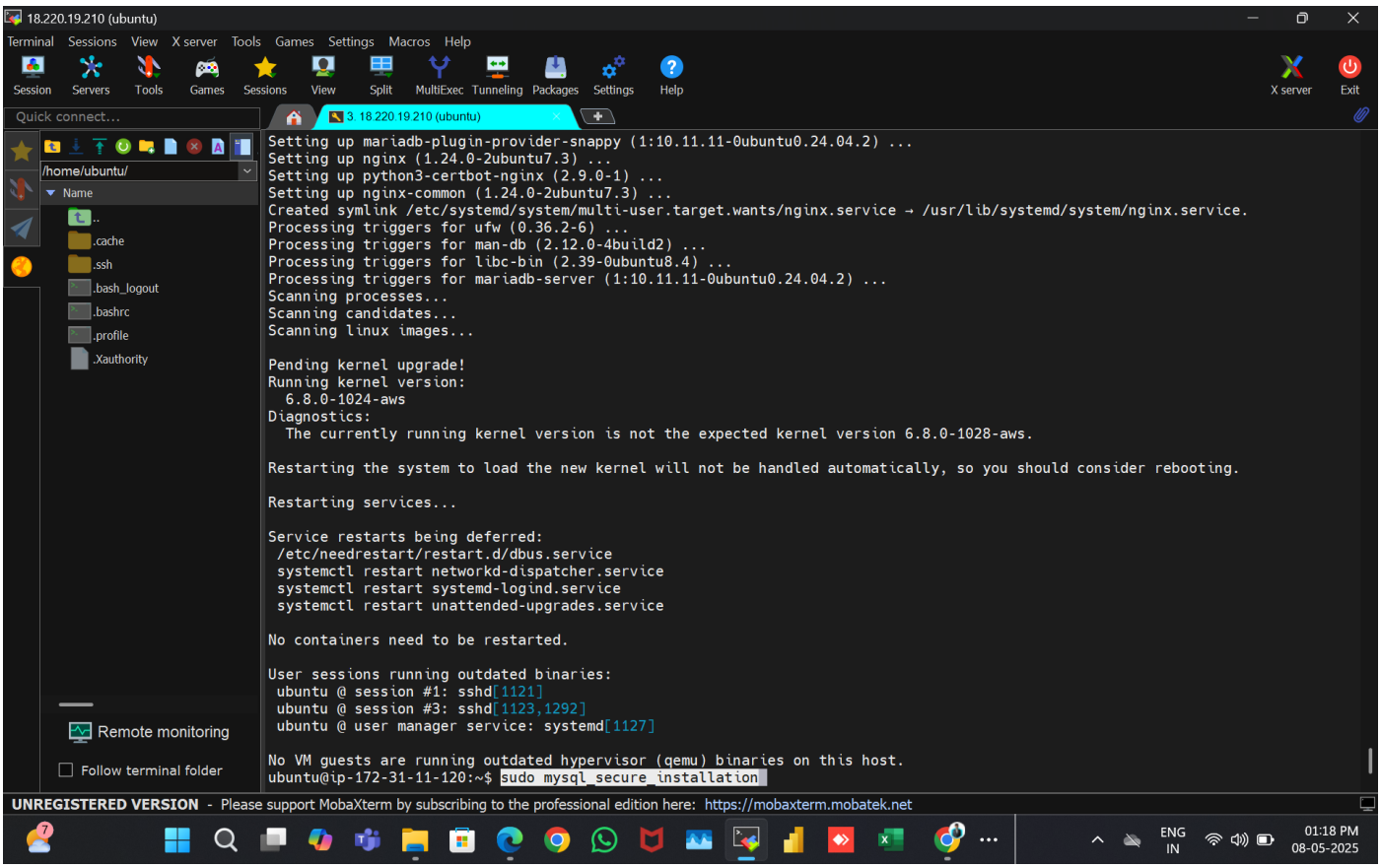


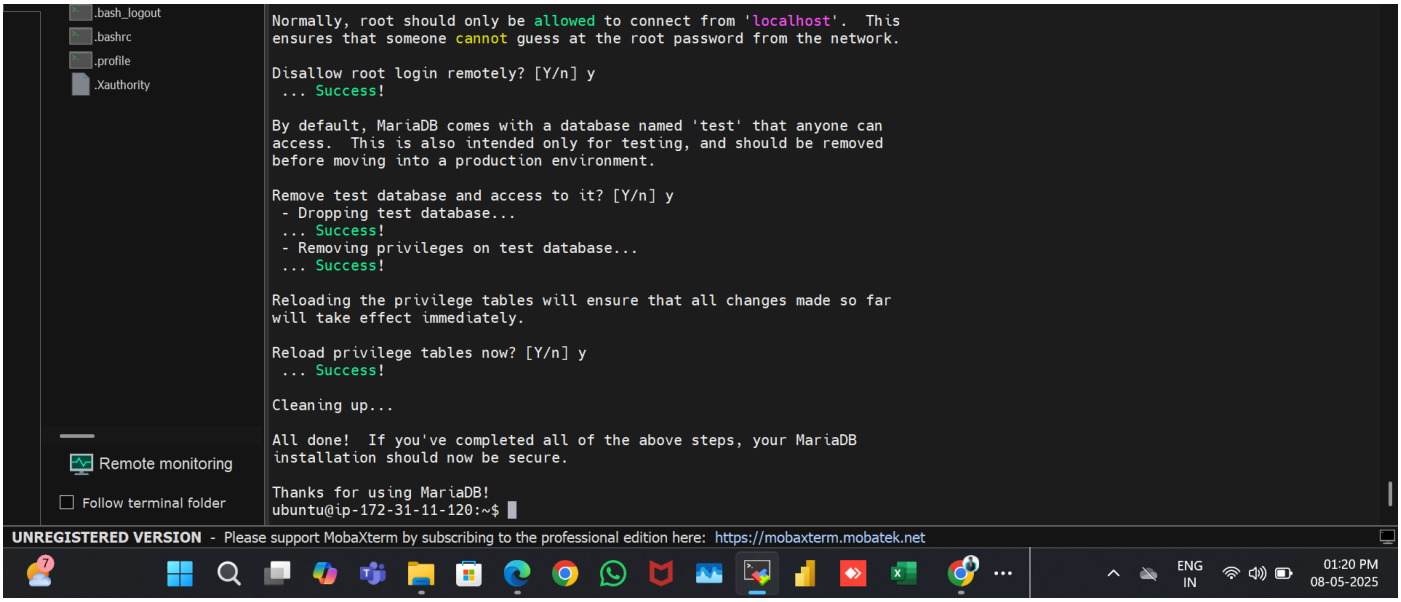
3 . Secure MariaDB

sudo mysql_secure_installation

Enhance the security of your MariaDB server by running sudo mysql_secure_installation. This guided setup allows you to:

- **Set a strong root password** to prevent unauthorized access.
- **Remove anonymous users** who could pose a security risk.
- **Disable remote root login** to ensure the root account can only be accessed locally, reducing exposure to external threats.

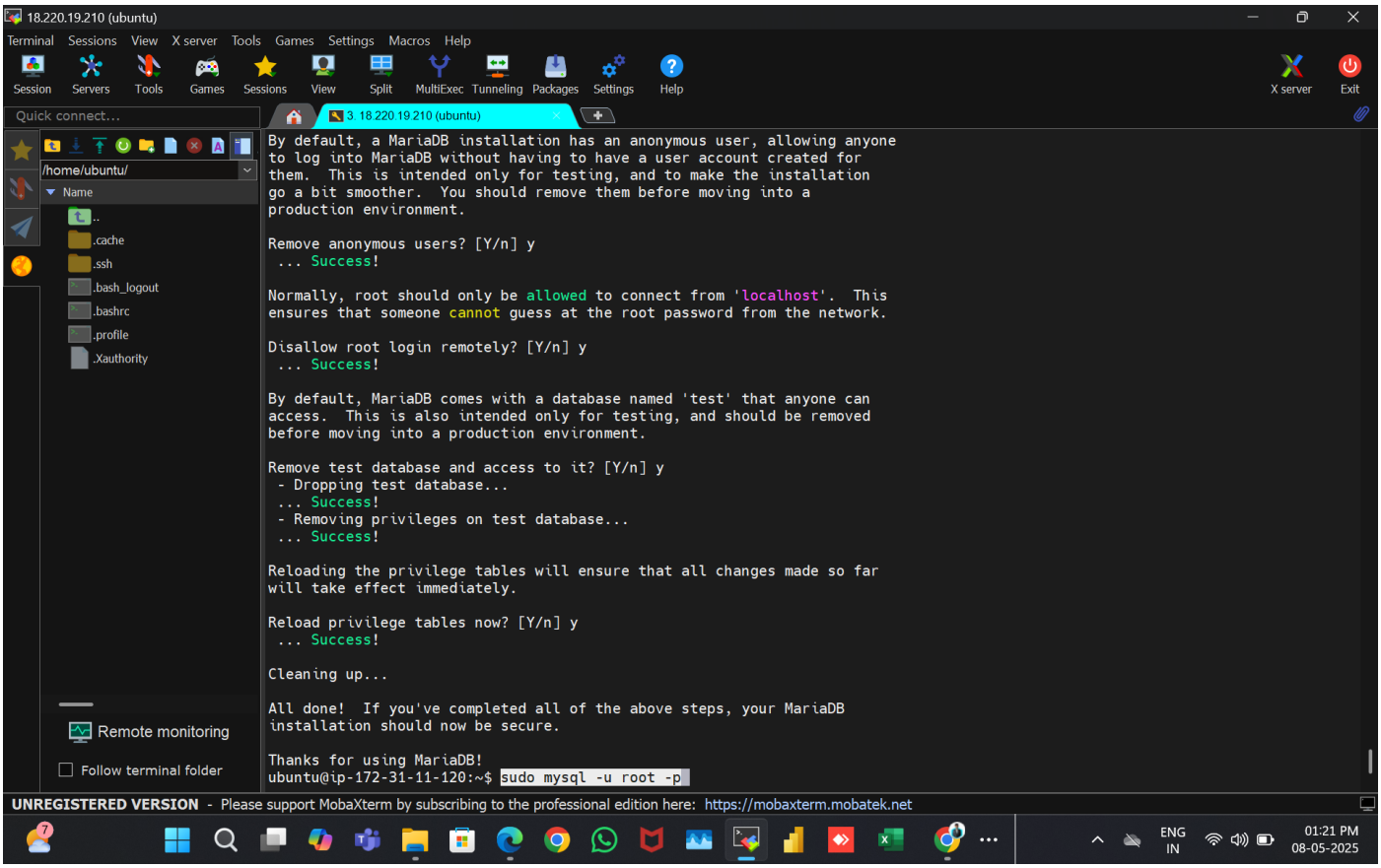




4. Create Gitea Database & User

sudo mysql -u root -p

Log in to MariaDB using `sudo mysql -u root -p`, then create a dedicated **Gitea database and user** with appropriate privileges to manage repositories and user data.



Access MariaDB and run the following commands to create the Gitea database and user with full privileges :

Run:

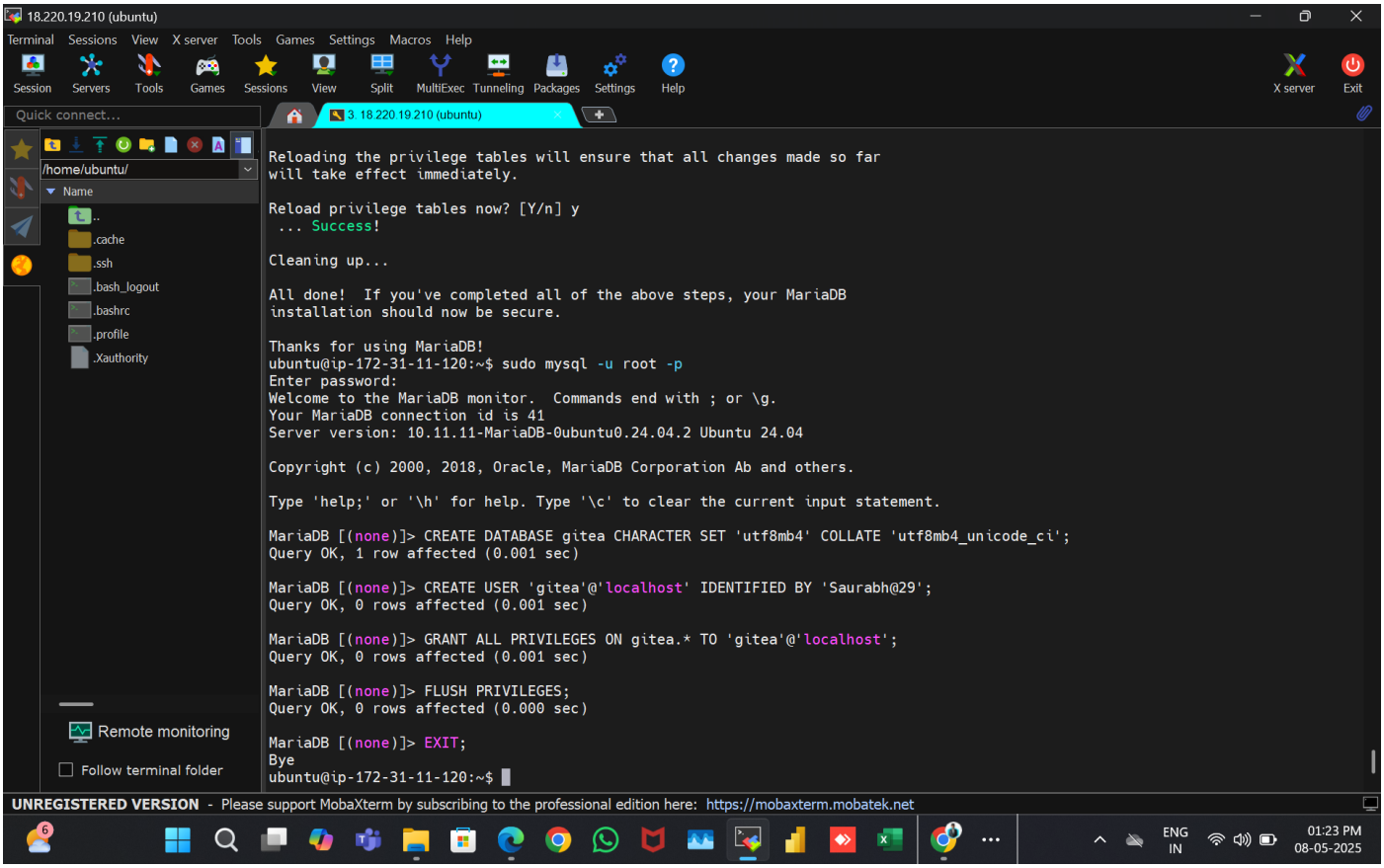
CREATE DATABASE gitea CHARACTER SET 'utf8mb4' COLLATE 'utf8mb4_unicode_ci';

CREATE USER 'gitea'@'localhost' IDENTIFIED BY 'Saurabh@29';

GRANT ALL PRIVILEGES ON gitea.* TO 'gitea'@'localhost';

FLUSH PRIVILEGES;

EXIT;



5 . Install & Configure Gitea

Download & Install Gitea

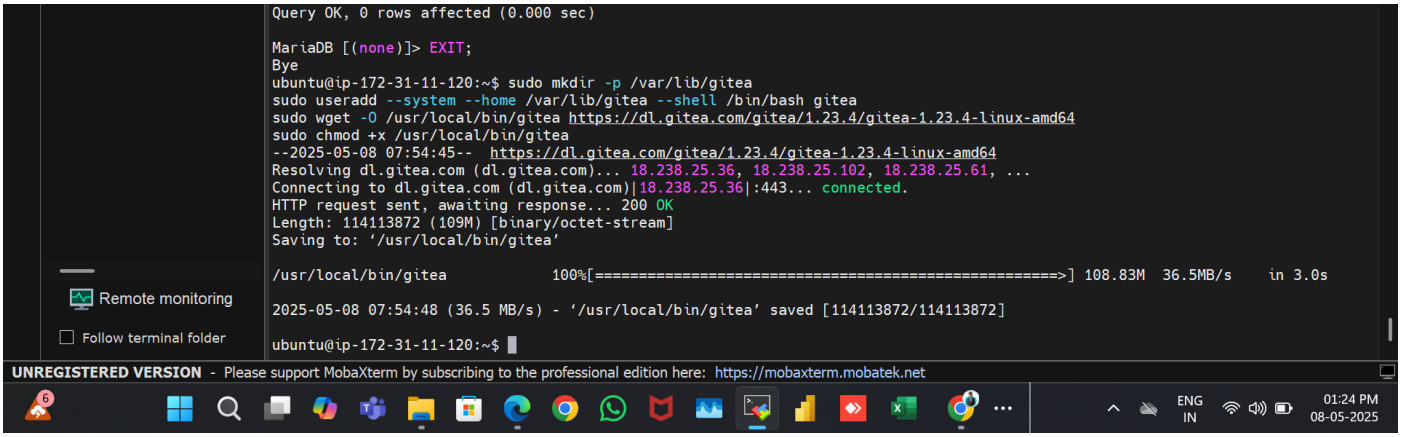
Create necessary directories and a system user for Gitea, then download the Gitea binary :

sudo mkdir -p /var/lib/gitea

sudo useradd --system --home /var/lib/gitea --shell /bin/bash gitea

sudo wget -O /usr/local/bin/gitea https://dl.gitea.com/gitea/1.23.4/gitea-1.23.4-linux-amd64

sudo chmod +x /usr/local/bin/gitea



Create Configuration & Data Directories.

Set up Gitea's required directories, assign proper ownership to the gitea user, and secure the configuration file :

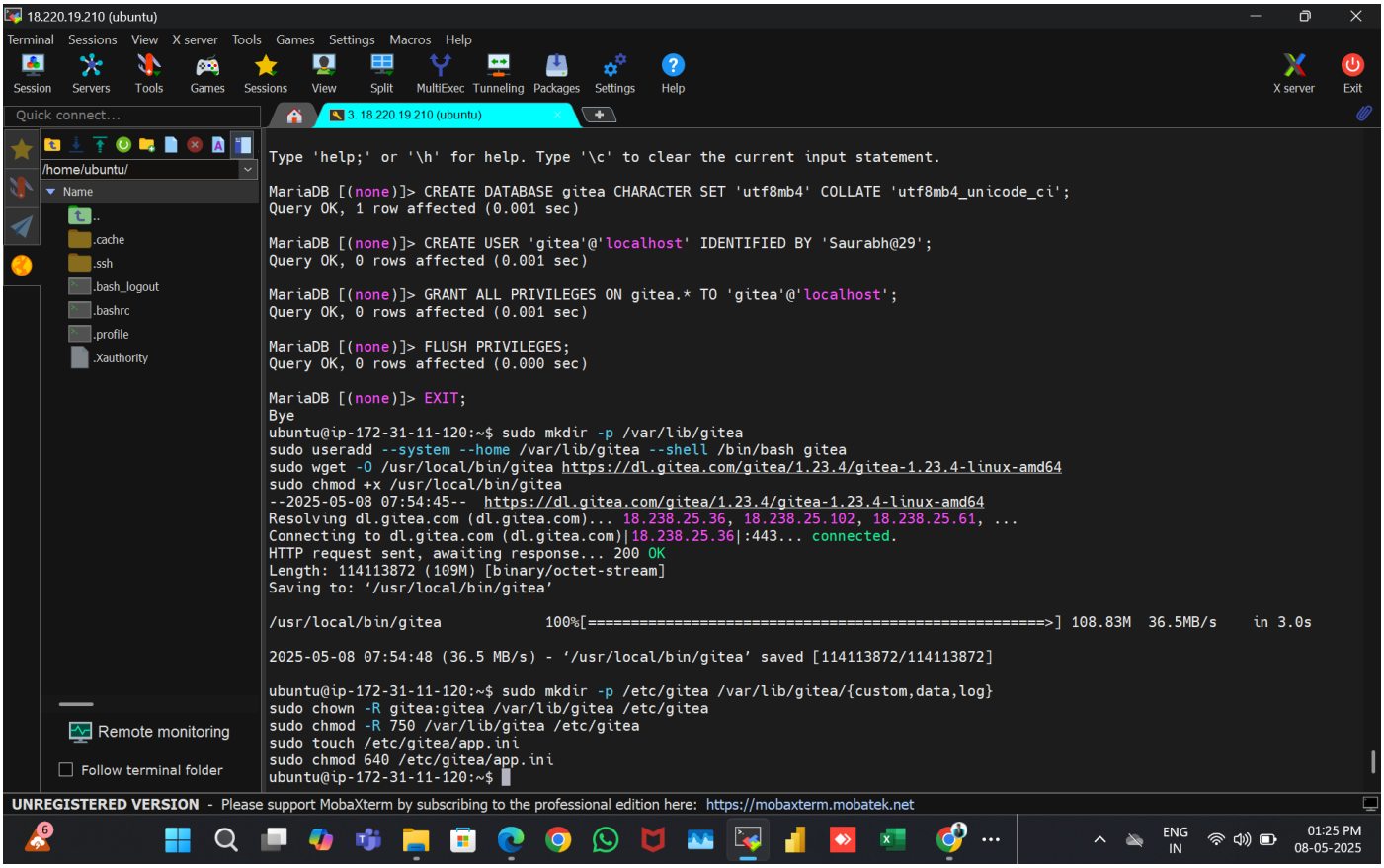
sudo mkdir -p /etc/gitea /var/lib/gitea/{custom,data,log}

sudo chown -R gitea:gitea /var/lib/gitea /etc/gitea

sudo chmod -R 750 /var/lib/gitea /etc/gitea

sudo touch /etc/gitea/app.ini

sudo chmod 640 /etc/gitea/app.ini

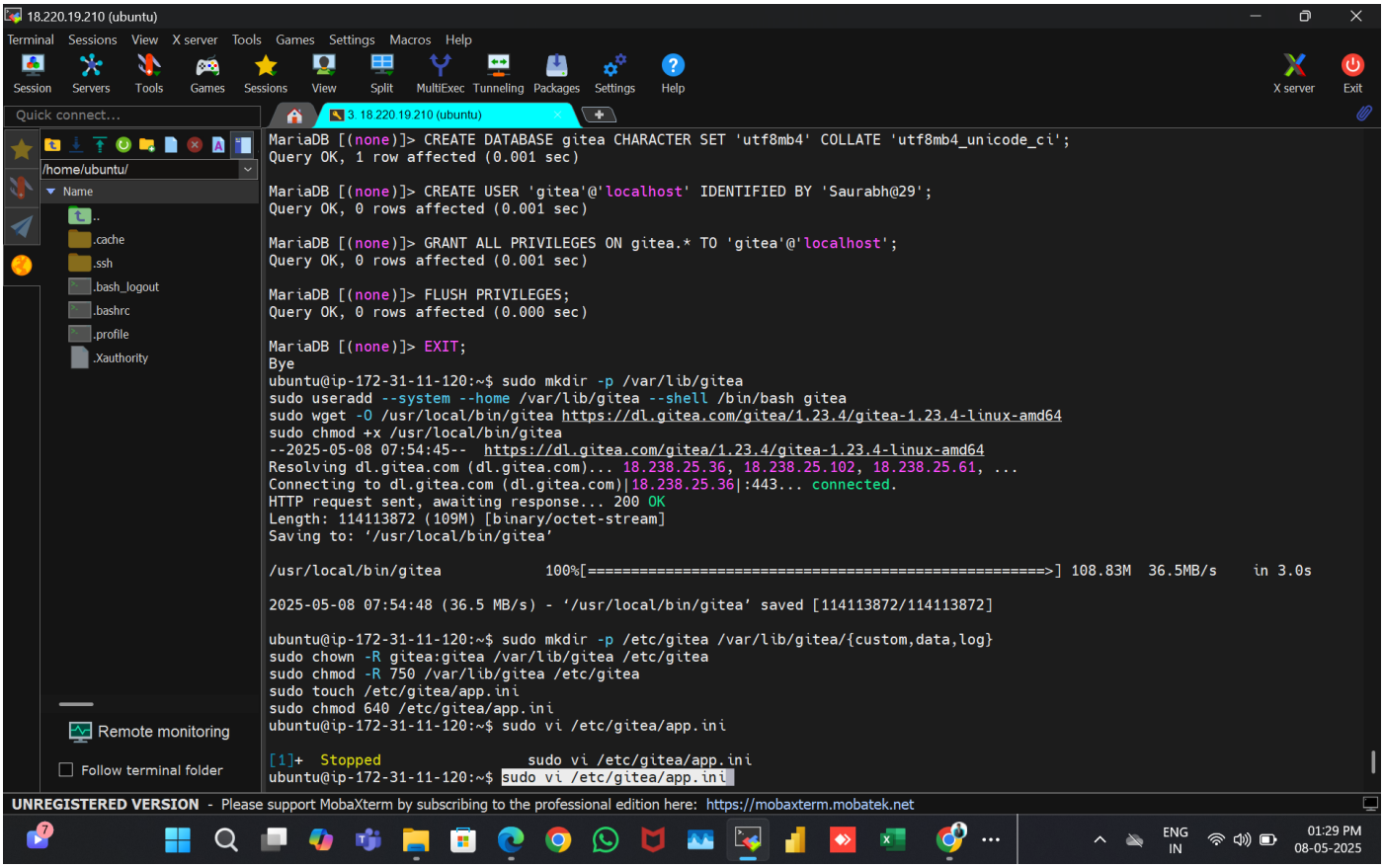


6 . Configure app.ini

Edit the Gitea configuration file to define repository paths, server settings, and logging details :

Open the Gitea configuration file :

sudo vi /etc/gitea/app.ini



Update Repository Paths

[repository]

ROOT = /var/lib/gitea/data/gitea-repositories

Update Server Configuration

[server]

APP_DATA_PATH = /var/lib/gitea/data

DOMAIN = git.saurabhcloud.fun

SSH_DOMAIN = git.saurabhcloud.fun

HTTP_PORT = 3000

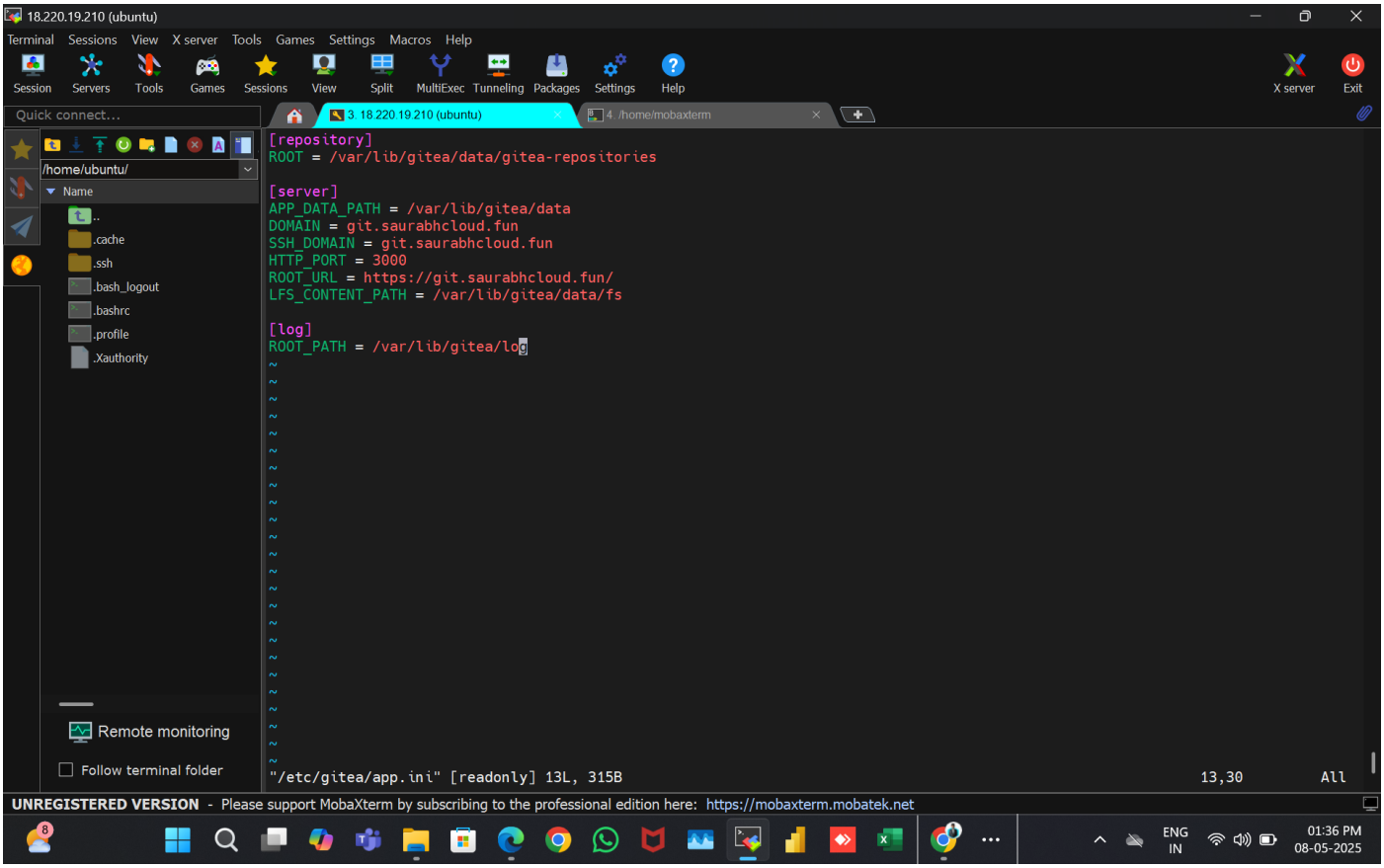
ROOT_URL = https://git.saurabhcloud.fun/

LFS_CONTENT_PATH = /var/lib/gitea/data/fs

Update Log Configuration

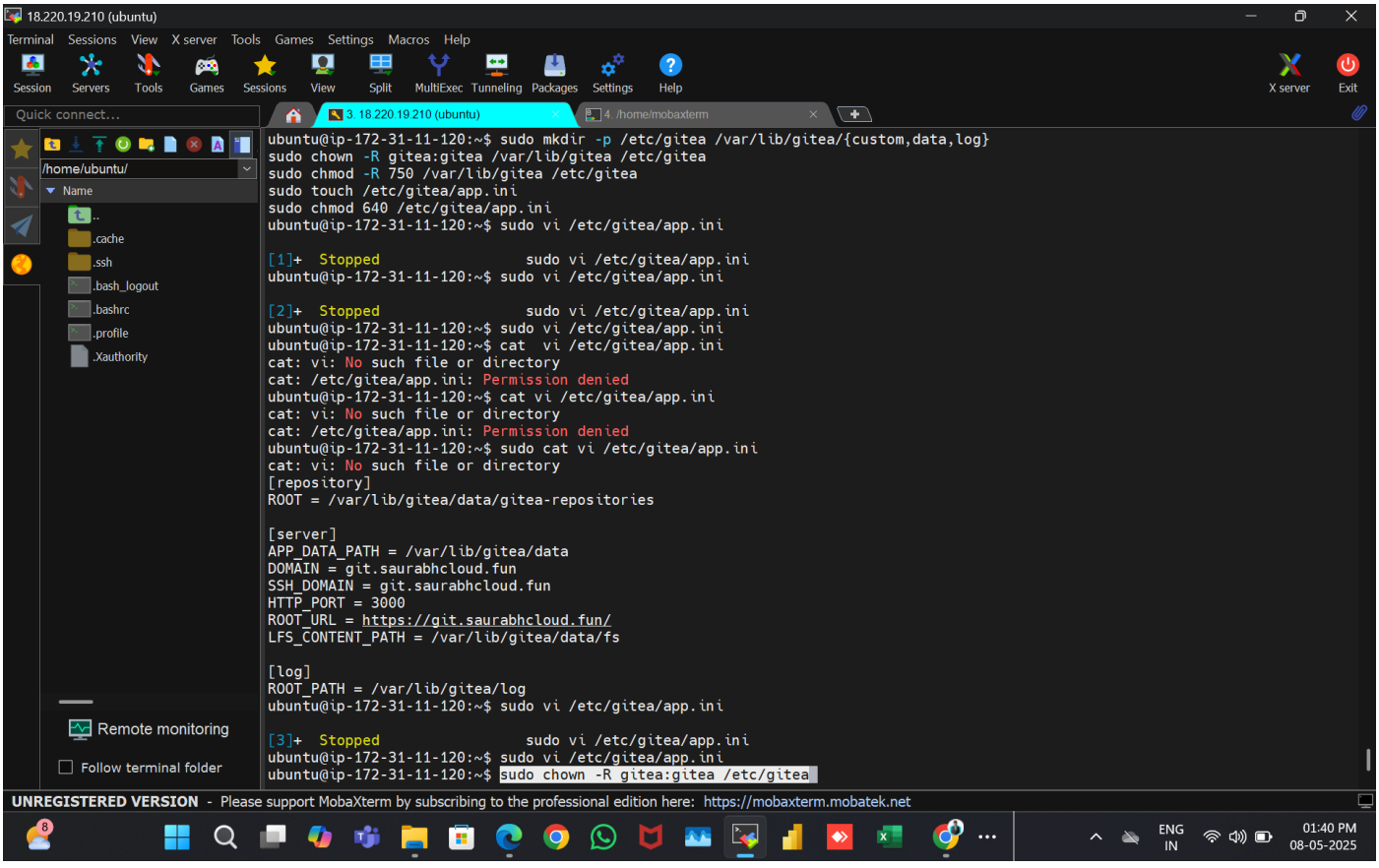
[log]

ROOT_PATH = /var/lib/gitea/log

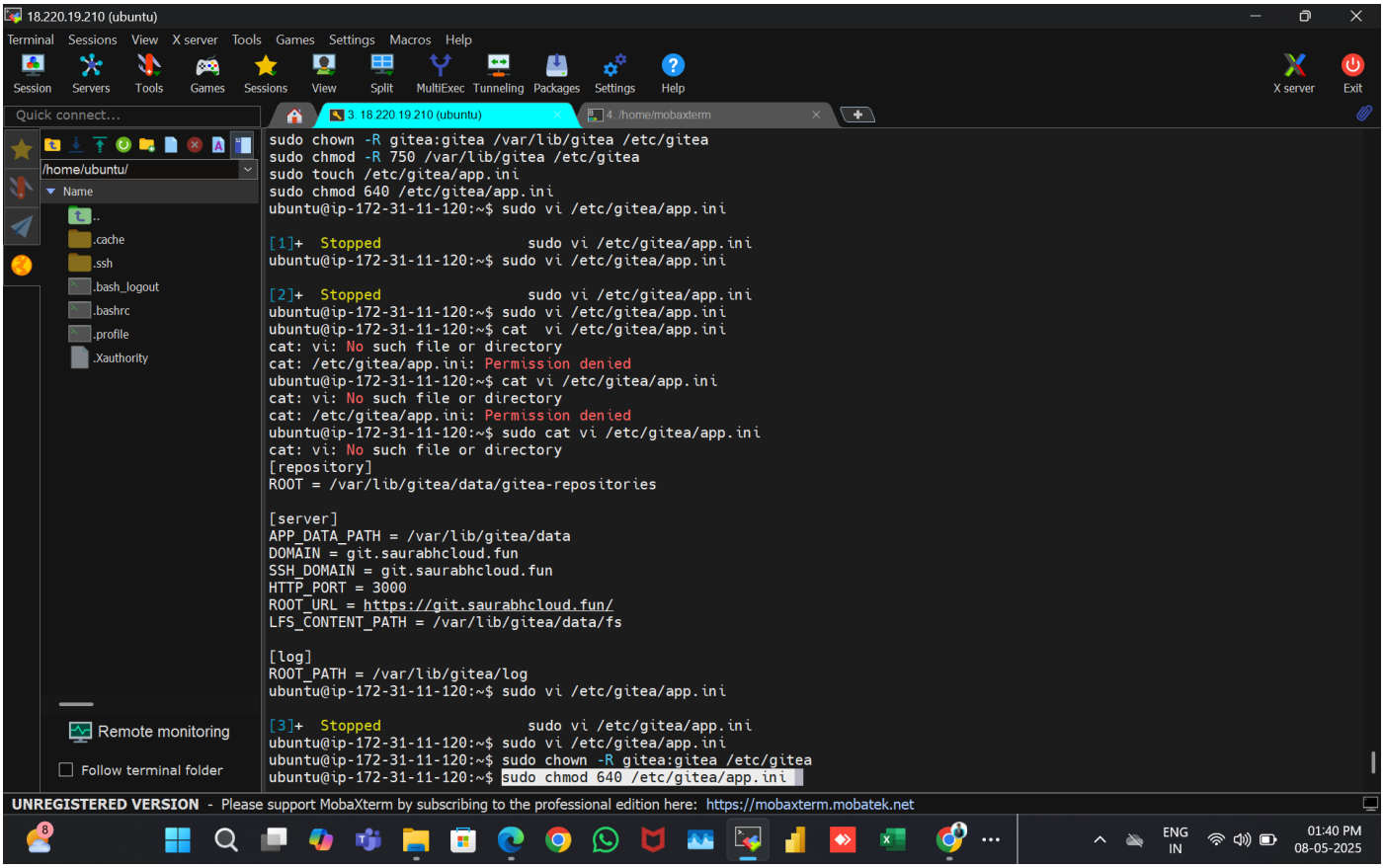


Fix permissions :

sudo chown -R gitea:gitea /etc/gitea



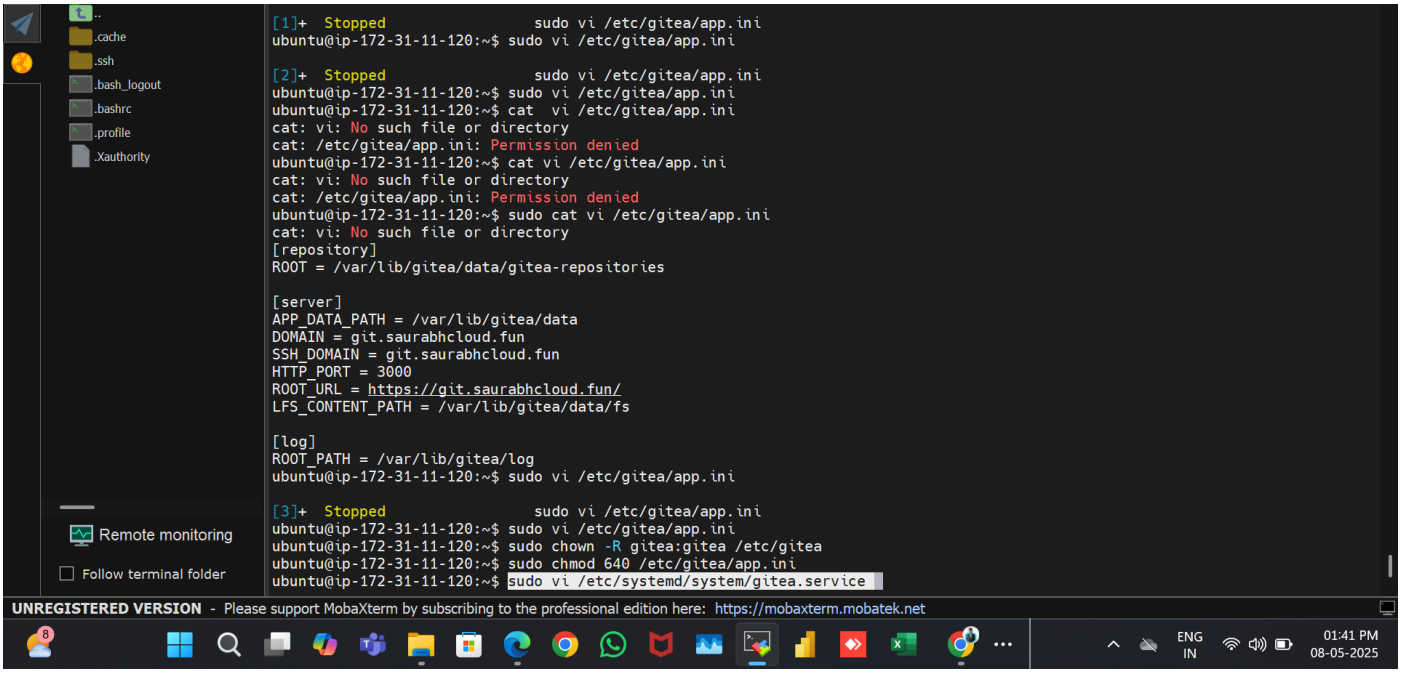
sudo chmod 640 /etc/gitea/app.ini



9. Configure Systemd Service for Gitea :

Create a systemd service file to manage Gitea as a background service :

sudo vi /etc/systemd/system/gitea.service



Add the following configuration :

[Unit]

Description=Gitea Self-Hosted Git Server

After=network.target mariadb.service

[Service]

User=gitea

Group=gitea

ExecStart=/usr/local/bin/gitea web --config /etc/gitea/app.ini

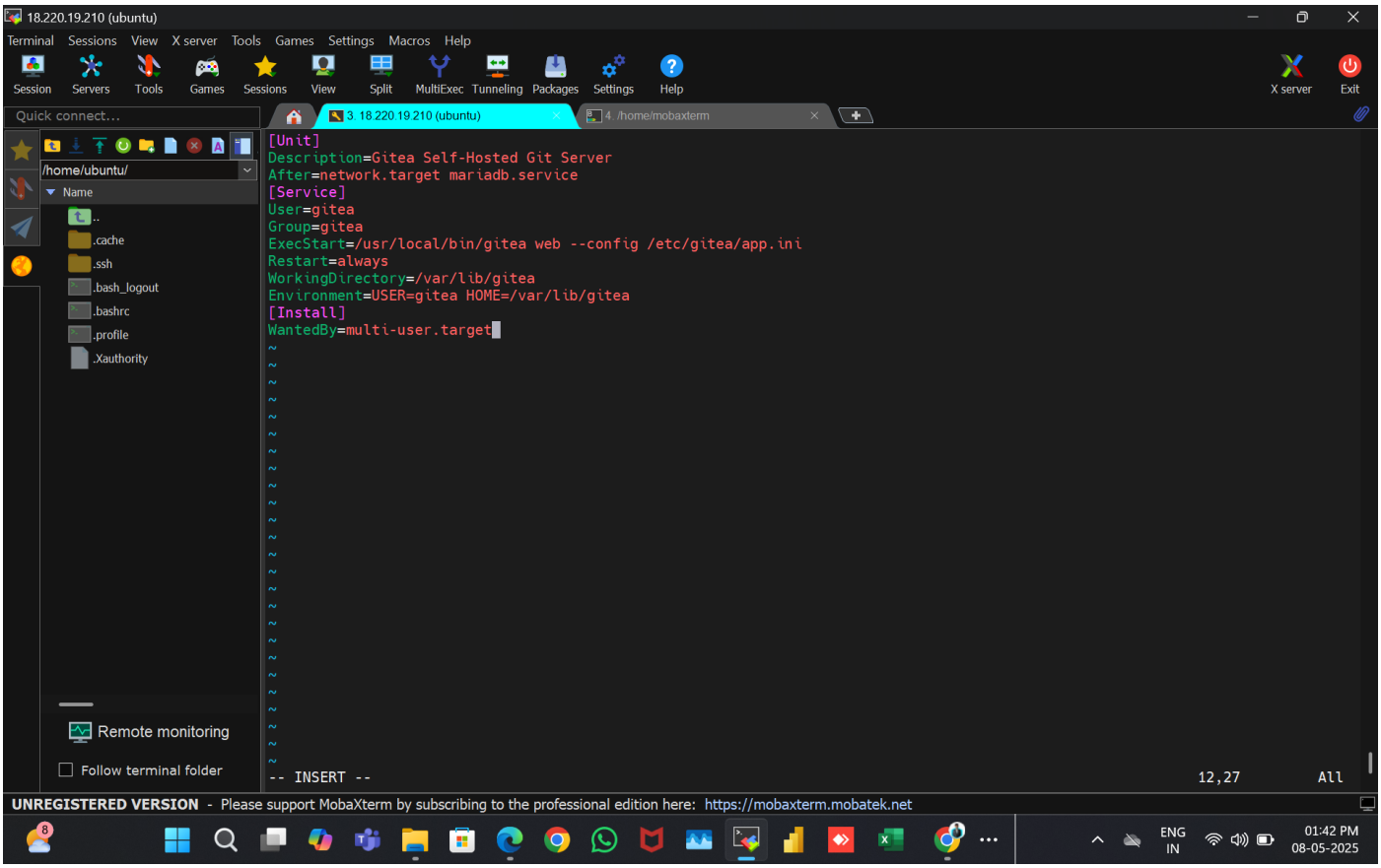
Restart=always

WorkingDirectory=/var/lib/gitea

Environment=USER=gitea HOME=/var/lib/gitea

[Install]

WantedBy=multi-user.target



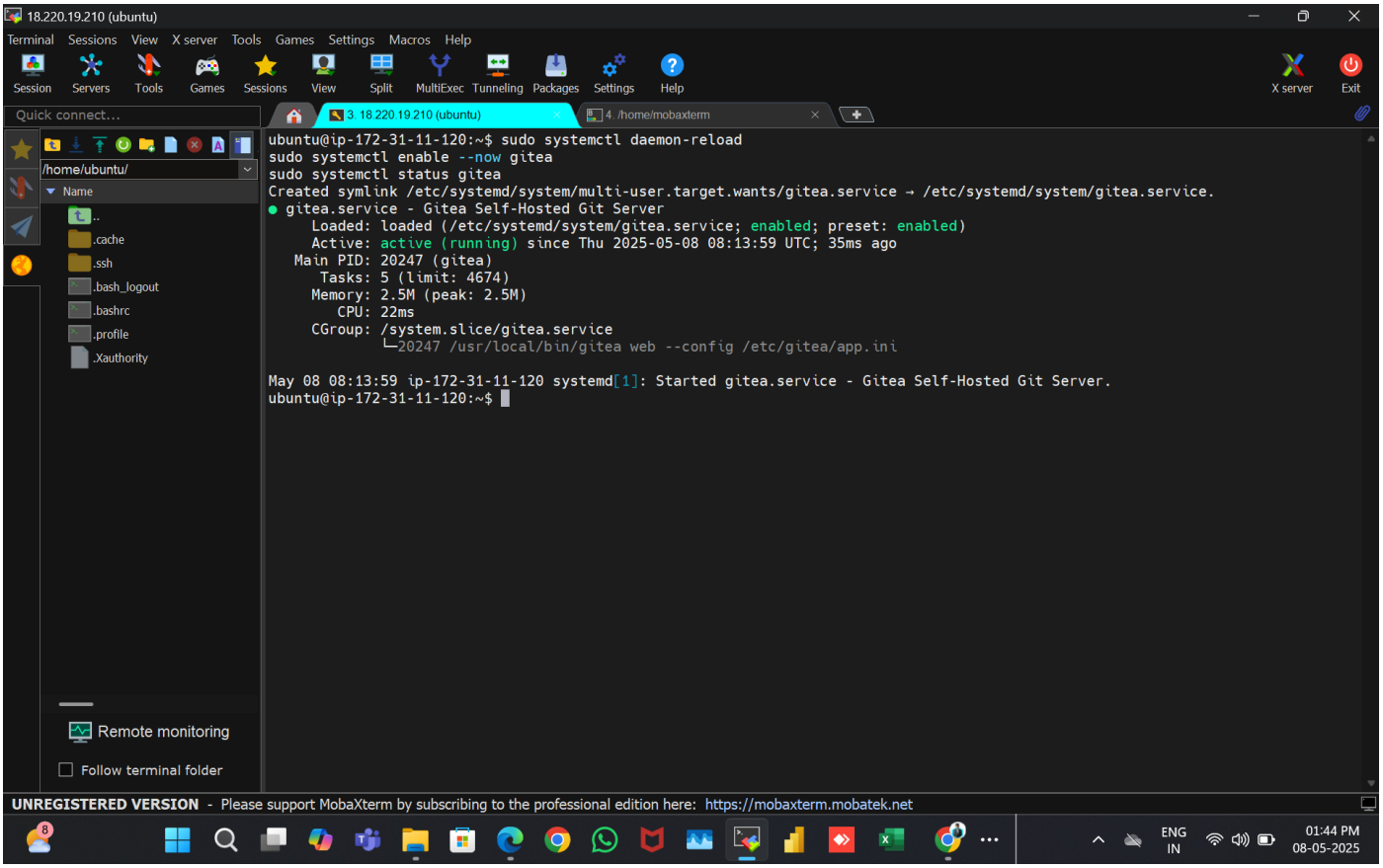
Reload Systemd & Start Gitea

Activate the Gitea service so it starts immediately and on boot :

sudo systemctl daemon-reload

sudo systemctl enable --now gitea

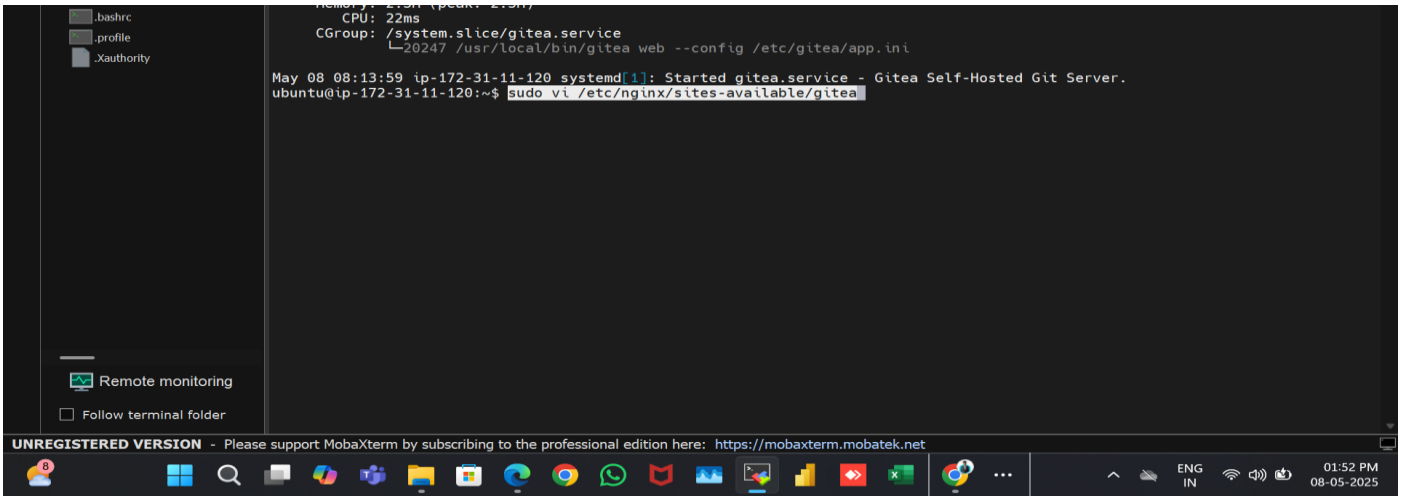
sudo systemctl status gitea



8 . Set Up Reverse Proxy with Nginx

Create an Nginx config file to route traffic from your custom domain to the Gitea app :

sudo vi /etc/nginx/sites-available/gitea



Add:

server {

listen 80;

server_name git.skjptpp.in;

location / {

proxy_pass http://localhost:3000;

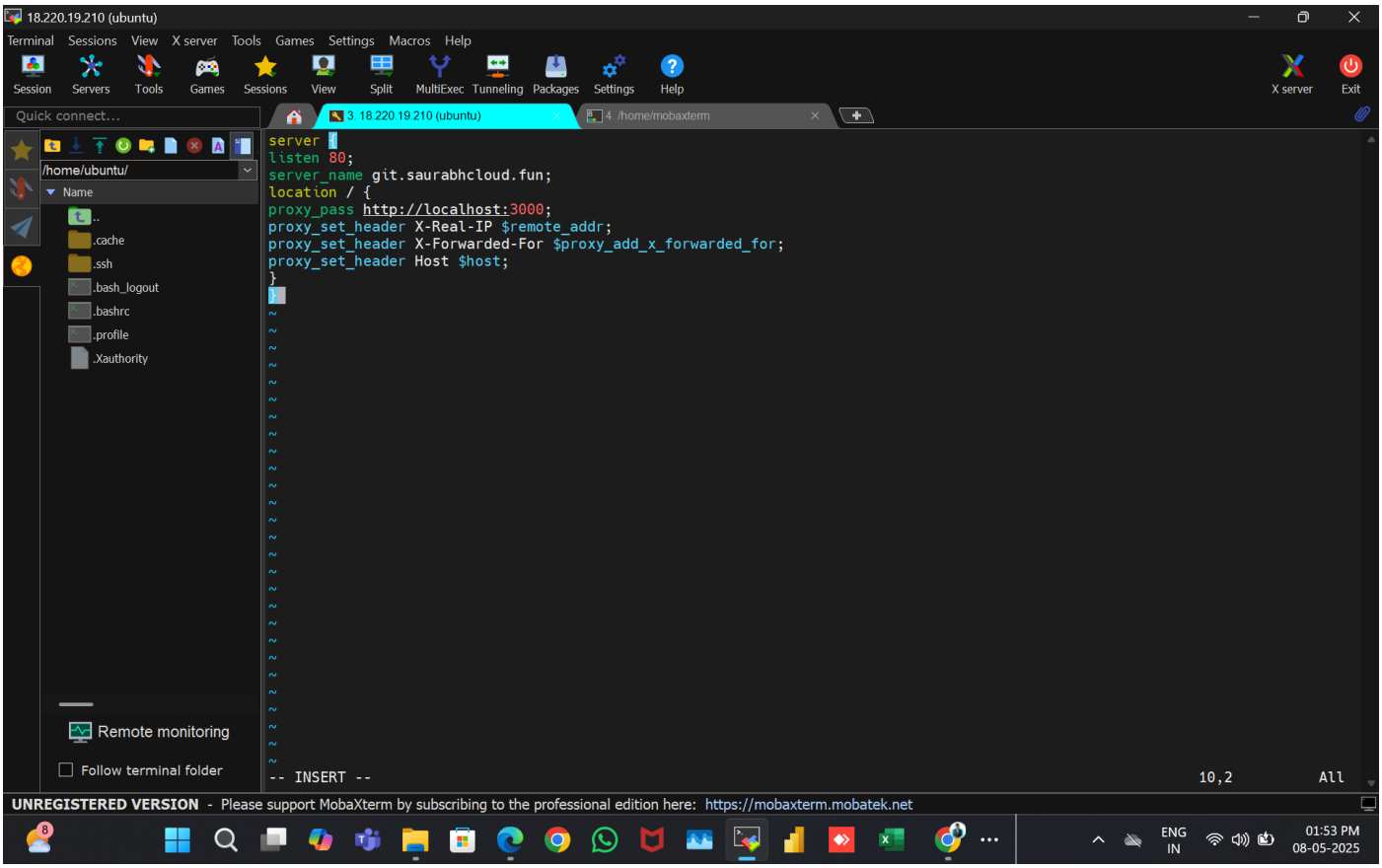
proxy_set_header X-Real-IP $remote_addr;

proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

proxy_set_header Host $host;

}

}

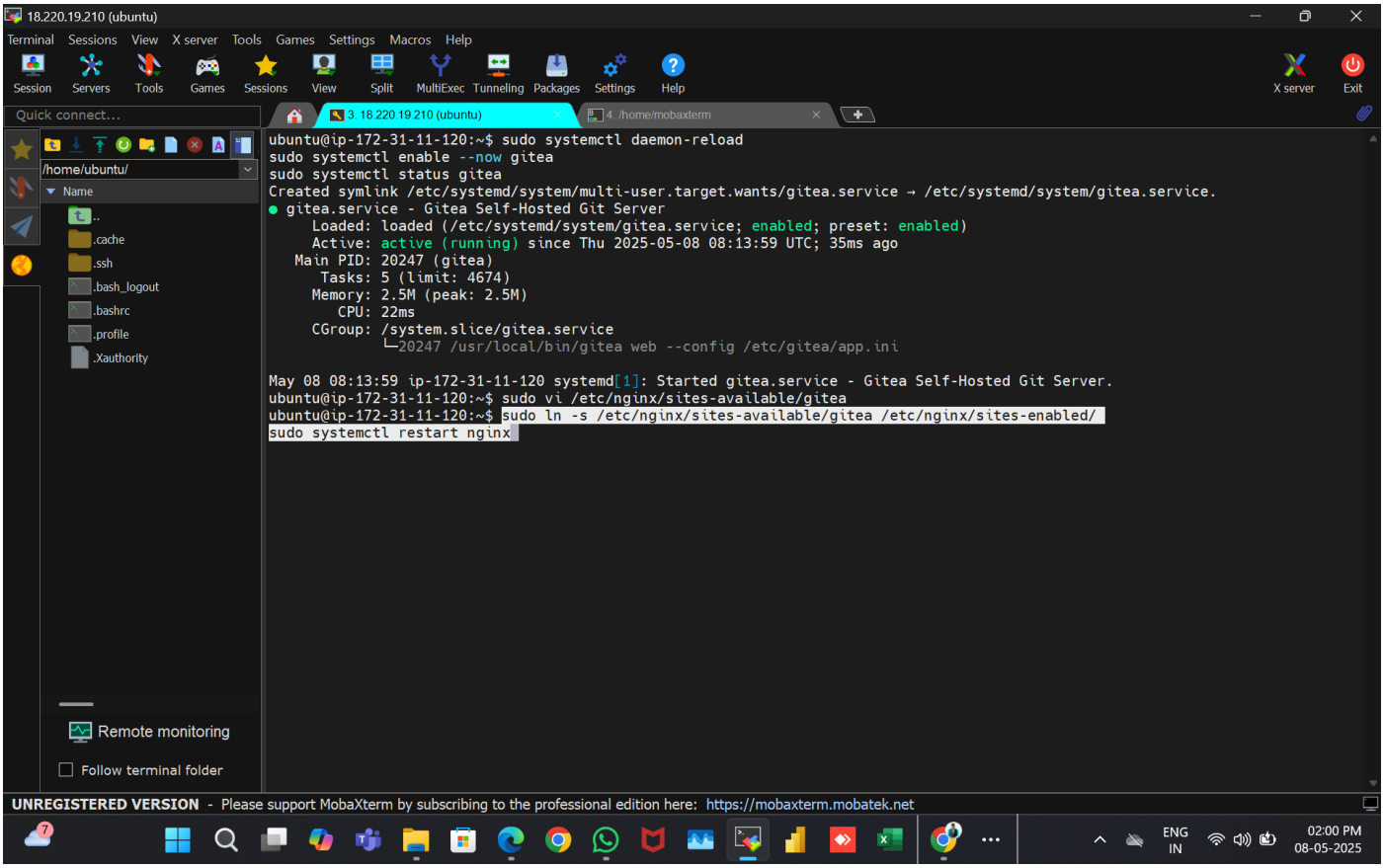


Enable & Restart Nginx :

Activate the Gitea Nginx configuration and restart the Nginx service to apply changes.

sudo ln -s /etc/nginx/sites-available/gitea /etc/nginx/sites-enabled/

sudo systemctl restart nginx

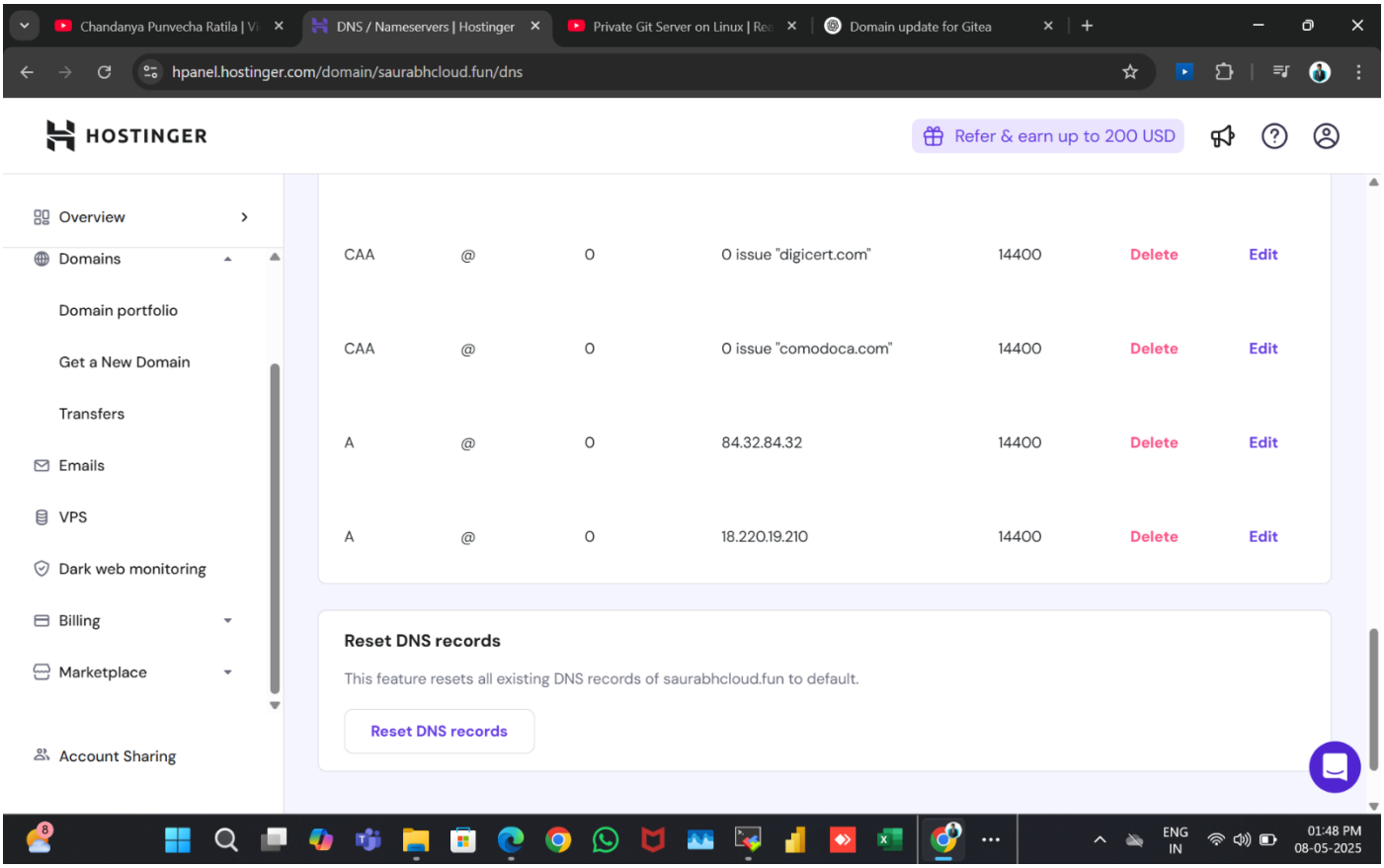


Point Your Domain to Your EC2 Instance

To make (git.saurabhcloud.fun) point to your EC2 instance:

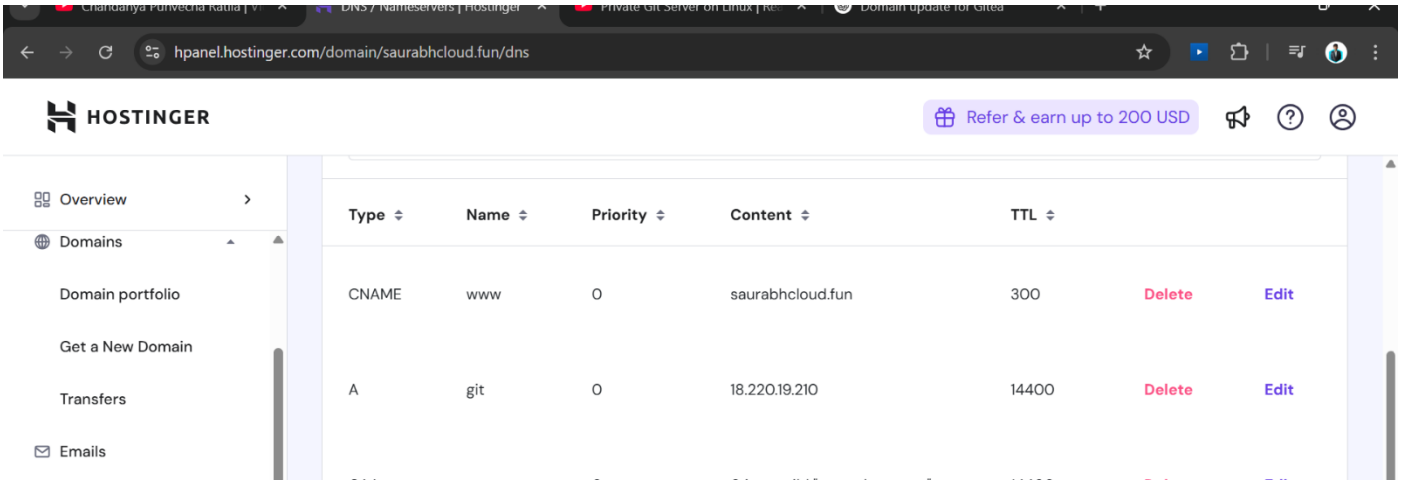
Step-by-step:

1. **Log in to your domain registrar's website**
   (e.g., GoDaddy, Namecheap, Hostinger, etc.)
2. **Go to DNS Management or DNS Zone Editor**
3. **Create or edit an "A" Record:**
   - **Host/Name:** git (or @ if you're pointing the root domain like saurabhcloud.fun)
   - **Type:** A
   - **Value:** Your EC2 Public IPv4 Address
   - **TTL:** 300 or default
4. **Save the record**



Type	Name	Value (EC2 IP)	TTL
A	git	13.234.XX.XX	300

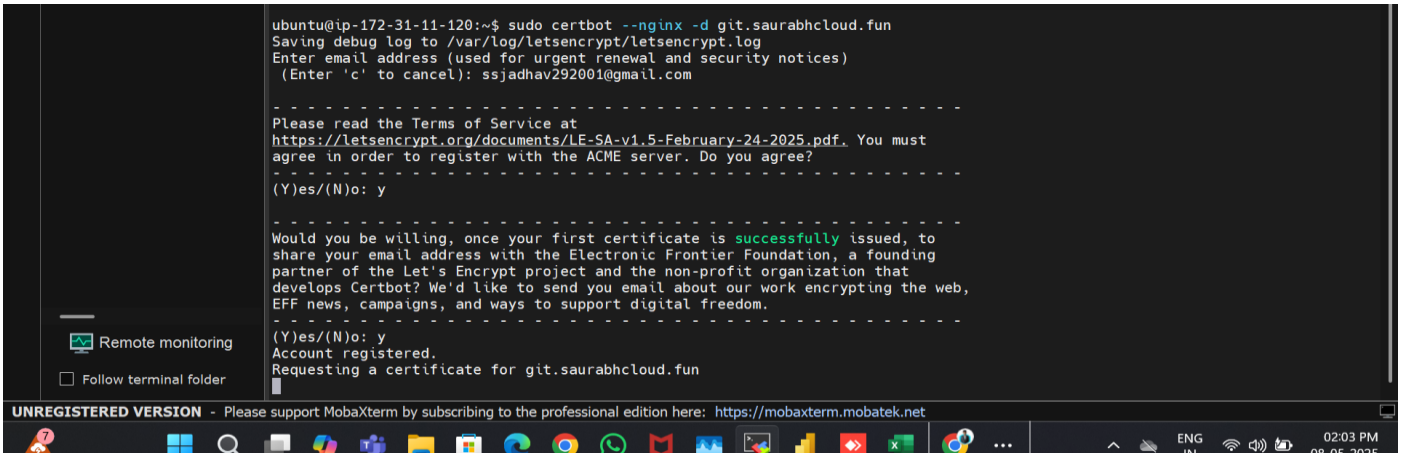
Now, when someone opens https://git.saurabhcloud.fun, it will route to your EC2 instance running Gitea.



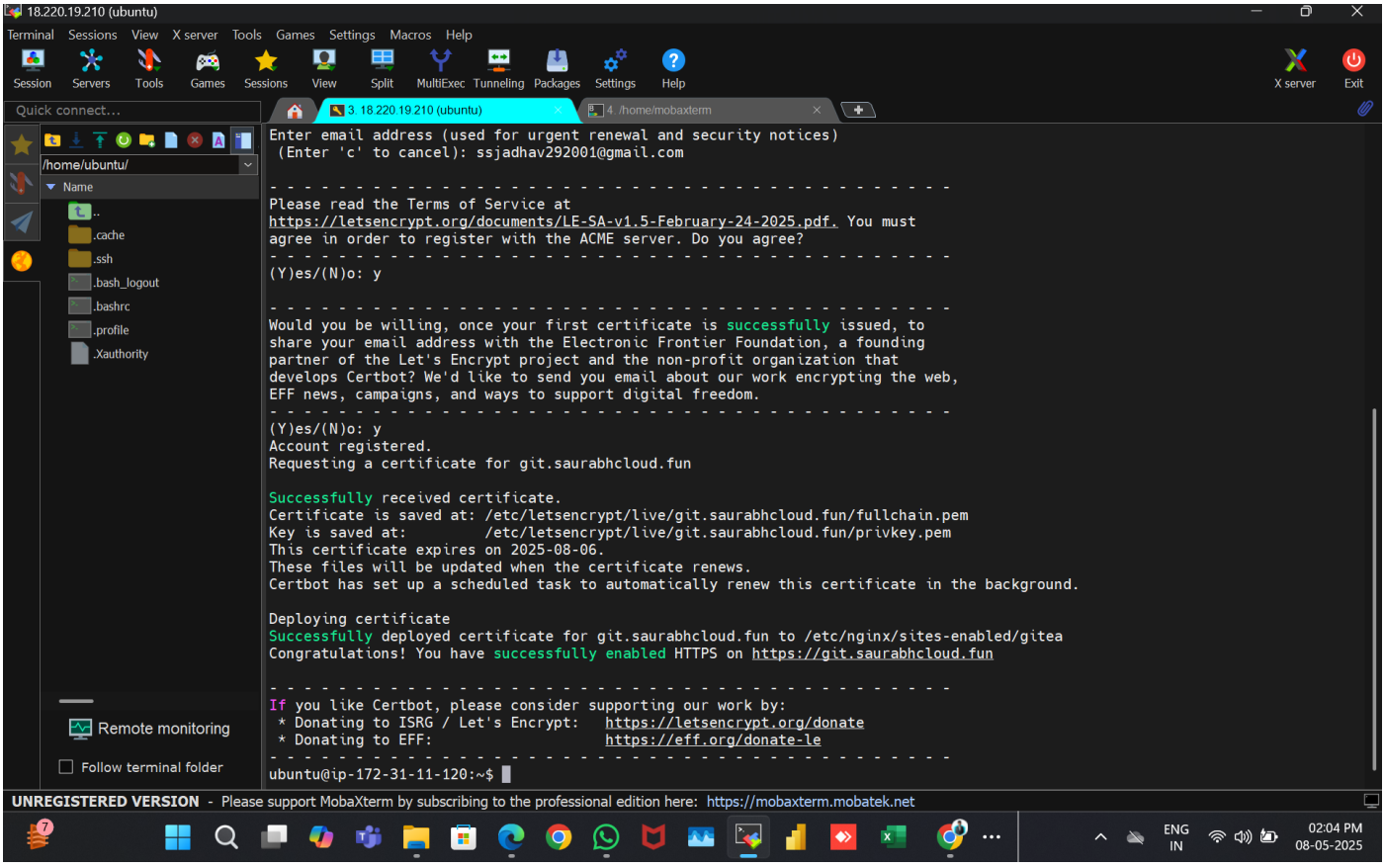
9. Secure Gitea with SSL (Let's Encrypt)

Use Certbot to automatically fetch and configure a free SSL certificate for your domain:

sudo certbot --nginx -d git.saurabhcloud.fun



Certificate Successfully Enabled .

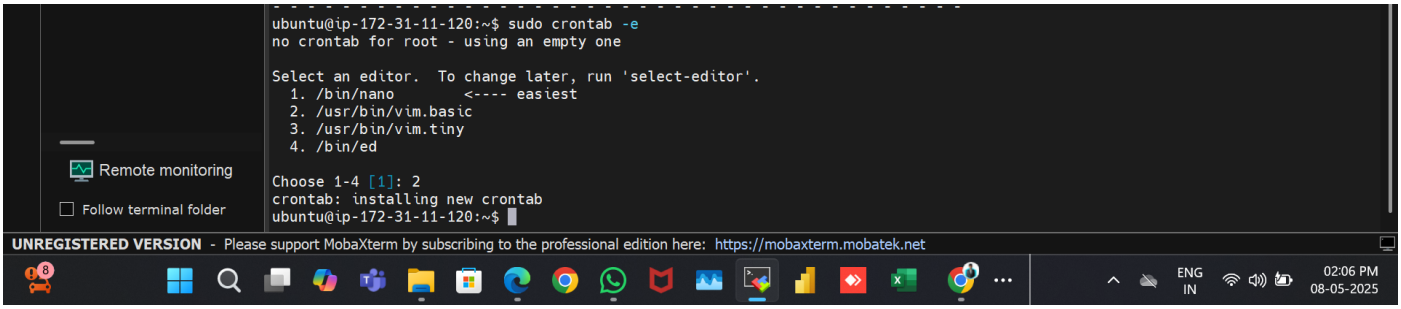


Set up auto-renewal :

Schedule automatic SSL certificate renewal using a cron job:

sudo crontab -e

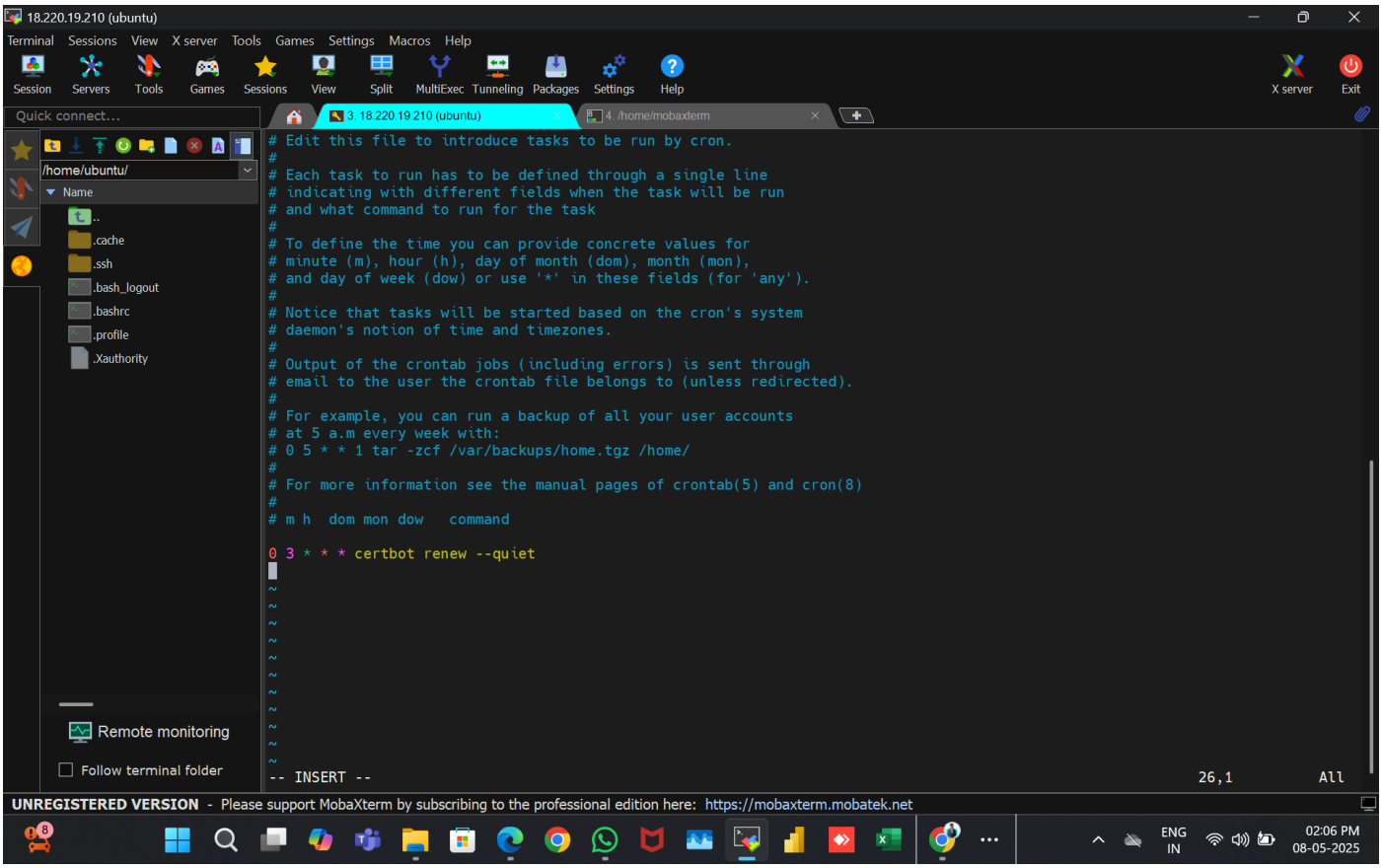
And Enter 2nd for Vim Editor



Add :

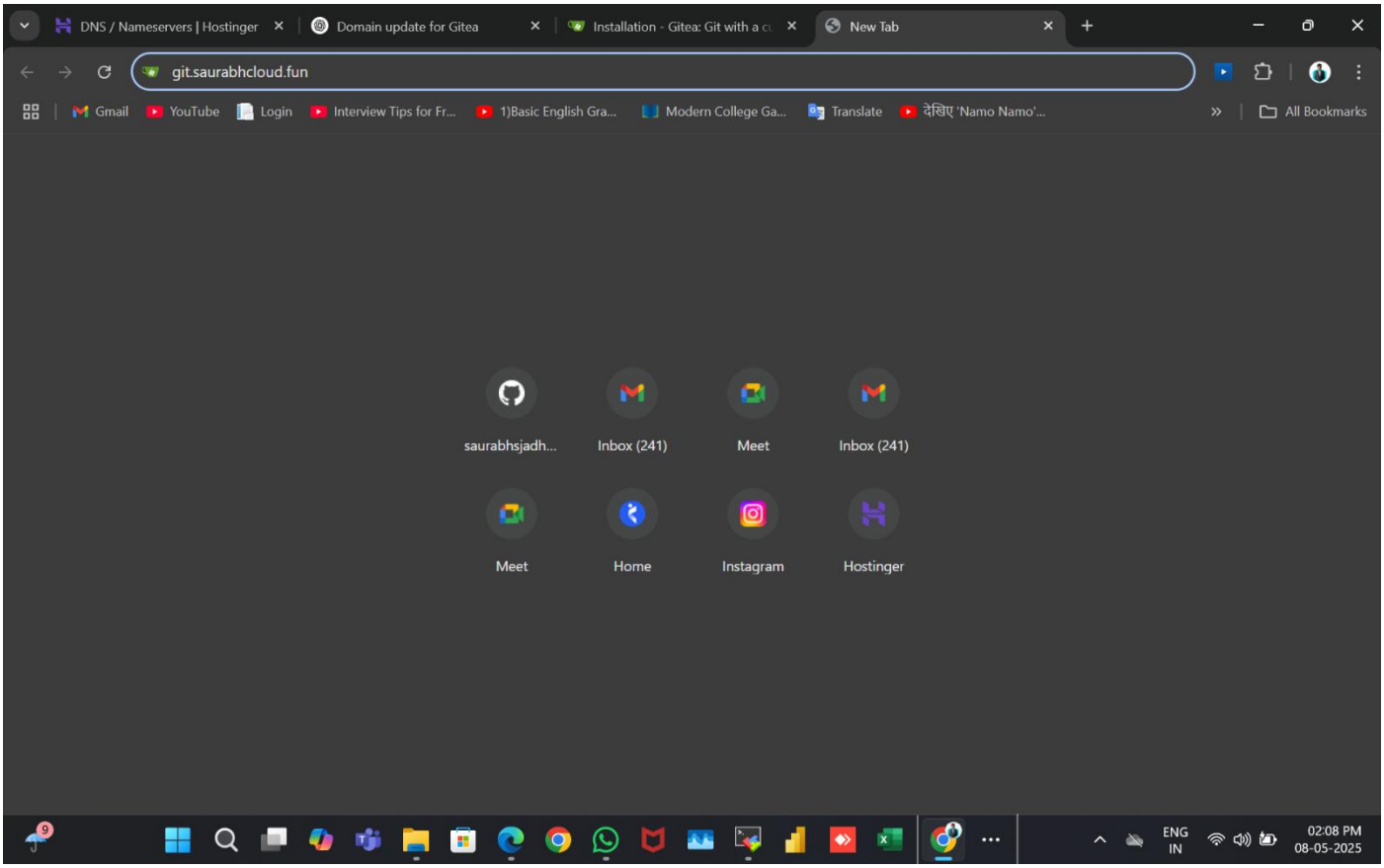
Add the following line to run renewal daily at 3 AM:

0 3 * * * certbot renew –quiet



Open Any Browser and hit your domain name.

Open https://git.saurabhcloud.fun

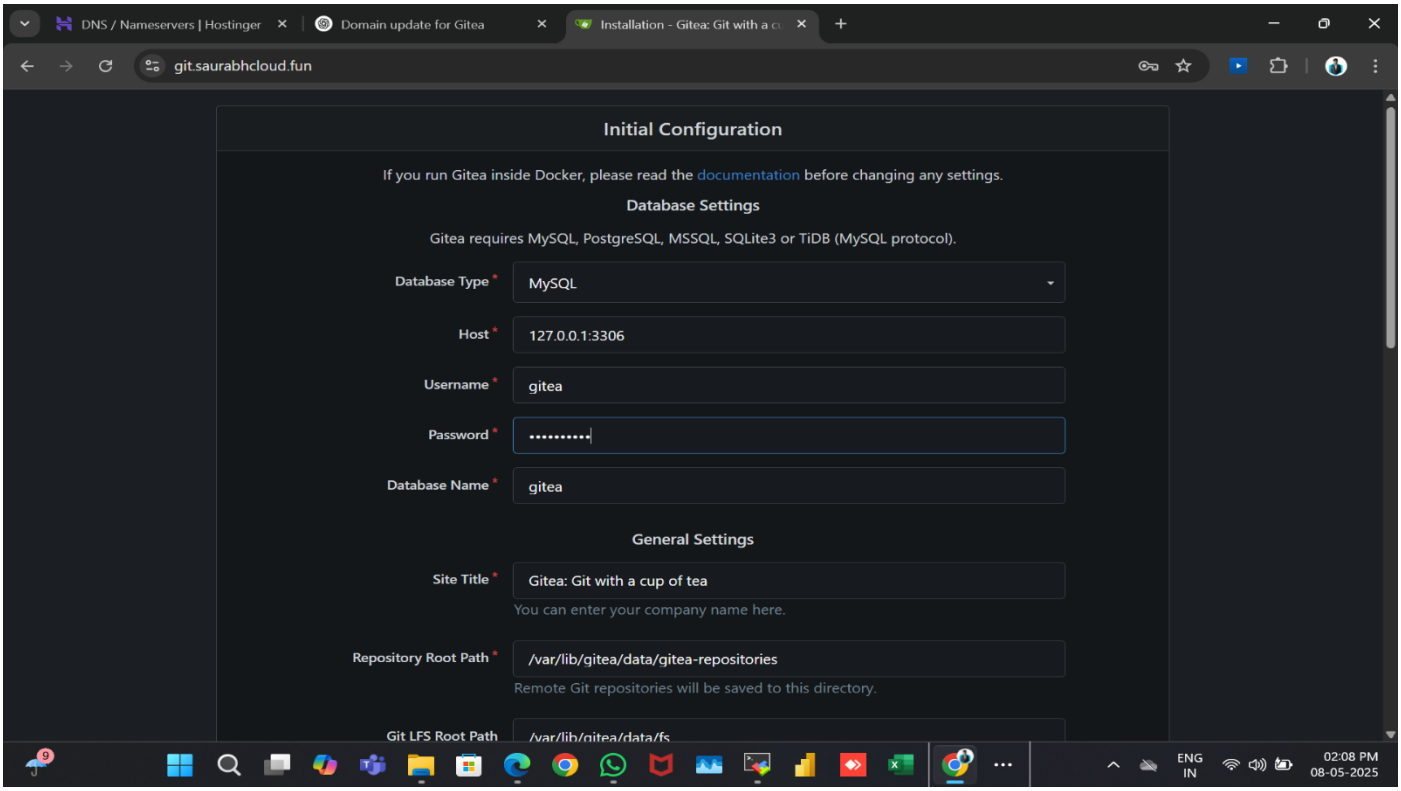


Access Gitea Web Interface

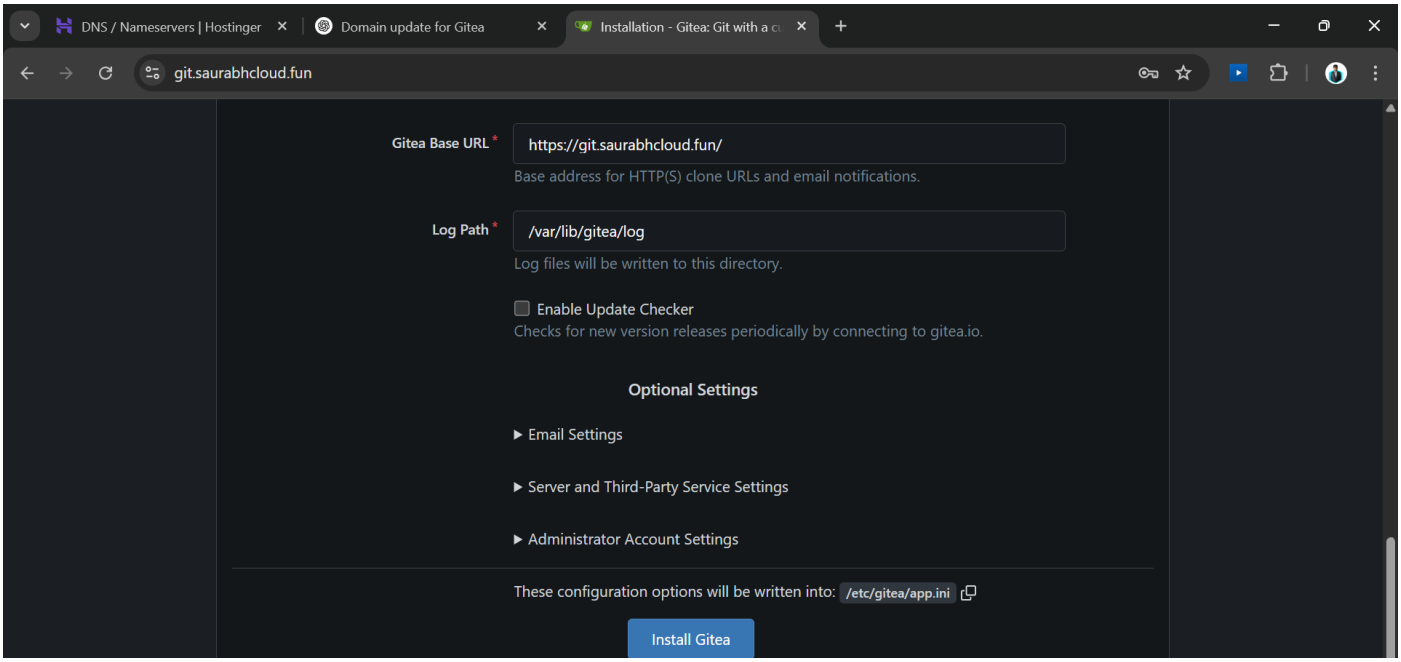
• Complete initial setup

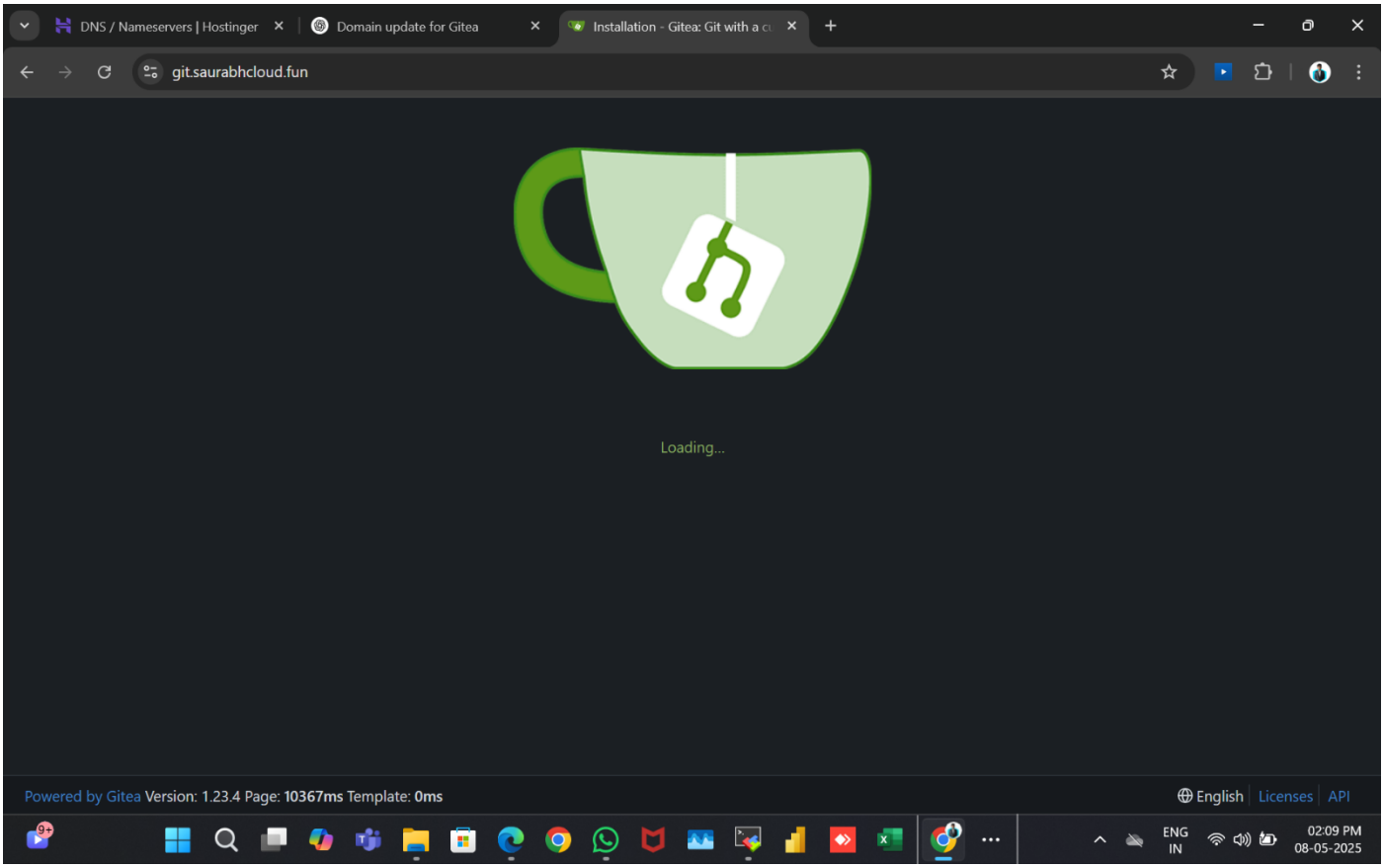
• Use Gitea database credentials

• Create an admin user



Install Gitea





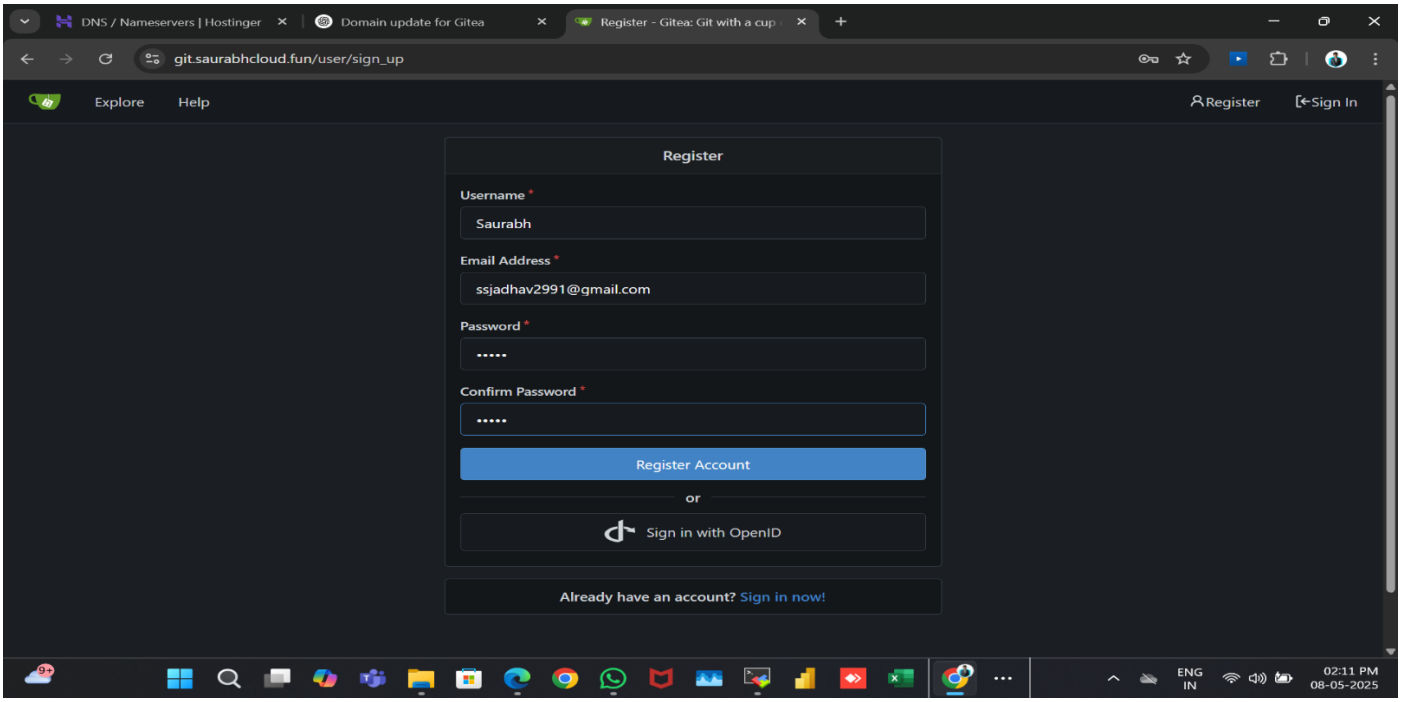
Enable SSH Access for Git Repositories.

Add your current user to the Gitea group to allow SSH-based Git operations, then restart Gitea:

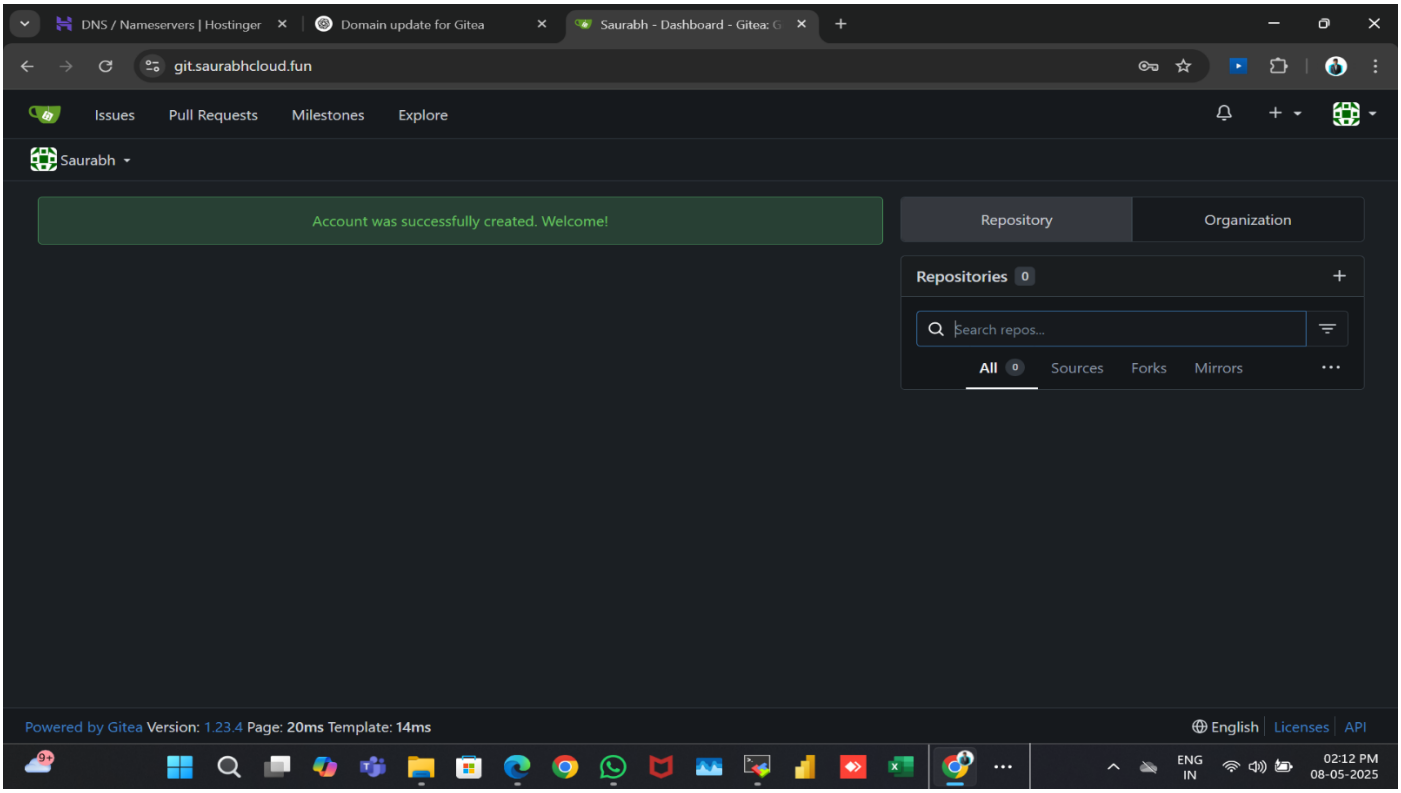
sudo usermod -aG gitea $(whoami)

sudo systemctl restart gitea

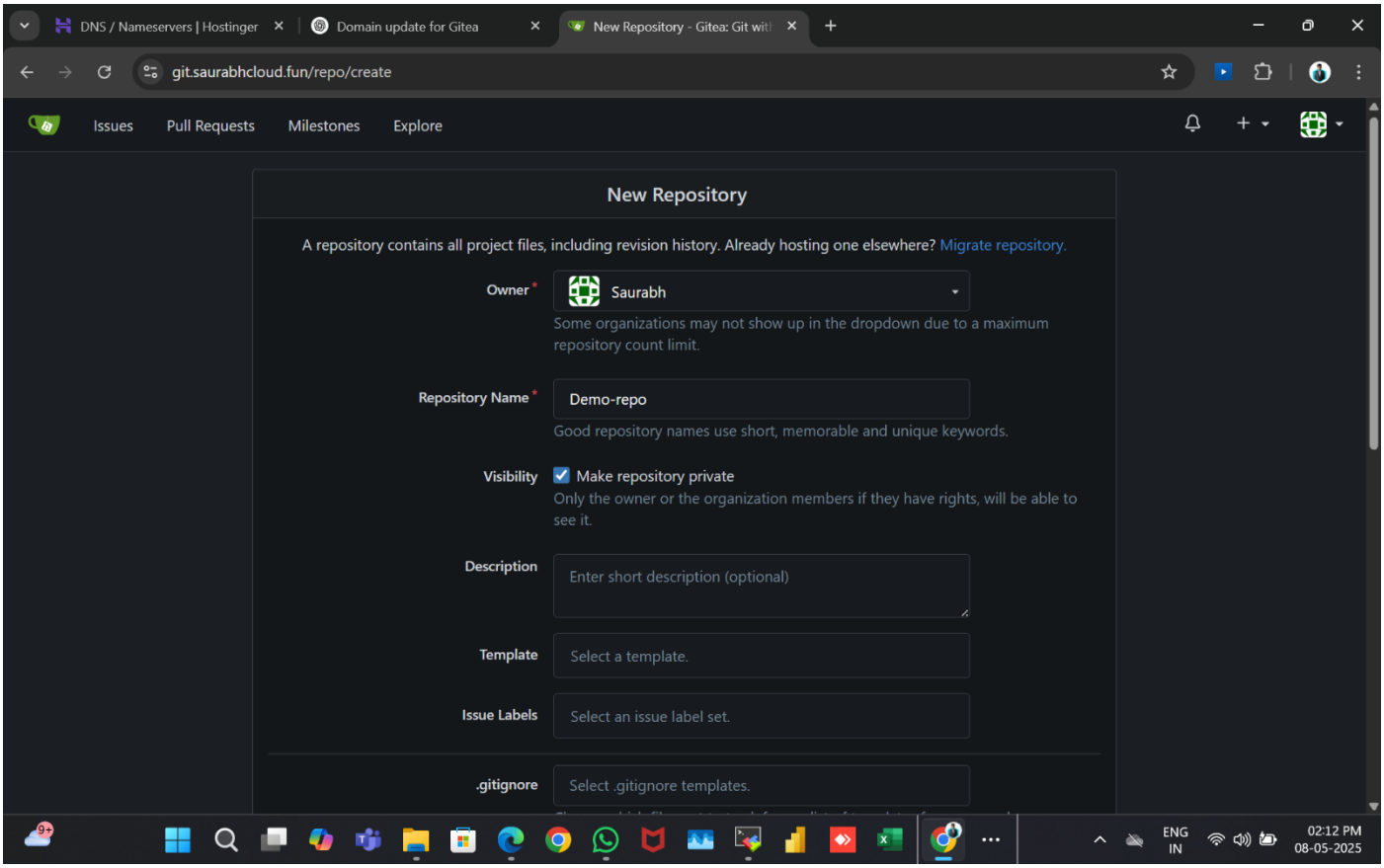
git clone git@git.saurabhcloud.fun:your-saurabhsjadhav/your-repo.git



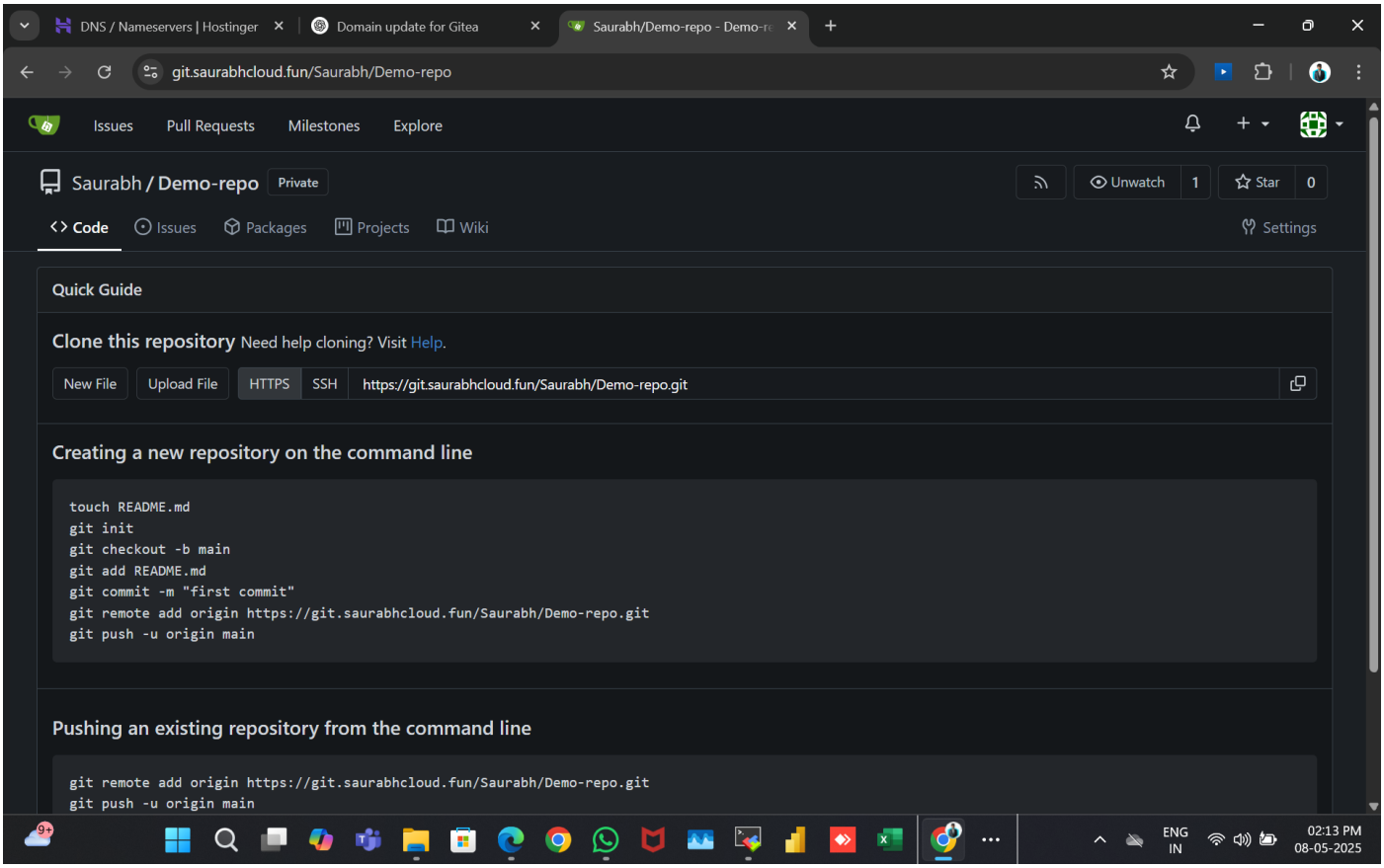
Account Successfully Created.



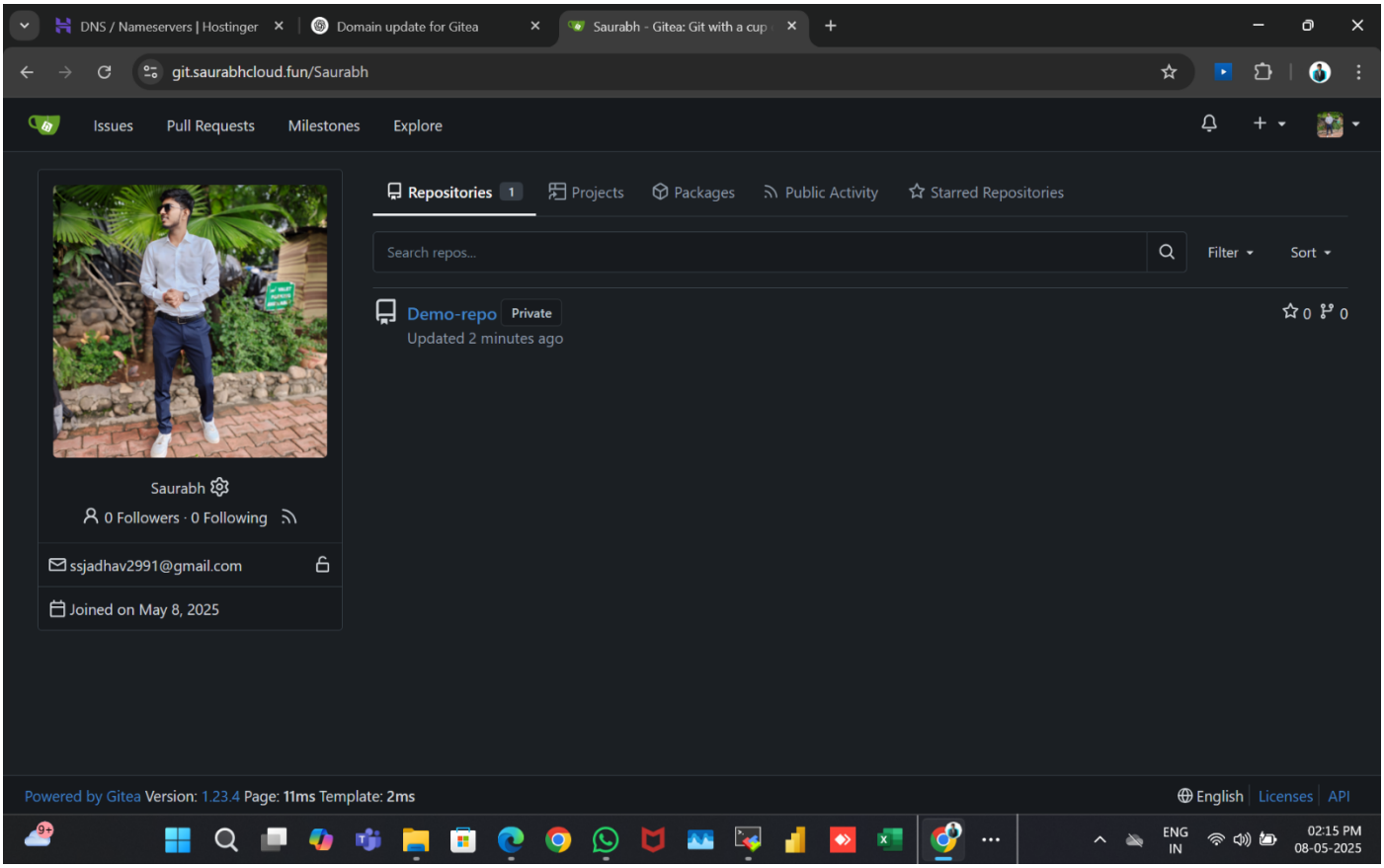
Create a repo.



Your Repo is Successfully Created.



Maintain Your Profile .



Successfully set up a secure, private Git server with Gitea on AWS using your custom domain — ready for powerful team collaboration and code management.

Thank You.