**Intel College Excellence Program   
Project Synopsis**

**“FPT Server”**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team member’s detail** | | | |
| **S. No.** | **Participant Name** | **Mobile No.** | **Email ID** |
| 1 | Chava Likhith | 9550240270 | ***chavalikhith70@gmail.com*** |
| **Faculty(college) mentor detail** | | | |
| **S. No.** | **Mentor Name** | **Mobile No.** | **Email ID** |
| 1 | Dr. Shelej Khera | 9466731632 | ***shelej.22390@lpu.co.in*** |
| **College/University Name** | | | |
| ***Lovely Professional University*** | | | |

**BACKGROUND**

FTP Server around the historical development of the File Transfer Protocol (FTP) and its purpose in enabling file transfers over a network. FTP was created during the nascent stages of network computing. Abhay Bhushan first presented it in 1971 as a component of the initial ARPANET protocols, which served as the forerunner to the internet. The objective was to develop a common protocol that would enable file transfers across a network between machines. FTP's early iterations were made to move files between computers connected to a network by using straightforward commands for uploading, downloading, renaming, deleting, and listing directory contents. RFC 765 established the protocol's standards in 1980, and RFC 959, which was later revised in 1985, is still the fundamental FTP specification to this day. The Fundamentals of FTP has a client-server architecture. The computer that houses the files is known as the FTP server, and the program that connects to the server and transfers files is known as the FTP client. By doing this, a user (client) can upload, download, or edit files that are kept on the distant server. FTP communicates via two distinct connections:

1.Control Connection (Port 21): Used to send commands (such as file requests or login credentials).

2.File transfers between the client and the server are accomplished via the data connection (Port 20 or other random ports).

Initial FTP has Security Issue Because early FTP implementations sent data in plain text, including usernames and passwords, they were vulnerable to being intercepted by hostile actors. A serious security risk resulted from this lack of encryption. More secure FTP versions were created in order to allay these security worries like SSH File Transfer Protocol, or SFTP, is a secure FTP method that encrypts data and commands and Secure file transfers via the FTP protocol are made possible by FTPS (FTP Secure), which is FTP plus SSL/TLS encryption. In order to guarantee safe, encrypted communication over public networks, businesses nowadays frequently favor SFTP or FTPS.

**PROBLEM IDENTIFICATION**

Despite being widely used for file transfers, FTP (File Transfer Protocol) has a number of documented issues and limitations, particularly in contemporary settings. By default, FTP transmits all information in plain text, including passwords and usernames. Because of this, it is extremely susceptible to man-in-the-middle (MITM) and eavesdropping attacks, which allow hackers to obtain confidential files or login passwords. Absence of data encryption Because file contents are not encrypted while being transmitted, if they are intercepted, sensitive information may be accessible to unauthorized parties.

Without extra security features like multi-factor authentication (MFA), basic FTP uses weak authentication techniques (password and username). In order to obtain unauthorized access, attackers may try a number of username/password combinations on FTP servers that are not setup correctly. The client and server establish a data connection, which firewalls may prevent, leading to problems with the connection. Firewall setups and port ranges can make this process more difficult, but the client starts the connection to a port that the server specifies. FTP needs a number of ports for data transfer (port 20 or a range of ports in passive mode) and control (port 21). It can be difficult to manage these ports and set up firewalls for FTP traffic. Users can log in without credentials to certain FTP servers that permit anonymous access. Anonymous FTP access can be a serious security issue if it is not appropriately regulated, enabling anyone to view or upload data without the required authorization. FTP does not automatically check the integrity of files that are sent. FTP does not have checksum verification or other ways to identify when a file is corrupted during transfer (for instance, because of network problems). Modern encryption methods like SSL (Secure Sockets Layer) and TLS (Transport Layer Security), which are now commonplace for safe online interactions, are not supported by the basic version of FTP. Other protocols, such as SFTP (SSH File Transfer Protocol) or FTPS (FTP Secure), must be utilized in order to secure FTP. FTP is inefficient for large file transfers when compared to more recent options like HTTP or cloud-based file-sharing services because it may function slowly over high-latency networks (such overseas connections or mobile networks).

**PROPOSED SOLUTION**

A number of fixes and best practices can be used to improve security, performance, and usability in order to solve the different issues related to FTP servers.

**Make Use of SFTP and FTP Secure:** Make use of SFTP (SSH File Transfer Protocol) to protect login credentials and file data by encrypting both the control and data connections. FTPS secures the communication channels by encrypting FTP using SSL/TLS. Additionally, it has the capability of client and server authentication. Use encryption to stop the transmission of plain text, encrypt all FTP connections.

**Turn Off Anonymous FTP Access:** Make sure that only authenticated users have access, and always turn off anonymous logins unless absolutely required. all user accounts have strong password requirements, particularly those pertaining to password complexity and expiration dates. Limit the number of log-in attempts to lessen the impact of brute-force attacks. increase security by requiring a second authentication factor to be used in order to get access, even in the event that a password is hacked.

**Implement Passive Mode Properly:** Set up the FTP server to operate in passive mode and specify a range of ports for data connections. Verify that routers and firewalls are set up to permit these ports. SFTP is firewall-friendly and eliminates the hassle of managing several ports by using a single connection via port 22, which is also used by SSH.

**Anonymous Access:** Turn off anonymous access and restrict access to the FTP server to authorized users only. Apply stringent read-only permissions and restrict access to particular directories if anonymous access is required.

**Use Checksum Verification:** Create checksums of files both before and after transfer by employing hash functions (such as MD5, SHA256, etc.) to provide external integrity checks. By doing this, users may confirm that the transferred file is identical to the original.

**Disable Plain FTP:** Make sure that all transfers are made using FTPS or SFTP and specifically turn off the server's support for plain FTP (unencrypted connections).

**Setup Essentials**

**Hardware Requirements:**

1. Server Machine.
2. Network Equipment.
3. Power Backup (**Uninterruptible Power Supply**).
4. Redundancy and Backups (**Redundant Array of Independent Disks**).

**Software Requirements:**

1. Operating System.
2. FTP Server Software.
3. Security Software.
4. User Management and Authentication.
5. Logging and Monitoring.
6. Port and Firewall Configuration.

**DESCRIPTION**

**FTP Server**

An FTP server is a software application that runs on a physical or virtual machine, allowing users to transfer files between the server and client devices over a network (local area network or internet). It uses the FTP protocol, which follows a client-server architecture. Clients connect to the FTP server using FTP client software, enabling them to download or upload files, manage directories, or perform file operations.

**Table of Contents**

1. Step 1: Installing Operating System (Ubuntu 20.04).

2. Step 2: System Update.

3. Step 3: Installing Packages Like VSFTPD.

4. Step 4: Configure VSFTPD.

5. Step 5: Enable and disable Few options while Configure VSFTPD.

6. Step 6: Restart Service.

7. Step 7: Allow FTP Traffic Through the Firewall.

8. Step 8: Create the FTP user and Password.

9. Step 9: Make a directory.

10. Step 10: Test the FTP server.

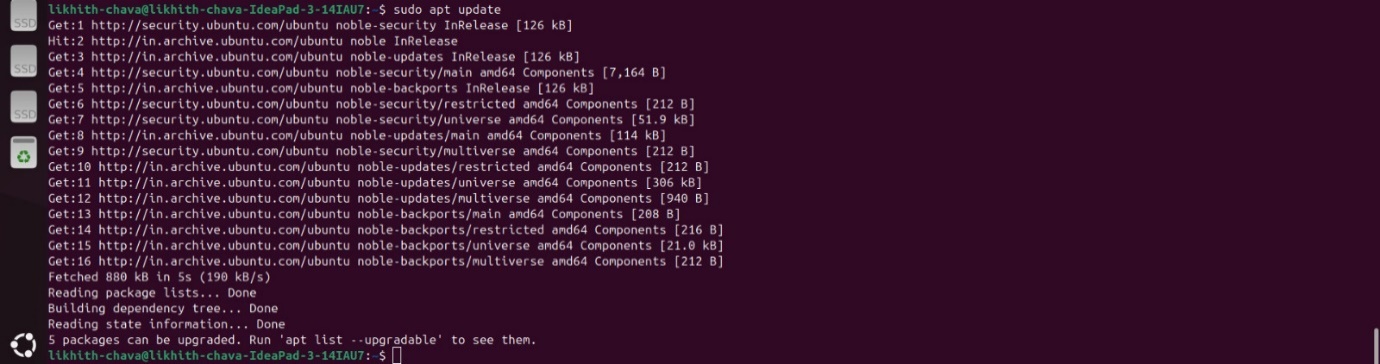
11. Step 11: Install FileZilla.

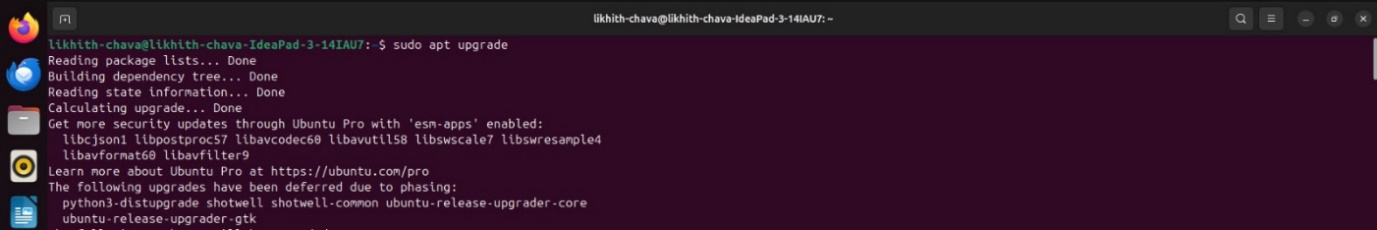
12. Step 12: Add I.P address, Host and Password in filezilla.

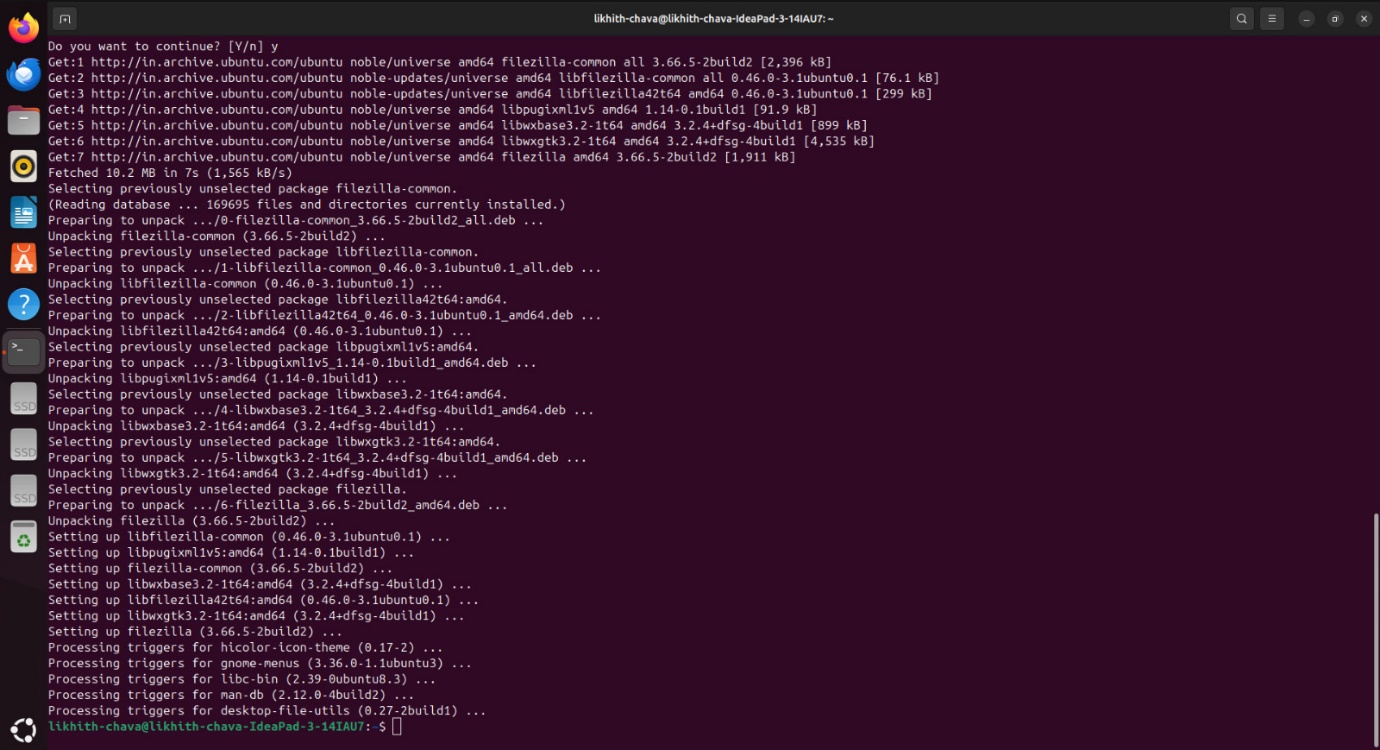
**Step 2: System Update:**

Make sure your system is up to date before installing the VSFTPD.

* Sudo apt update.
* Sudo apt upgrade.



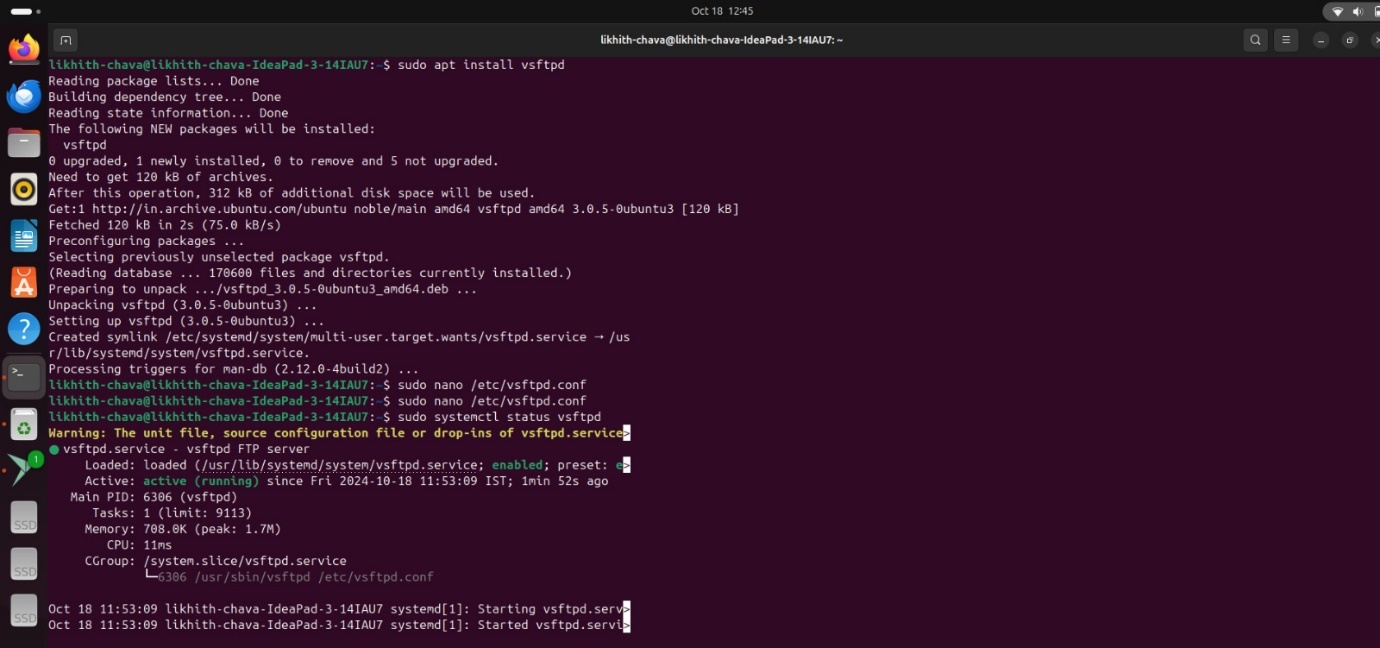
****



**Step 3: Install Packages:**

install VSFTPD (Very Secure FTP Daemon) to set up an FTP server on a Linux. VSFTPD enables the server to handle FTP connections, allowing users to upload, download, and manage files on the server over a network, whether it be locally or over the internet. With an FTP server, you can access files remotely from different devices, making it easier to share files between clients and the server. Users can connect via an FTP client (like FileZilla etc.) to upload or download files from anywhere.

* Sudo apt install vsftpd.
* Sudo systemctl status vsftpd.

****

**Step 4: Configure the Packages:**

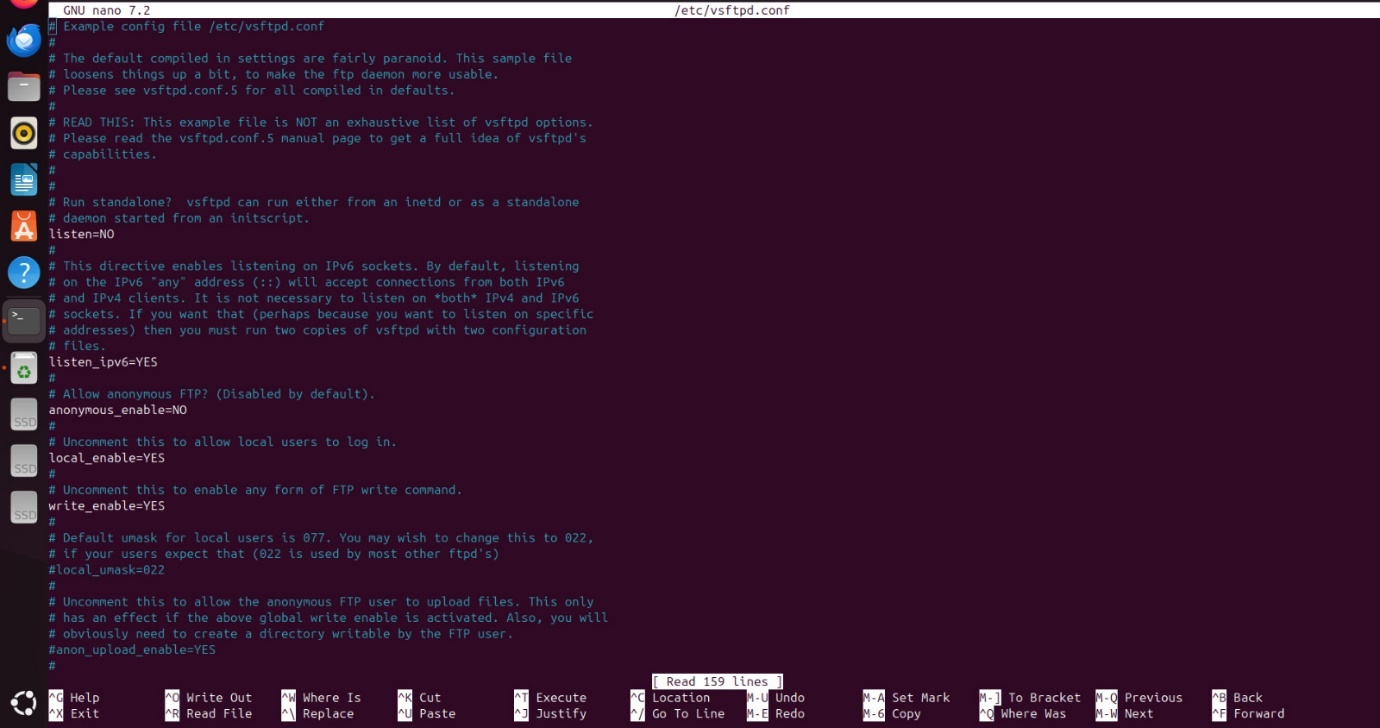
* Sudo nano /etc/vsftpd.conf.

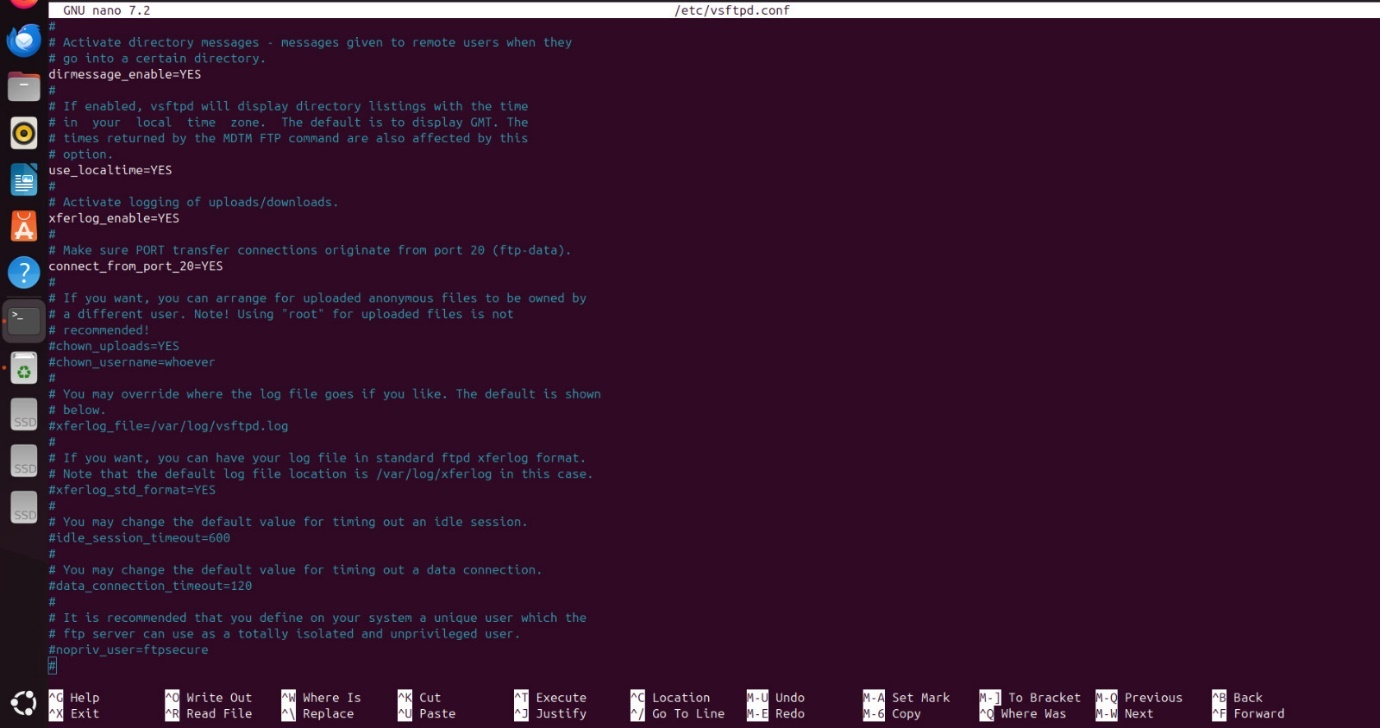
The command Sudo nano /etc/vsftpd.conf is used to open and edit the configuration file for the vsftpd FTP server using the nano text editor, with superuser privileges.

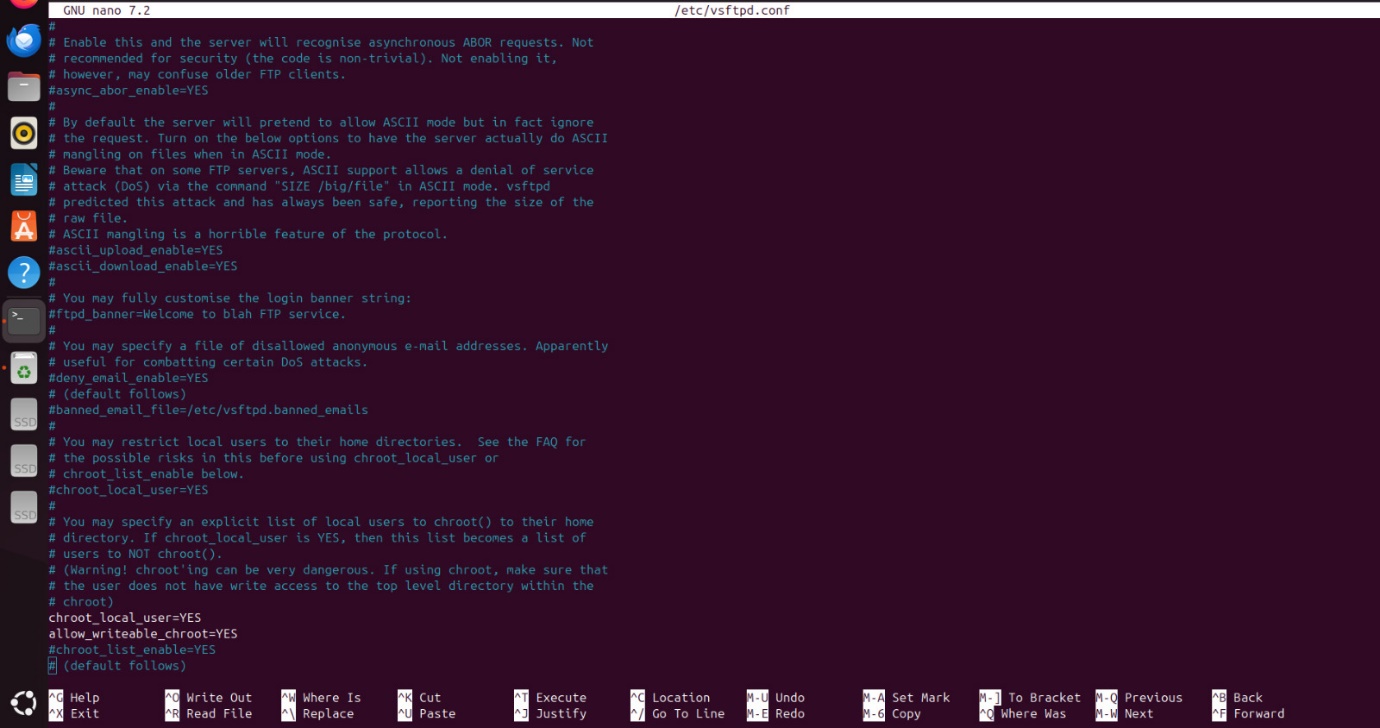


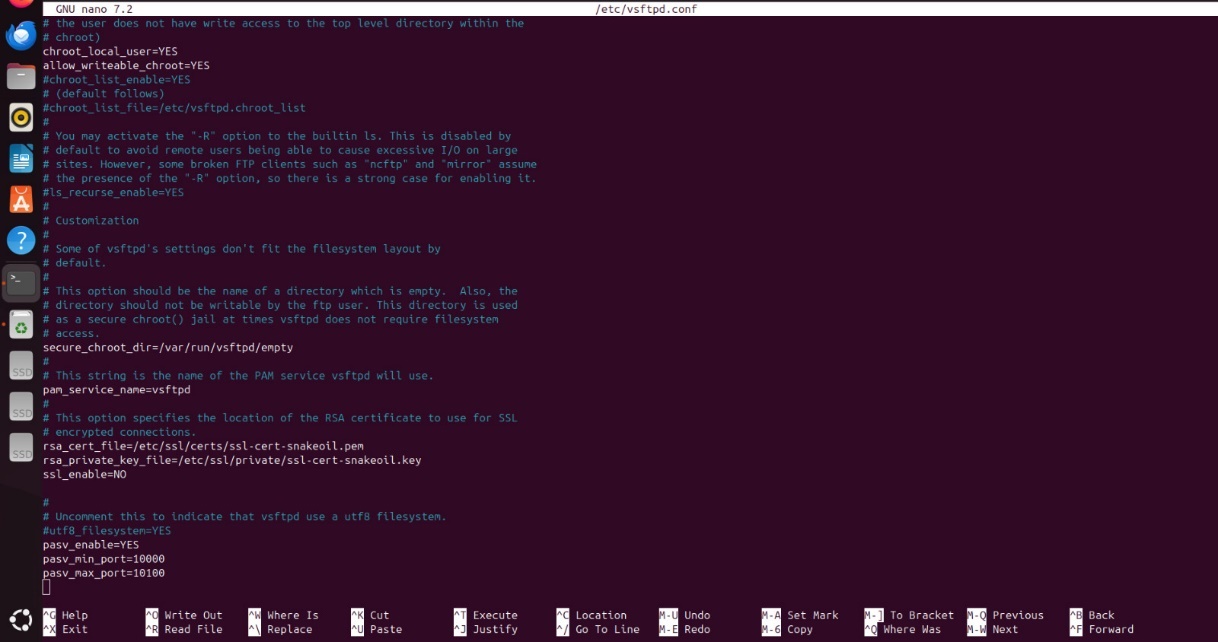
**Step 5: Enable and disable Few options while Configure VSFTPD:**

1. anonymous\_enable=NO: Disables anonymous FTP access to prevent unauthenticated users from logging in.
2. local\_enable=YES: Allows local users (with system accounts) to log in to the FTP server.
3. write\_enable=YES: Permits file uploads and modifications by users.
4. chroot\_local\_user=YES: Restricts local users to their home directories, enhancing security by preventing access to other system files.
5. pasv\_enable=YES: Enables passive mode for FTP, which is useful for clients behind firewalls.
6. pasv\_min\_port=10000: Sets the minimum port for passive FTP connections, used for data transfer.
7. pasv\_max\_port=10100: Sets the maximum port for passive FTP connections, defining a controlled range for better firewall configuration.

****

****

****

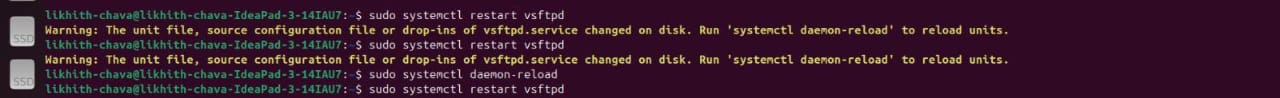
****

**Step 6: Restart Service:**

After making modifications to the **VSFTPD** configuration file (e.g., /etc/vsftpd.conf), you need to restart the service for the changes to take effect.

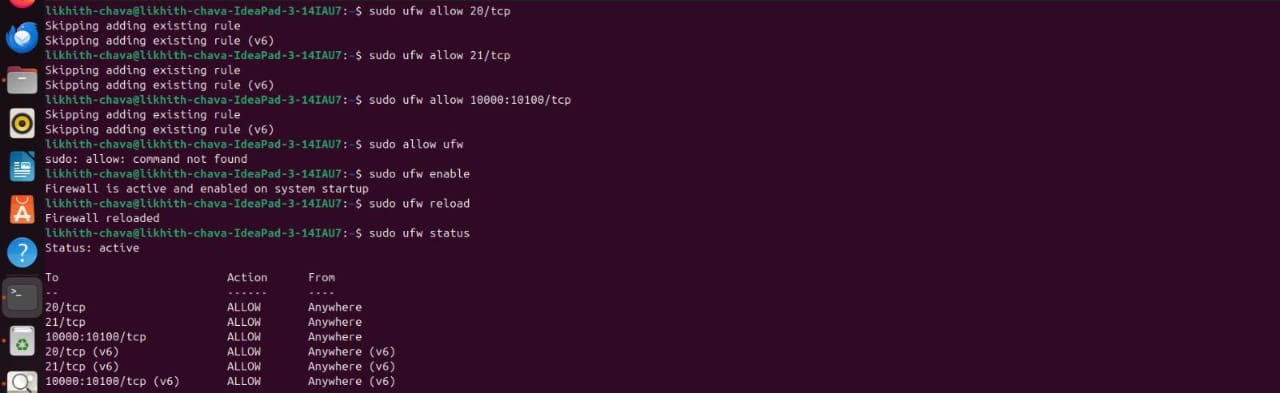
make changes to service unit files or system configurations (like adding, modifying, or deleting service files), system needs to reload its configuration to apply those changes.

* Sudo systemctl daemon-reload
* Sudo systemctl restart vsftpd



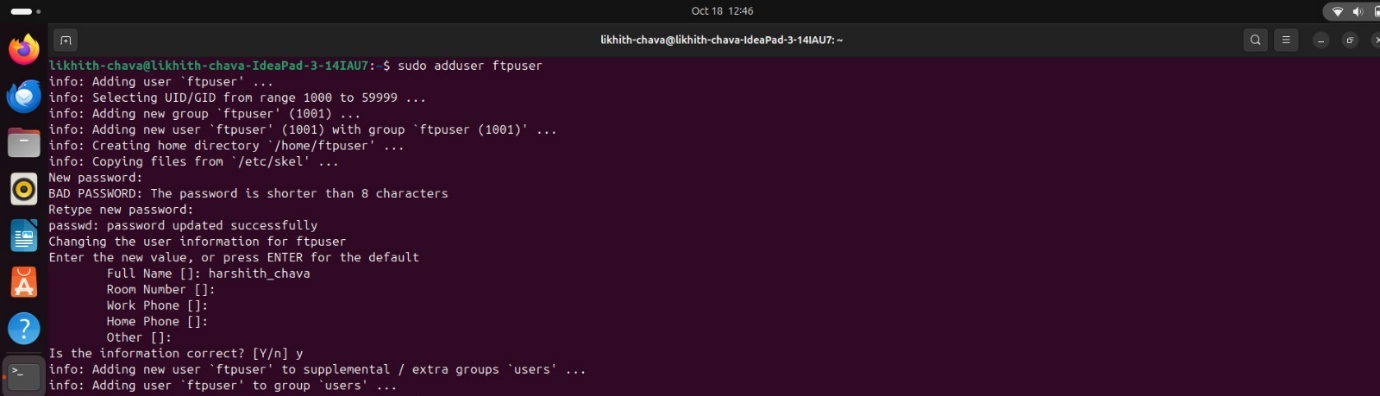
**Step 7: Allow FTP Traffic Through the Firewall:**

* sudo ufw allow 20/tcp: Allows incoming traffic on port 20 (FTP data transfer port) through the firewall.
* sudo ufw allow 21/tcp: Allows incoming traffic on port 21 (FTP control port) through the firewall.
* sudo ufw allow 10000:10100/tcp: Opens ports 10000 to 10100 for passive mode FTP data transfers.
* sudo allow ufw: This command is incorrect—possibly a typo, as "ufw" itself manages firewall rules.
* sudo ufw enable: Enables the Uncomplicated Firewall (UFW), activating the firewall on your system.
* sudo ufw reload: Reloads the UFW configuration to apply any changes made to firewall rules.
* sudo ufw status: Displays the current status and active firewall rules on your system.

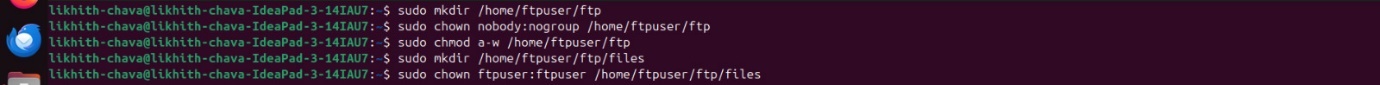


**Step 8:** **Create the FTP user and Password:**

* **sudo adduser ftpuser**

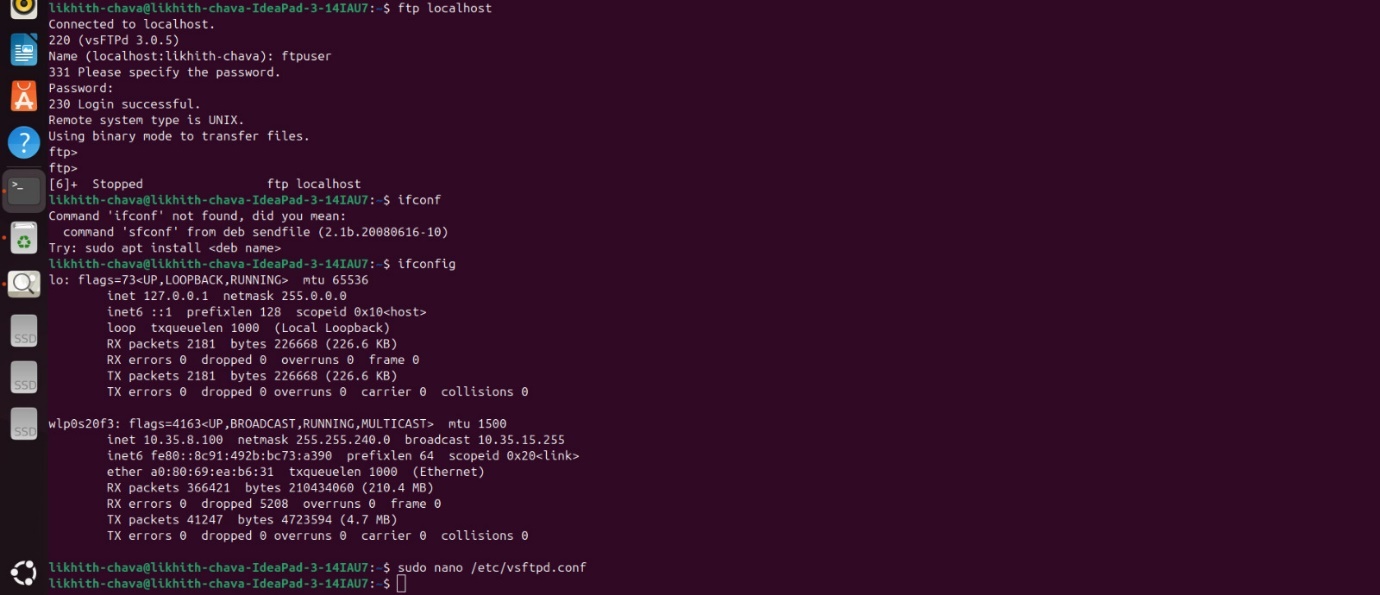


**9. Step 9: Make a directory:**



**10. Step 10: Test the FTP server:**

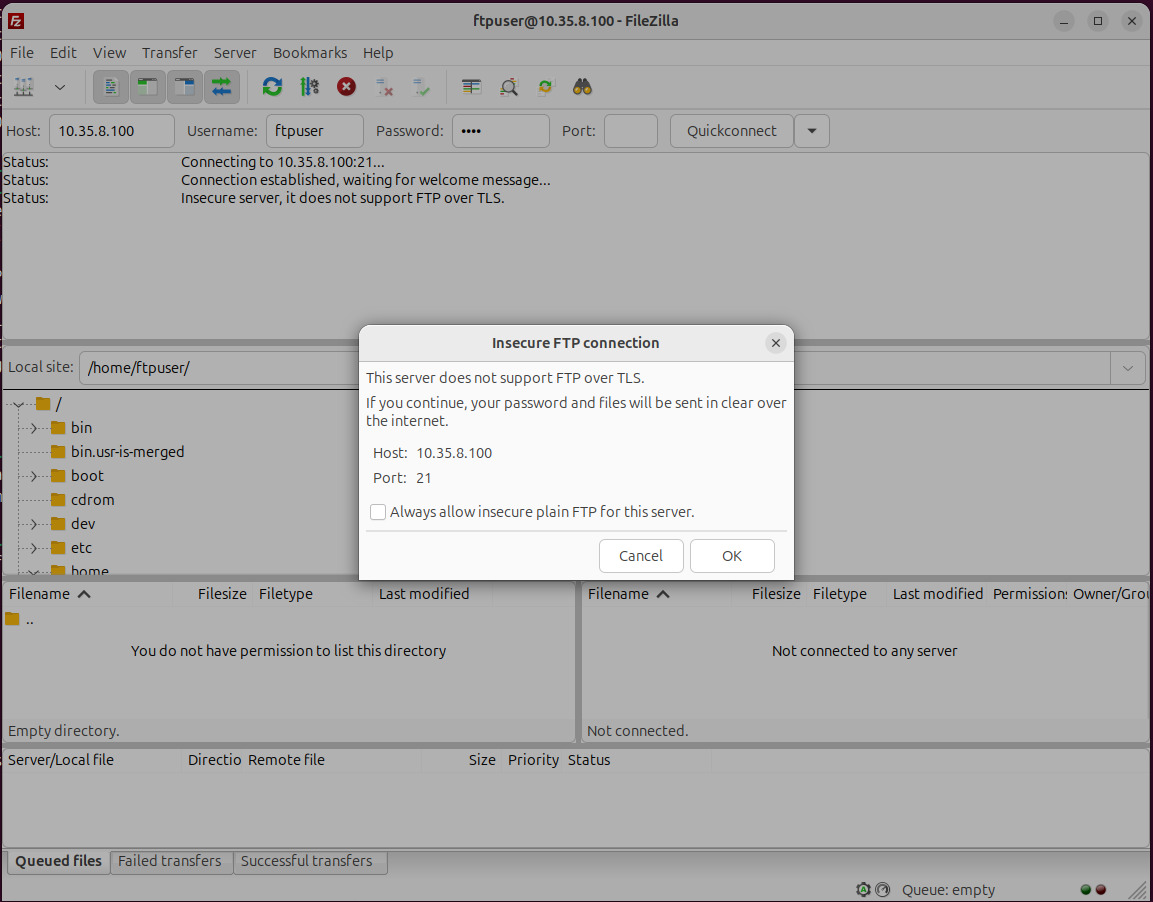
* **ftp localhost**
* **ifconfig**

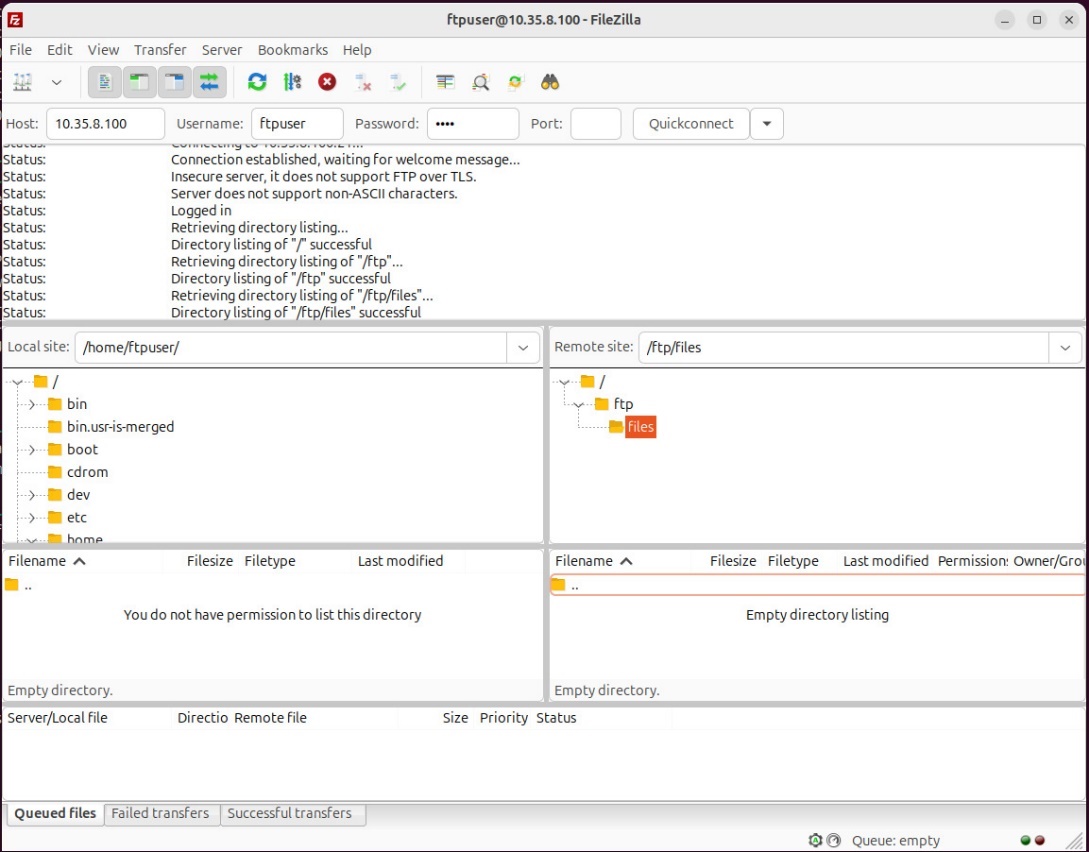


**Step 11: Install FileZilla:**

Install FileZilla using the Command “sudo apt install filezilla”

**12. Step 12: Add I.P address, Host and Password in filezilla:**





**FUTURE SCOPE**

In the future, the protocol for modern use cases and addressing its inherent limitations in terms of security, performance, and scalability. Despite being one of the oldest protocols for file transfers, FTP continues to evolve, though modern organizations often favor more secure alternatives like SFTP and FTPS. However, FTP still plays an essential role in various applications, and several trends and future developments could shape its scope. SFTP offers encrypted file transfers over SSH, and its use is expected to increase for organizations concerned with data privacy and security. FTPS version of FTP uses SSL/TLS to encrypt connections. As more organizations focus on compliance (GDPR, HIPAA), the demand for secure file transfers using FTPS will grow. FTP servers could act as gateways, allowing legacy systems to connect to modern cloud services like AWS S3, Google Cloud, or Microsoft Azure. Lightweight FTP servers may be used on edge devices to enable localized data management and efficient file transfers to the cloud or data centers.

**CONCLUSION**

In conclusion, The FTP server project demonstrates the effective setup and configuration of an FTP server using vsftpd on a Linux-based system. Through a detailed process involving user management, firewall configuration, and file permissions, the project highlights how FTP can be used to securely transfer files between local and remote machines. Despite its limitations, such as the lack of encryption in standard FTP, using secure variants like FTPS and SFTP can address these concerns, ensuring data protection and compliance with modern security standards. By carefully managing access controls and setting up the system for efficient file transfers, this project underscores FTP's ongoing relevance in data exchange, particularly in controlled environments. this project offers a foundation for future developments, including cloud integration, automation, and enhanced security. With the growing focus on secure, scalable, and efficient data transfer, FTP servers can be extended or integrated into larger network architectures, supporting hybrid cloud environments and edge computing. Although newer technologies are often preferred for secure data transfers, FTP's simplicity and reliability continue to make it a valuable tool, especially in legacy systems or environments where speed and ease of use are prioritized. This project serves as a practical demonstration of FTP’s capabilities while acknowledging the need for future adaptations in a rapidly evolving technological landscape.

**REFERENCES**

Git-hub link:

https://github.com/LikhithChava/Likhith\_5th\_sem\_project