**File Server**

**INTRODUCTION**

**What is File server?**

* A file server is a computer attached to a network that has the primary purpose of providing a location for shared disk access, i.e. shared storage of computer files (such as documents, sound files, photographs, movies, images, databases, etc.) that can be accessed by the workstations that are attached to the same computer network.
* The term server highlights the role of the machine in the client–server scheme, where the clients are the workstations using the storage.
* A file server is not intended to perform computational tasks, and does not run programs on behalf of its clients.
* It is designed primarily to enable the storage and retrieval of data while the computation is carried out by the workstations.

**Types of file servers**

* A file server may be dedicated or non-dedicated. A dedicated server is designed specifically for use as a file server, with workstations attached for reading and writing files and databases.
* File servers may also be categorized by the method of access: Internet file servers are frequently accessed by File Transfer Protocol (FTP) or by HTTP (but are different from web servers, that often provide dynamic web content in addition to static files). Servers on a LAN are usually accessed by SMB/CIFS protocol (Windows and Unix-like) or NFS protocol (Unix-like systems).
* Database servers, that provide access to a shared database via a database device driver, are not regarded as file servers as they may require Record locking.

**Design of file servers**

* In modern businesses, the design of file servers is complicated by competing demands for storage space, access speed, recoverability, ease of administration, security, and budget.
* This is further complicated by a constantly changing environment, where new hardware and technology rapidly obsolesces old equipment, and yet must seamlessly come online in a fashion

compatible with the older machinery.

* To manage throughput, peak loads, and response time, vendors may utilize queuing theory to model how the combination of hardware and software will respond over various levels of demand.
* Servers may also employ dynamic load balancing scheme to distribute requests across various pieces of hardware.
* The primary piece of hardware equipment for servers over the last couple of decades has proven to be the hard disk drive.
* Although other forms of storage are viable (such as magnetic tape and solid-state drives) disk drives have continued to offer the best fit for cost, performance, and capacity.

**Storage**

* Since the crucial function of a file server is storage, technology has been developed to operate multiple disk drives together as a team, forming a disk array.
* A disk array typically has cache (temporary memory storage that is faster than the magnetic disks), as well as advanced functions like RAID and storage virtualization.
* Typically, disk arrays increase level of availability by using redundant components other than RAID, such as power supplies. Disk arrays may be consolidated or virtualized in a SAN.

**Network-attached storage**

* Network-attached storage (NAS) is file-level computer data storage connected to a computer network providing data access to a heterogeneous group of clients.
* NAS devices specifically are distinguished from file servers generally in a NAS being a computer appliance – a specialized computer built from the ground up for serving files – rather than a general-purpose computer being used for serving files (possibly with other functions).
* In discussions of NASs, the term "file server" generally stands for a contrasting term, referring to general purpose computers only.
* As of 2010 NAS devices are gaining popularity, offering a convenient method for sharing files between multiple computers.
* Potential benefits of network-attached storage, compared to non-dedicated file servers, include faster data access, easier administration, and simple configuration.
* NAS systems are networked appliances containing one or more hard drives, often arranged into logical, redundant storage containers or RAID arrays.
* Network Attached Storage removes the responsibility of file serving from other servers on the network.
* They typically provide access to files using network file sharing protocols such as NFS, SMB/CIFS (Server Message Block/Common Internet File System), or AFP.

**Security**

* File servers generally offer some form of system security to limit access to files to specific users or groups.
* In large organizations, this is a task usually delegated to what is known as directory services such as openLDAP, Novell's eDirectory or Microsoft's Active Directory.
* These servers work within the hierarchical computing environment which treat users, computers, applications and files as distinct but related entities on the network and grant access based on user or group credentials.
* In many cases, the directory service spans many file servers, potentially hundreds for large organizations. In the past, and in smaller organizations, authentication could take place directly at the server itself.



**File Transfer Protocol**

* The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files from a server to a client using the Client–server model on a computer network.
* FTP is built on a client-server model architecture and uses separate control and data connections between the client and the server.
* FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it.
* For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS). SSH File Transfer Protocol (SFTP) is sometimes also used instead, but is technologically different.
* The first FTP client applications were command-line programs developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix and Linux operating systems. Many FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware.

**Protocol overview**

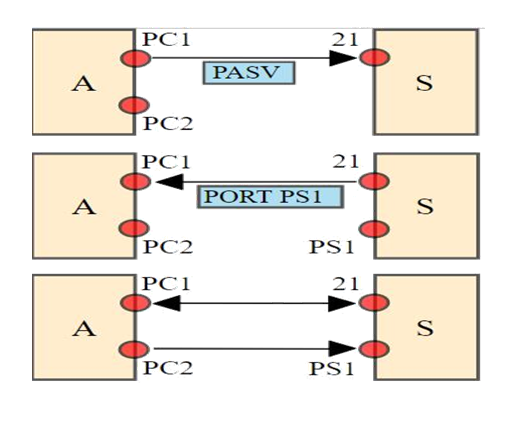
**Communication and data transfer**

FTP may run in active or passive mode, which determines how the data connection is established. In both cases, the client creates a TCP control connection from a random, usually an unprivileged, port N to the FTP server command port 21.

* In active mode, the client starts listening for incoming data connections from the server on port M. It sends the FTP command PORT M to inform the server on which port it is listening. The server then initiates a data channel to the client from its port 20, the FTP server data port.
* In situations where the client is behind a firewall and unable to accept incoming TCP connections, passive mode may be used. In this mode, the client uses the control connection to send a PASV command to the server and then receives a server IP address and server port number from the server which the client then uses to open a data connection from an arbitrary client port to the server IP address and server port number received.

Both modes were updated in September 1998 to support IPv6. Further changes were introduced to the passive mode at that time, updating it to extended passive mode.

The server responds over the control connection with three-digit status codes in ASCII with an optional text message. For example, "200" (or "200 OK") means that the last command was successful. The numbers represent the code for the response and the optional text represents a human-readable explanation or request (e.g. <Need account for storing file>).An ongoing transfer of file data over the data connection can be aborted using an interrupt message sent over the control connection.



**While transferring data over the network, four data representations can be used:**

* ASCII mode: Used for text. Data is converted, if needed, from the sending host's character representation to "8-bit ASCII" before transmission, and (again, if necessary) to the receiving host's character representation. As a consequence, this mode is inappropriate for files that contain data other than plain text.
* Image mode (commonly called Binary mode): The sending machine sends each file byte by byte, and the recipient stores the bitstream as it receives it. (Image mode support has been recommended for all implementations of FTP).
* EBCDIC mode: Used for plain text between hosts using the EBCDIC character set.
* Local mode: Allows two computers with identical setups to send data in a proprietary format without the need to convert it to ASCII.
* For text files, different format control and record structure options are provided. These features were designed to facilitate files containing Telnet or ASA.

**Data transfer can be done in any of three modes:**

* Stream mode: Data is sent as a continuous stream, relieving FTP from doing any processing. Rather, all processing is left up to TCP. No End-of-file indicator is needed, unless the data is divided into records.
* Block mode: FTP breaks the data into several blocks (block header, byte count, and data field) and then passes it on to TCP.
* Compressed mode: Data is compressed using a simple algorithm (usually run-length encoding).
* Some FTP software also implements a DEFLATE-based compressed mode, sometimes called "Mode Z" after the command that enables it. This mode was described in an Internet Draft, but not standardized.

**Login**

* FTP login utilizes a normal username and password scheme for granting access.
* The username is sent to the server using the USER command, and the password is sent using the PASS command.
* If the information provided by the client is accepted by the server, the server will send a greeting to the client and the session will commence.
* If the server supports it, users may log in without providing login credentials, but the same server may authorize only limited access for such sessions.

**Anonymous FTP**

* A host that provides an FTP service may provide anonymous FTP access. Users typically log into the service with an 'anonymous' (lower-case and case-sensitive in some FTP servers) account when prompted for user name.
* Although users are commonly asked to send their email address instead of a password, no verification is actually performed on the supplied data.
* Many FTP hosts whose purpose is to provide software updates will allow anonymous logins.

**NAT and firewall traversal**

* FTP normally transfers data by having the server connect back to the client, after the PORT command is sent by the client.
* This is problematic for both NATs and firewalls, which do not allow connections from the Internet towards internal hosts.
* For NATs, an additional complication is that the representation of the IP addresses and port number in the PORT command refer to the internal host's IP address and port, rather than the public IP address and port of the NAT.
* There are two approaches to this problem. One is that the FTP client and FTP server use the PASV command, which causes the data connection to be established from the FTP client to the server.
* This is widely used by modern FTP clients. Another approach is for the NAT to alter the values of the PORT command, using an application-level gateway for this purpose.

**Differences from HTTP**

* HTTP essentially fixes the bugs in FTP that made it inconvenient to use for many small ephemeral transfers as are typical in web pages.
* FTP has a state full control connection which maintains a current working directory and other flags, and each transfer requires a secondary connection through which the data are transferred. In "passive" mode this secondary connection is from client to server, whereas in the default "active" mode this connection is from server to client.
* This apparent role reversal when in active mode, and random port numbers for all transfers, is why firewalls and NAT gateways have such a hard time with FTP.
* HTTP is stateless and multiplexes control and data over a single connection from client to server on well-known port numbers, which trivially passes through NAT gateways and is simple for firewalls to manage.
* Setting up an FTP control connection is quite slow due to the round-trip delays of sending all of the required commands and awaiting responses, so it is customary to bring up a control connection and hold it open for multiple file transfers rather than drop and re-establish the session afresh each time.
* In contrast, HTTP originally dropped the connection after each transfer because doing so was so cheap. While HTTP has subsequently gained the ability to reuse the TCP connection for multiple transfers, the conceptual model is still of independent requests rather than a session.
* When FTP is transferring over the data connection, the control connection is idle. If the transfer takes too long, the firewall or NAT may decide that the control connection is dead and stop tracking it, effectively breaking the connection and confusing the download.
* The single HTTP connection is only idle between requests and it is normal and expected for such connections to be dropped after a time-out.

**Web browser support**

* Most common web browsers can retrieve files hosted on FTP servers, although they may not support protocol extensions such as FTPS.
* When an FTP—rather than an HTTP—URL is supplied, the accessible contents on the remote server are presented in a manner that is similar to that used for other web content.
* A full-featured FTP client can be run within Firefox in the form of an extension called FireFTP.

**Security**

**FTP was not designed to be a secure protocol, and has many security weaknesses. In May 1999, the authors of RFC 2577 listed a vulnerability to the following problems:**

* Brute force attack
* FTP bounce attack
* Packet capture
* Port stealing (guessing the next open port and usurping a legitimate connection)
* Spoofing attack
* Username enumeration

FTP does not encrypt its traffic; all transmissions are in clear text, and usernames, passwords, commands and data can be read by anyone able to perform packet capture (sniffing) on the network. This problem is common to many of the Internet Protocol specifications (such as SMTP, Telnet, POP and IMAP) that were designed prior to the creation of encryption mechanisms such as TLS or SSL.

**Common solutions to this problem include:**

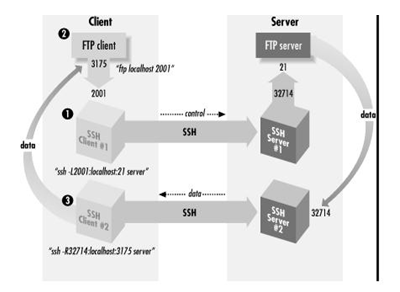
* Using the secure versions of the insecure protocols, e.g., FTPS instead of FTP and TelnetS instead of Telnet.
* Using a different, more secure protocol that can handle the job, e.g. SSH File Transfer Protocol or Secure Copy Protocol.
* Using a secure tunnel such as Secure Shell (SSH) or virtual private network (VPN).

**FTPS**

* Explicit FTPS is an extension to the FTP standard that allows clients to request FTP sessions to be encrypted. This is done by sending the "AUTH TLS" command.
* The server has the option of allowing or denying connections that do not request TLS.
* This protocol extension is defined in RFC 4217. Implicit FTPS is an outdated standard for FTP that required the use of a SSL or TLS connection.
* It was specified to use different ports than plain FTP.

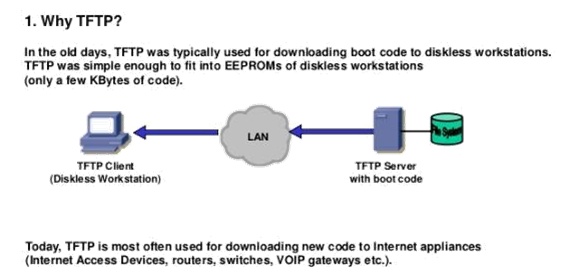
**SSH File Transfer Protocol**

* The SSH file transfer protocol (chronologically the second of the two protocols abbreviated SFTP) transfers files and has a similar command set for users, but uses the Secure Shell protocol (SSH) to transfer files.
* Unlike FTP, it encrypts both commands and data, preventing passwords and sensitive information from being transmitted openly over the network.
* It cannot interoperate with FTP software.



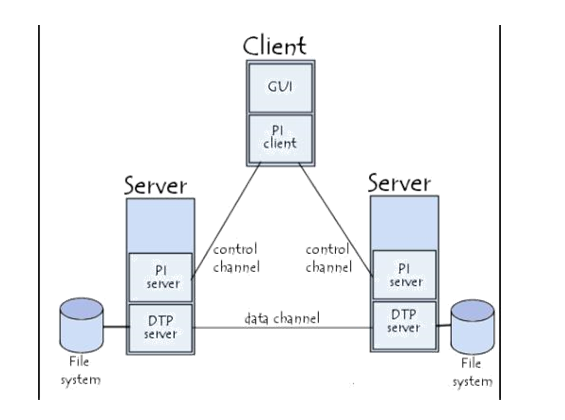
**Trivial File Transfer Protocol**

* Trivial File Transfer Protocol (TFTP) is a simple, lock-step FTP that allows a client to get a file from or put a file onto a remote host.
* One of its primary uses is in the early stages of booting from a local area network, because TFTP is very simple to implement.
* TFTP lacks security and most of the advanced features offered by more robust file transfer protocols such as File Transfer Protocol.
* TFTP was first standardized in 1981 and the current specification for the protocol can be found in RFC 1350.



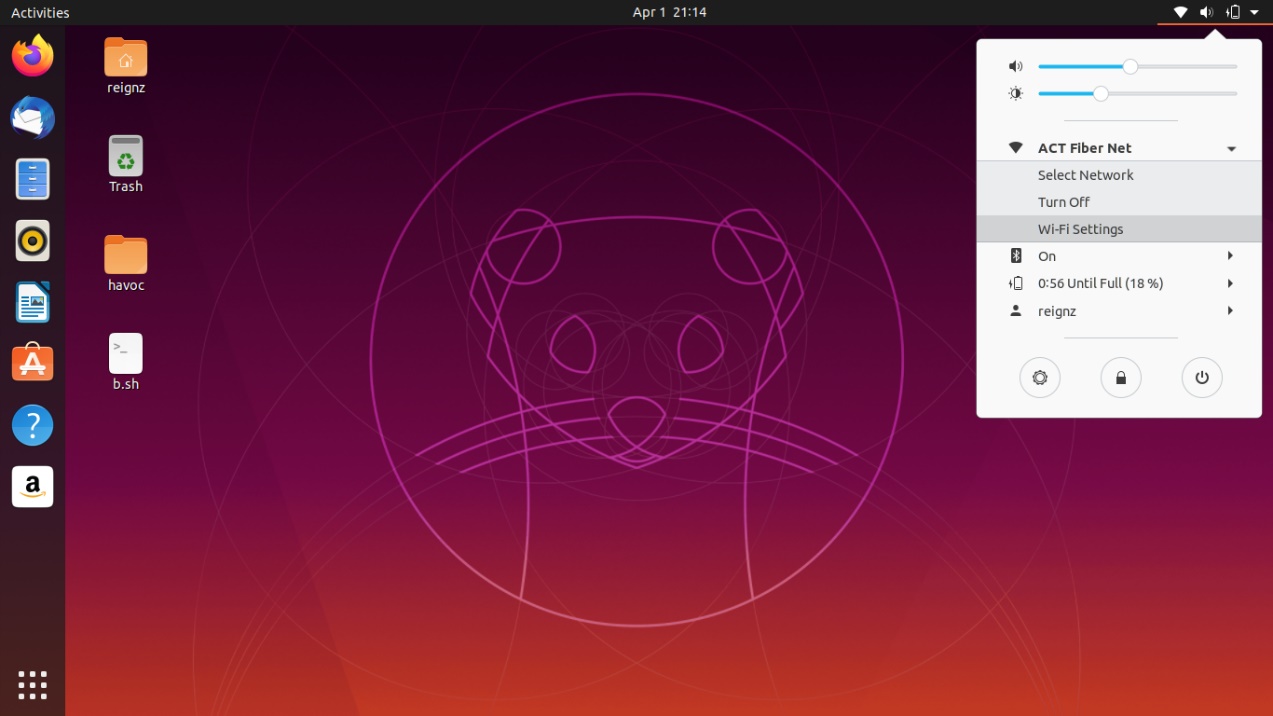
**Simple File Transfer Protocol**

* Simple File Transfer Protocol (the first protocol abbreviated SFTP), as defined by RFC 913, was proposed as an (unsecured) file transfer protocol with a level of complexity intermediate between TFTP and FTP.
* It was never widely accepted on the Internet, and is now assigned Historic status by the IETF.
* It runs through port 115, and often receives the initialism of SFTP. It has a command set of 11 commands and support three types of data transmission: ASCII, binary and continuous.
* For systems with a word size that is a multiple of 8 bits, the implementation of binary and continuous is the same.
* The protocol also supports login with user ID and password, hierarchical folders and file management (including rename, delete, upload, download, download with overwrite, and download with append).

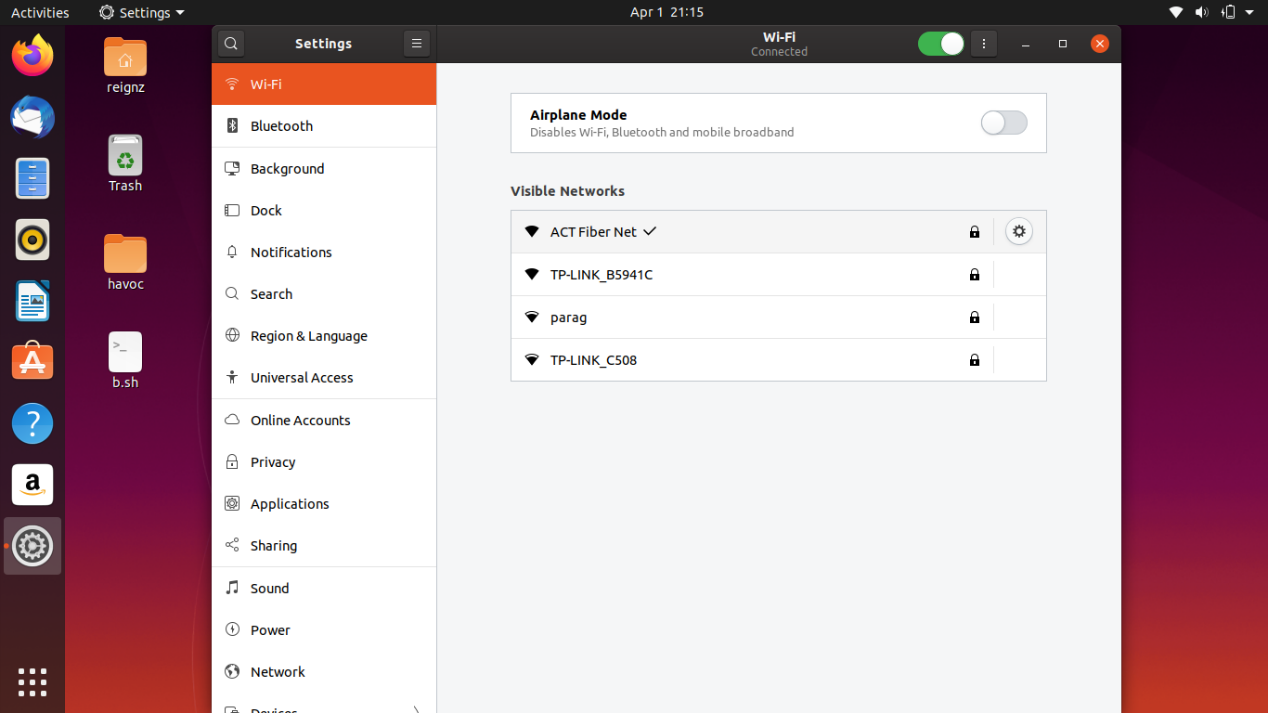


**Screenshots:**

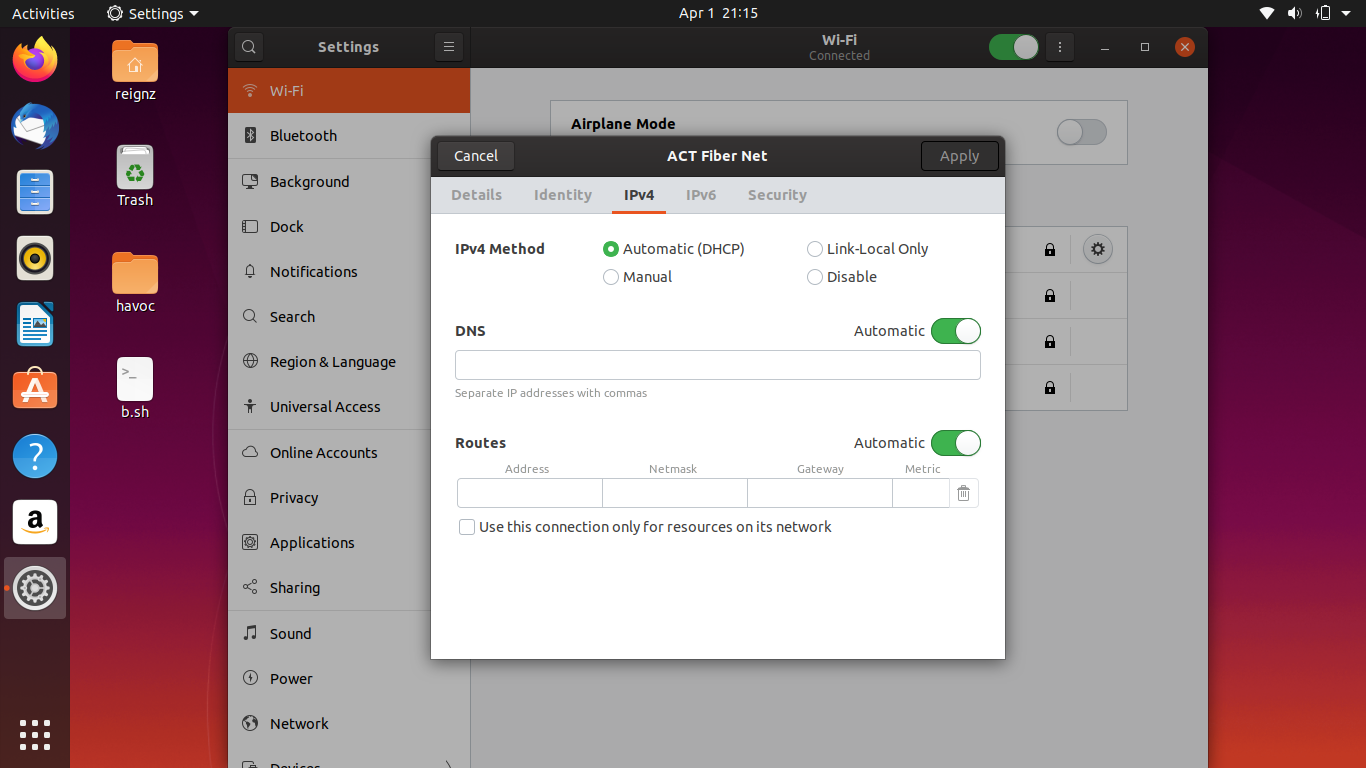
Step 1: Click on the top right network icon and select settings of the network interface you wish to configure to use a static IP address.



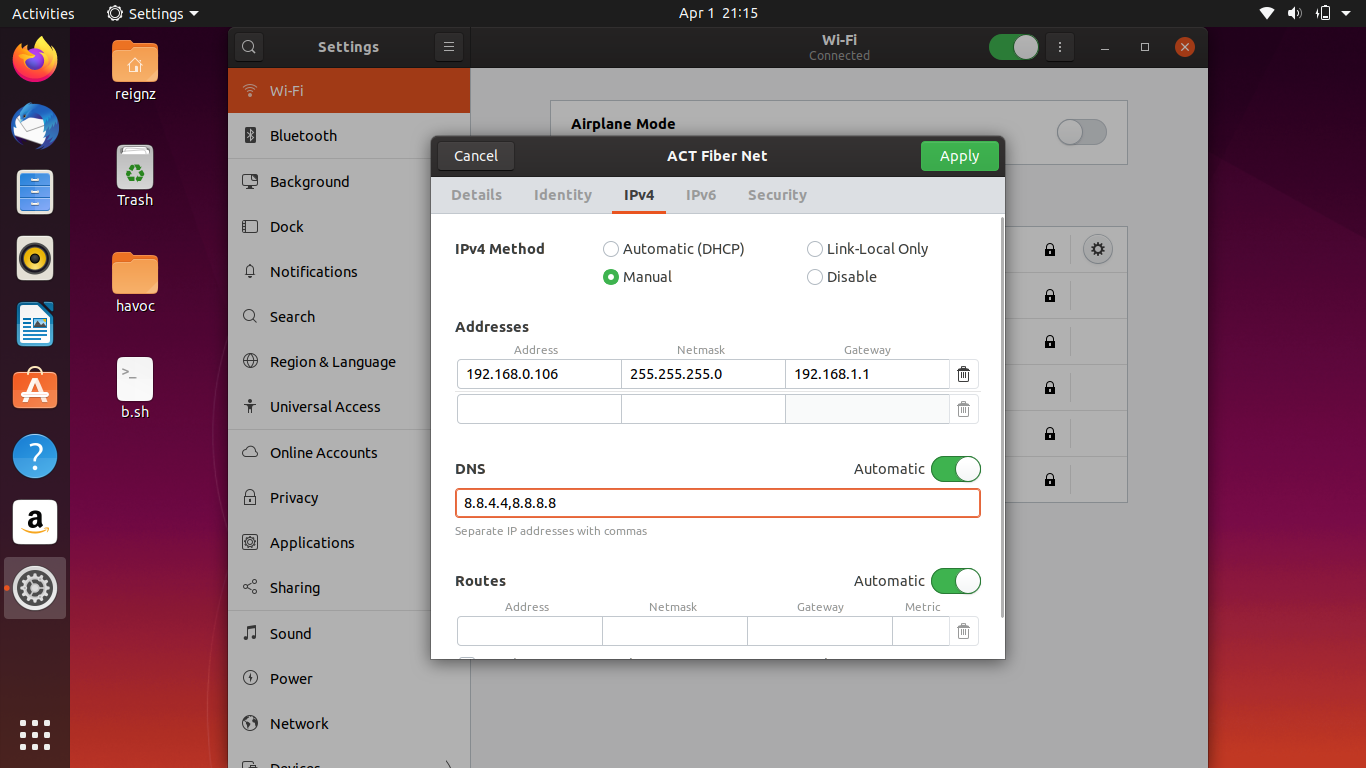
Step 2: Click on the settings icon to start configuration.



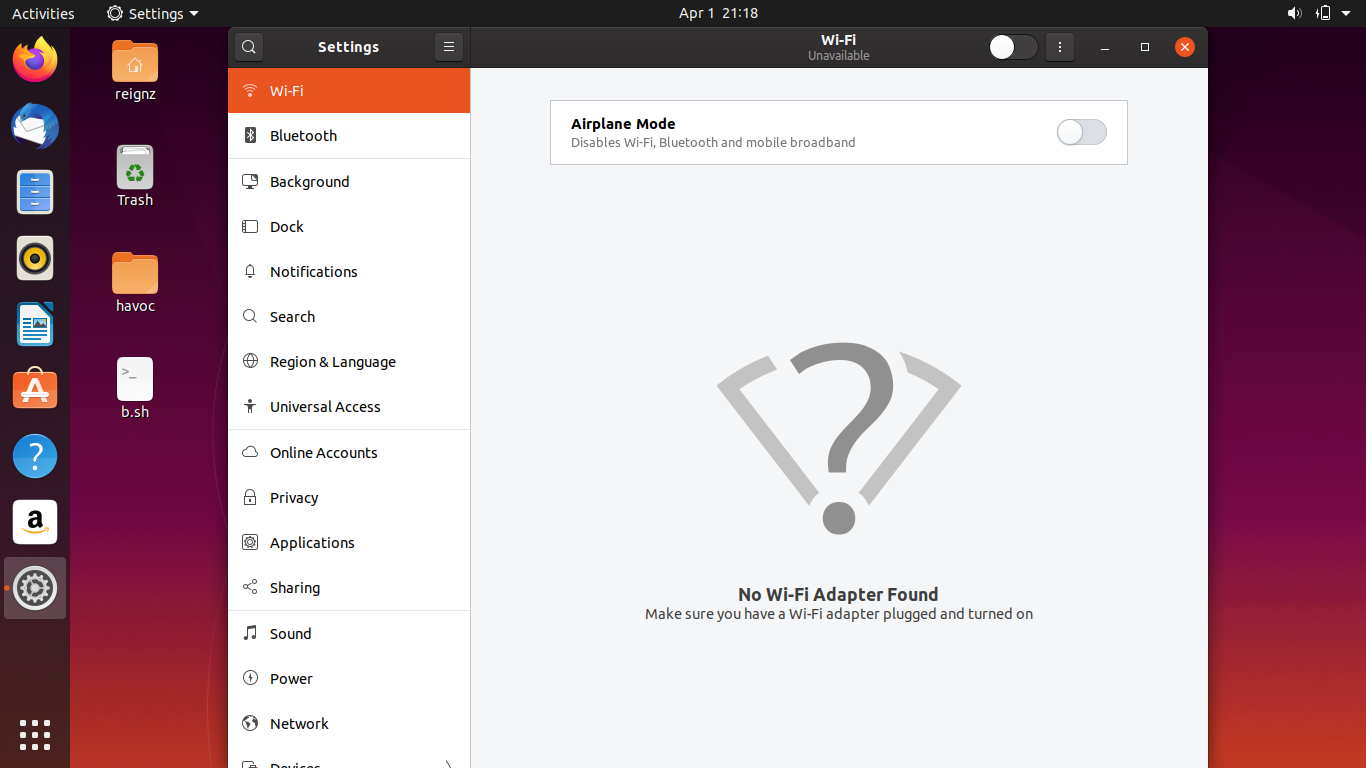
Step 3: Select IPV4 Tab



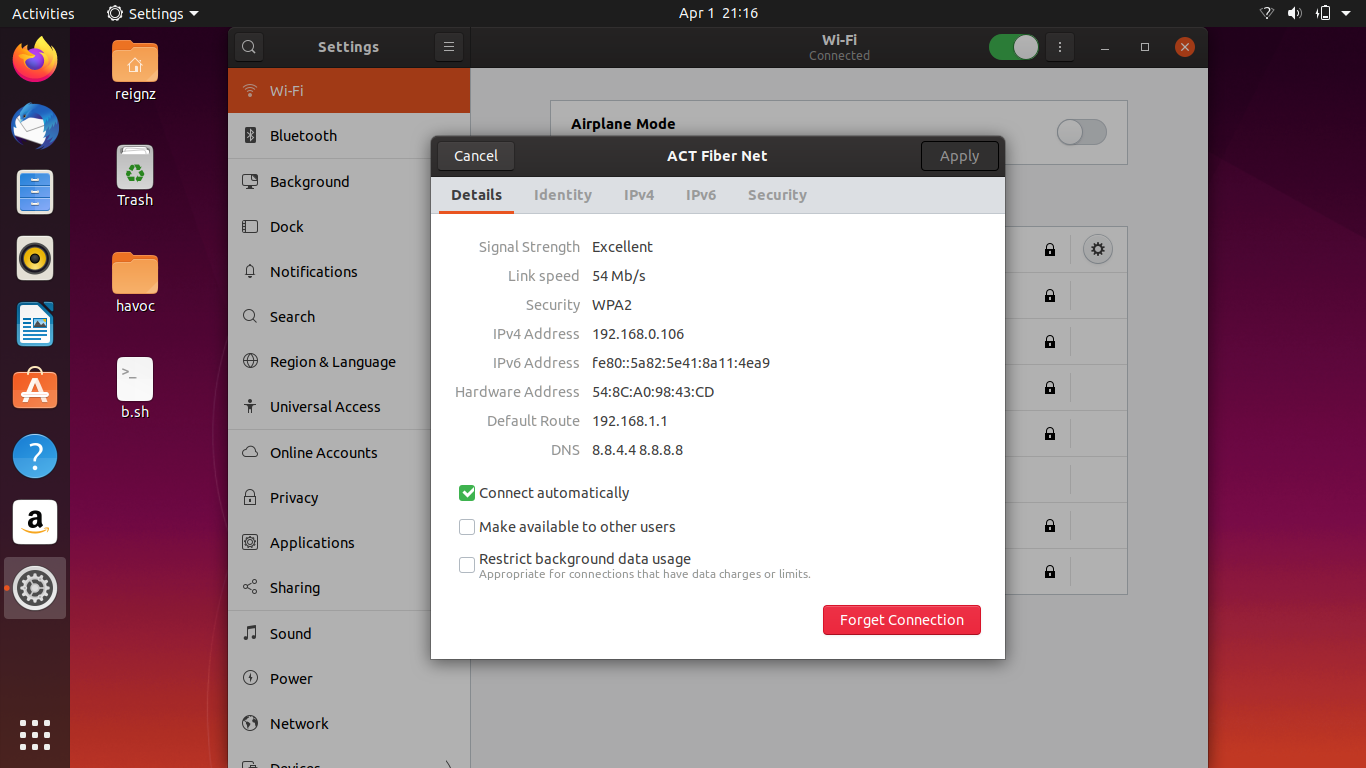
Step 4: Select manual and enter your desired static IP address, netmask, gateway and DNS settings. Once ready click apply button.



Step 5: Turn OFF and ON switch to apply your new network configuration settings.



Step 6: Click on the network settings icon once again to confirm your new IP address settings.

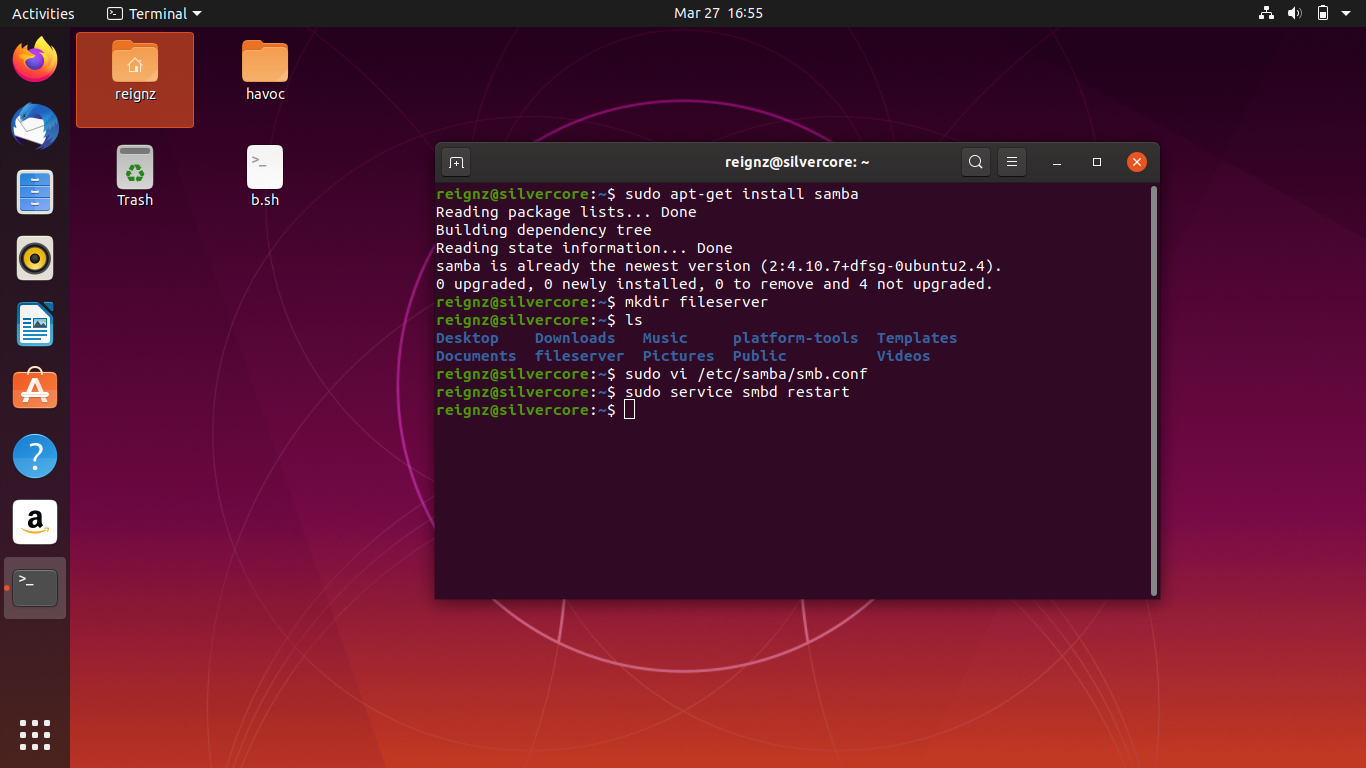


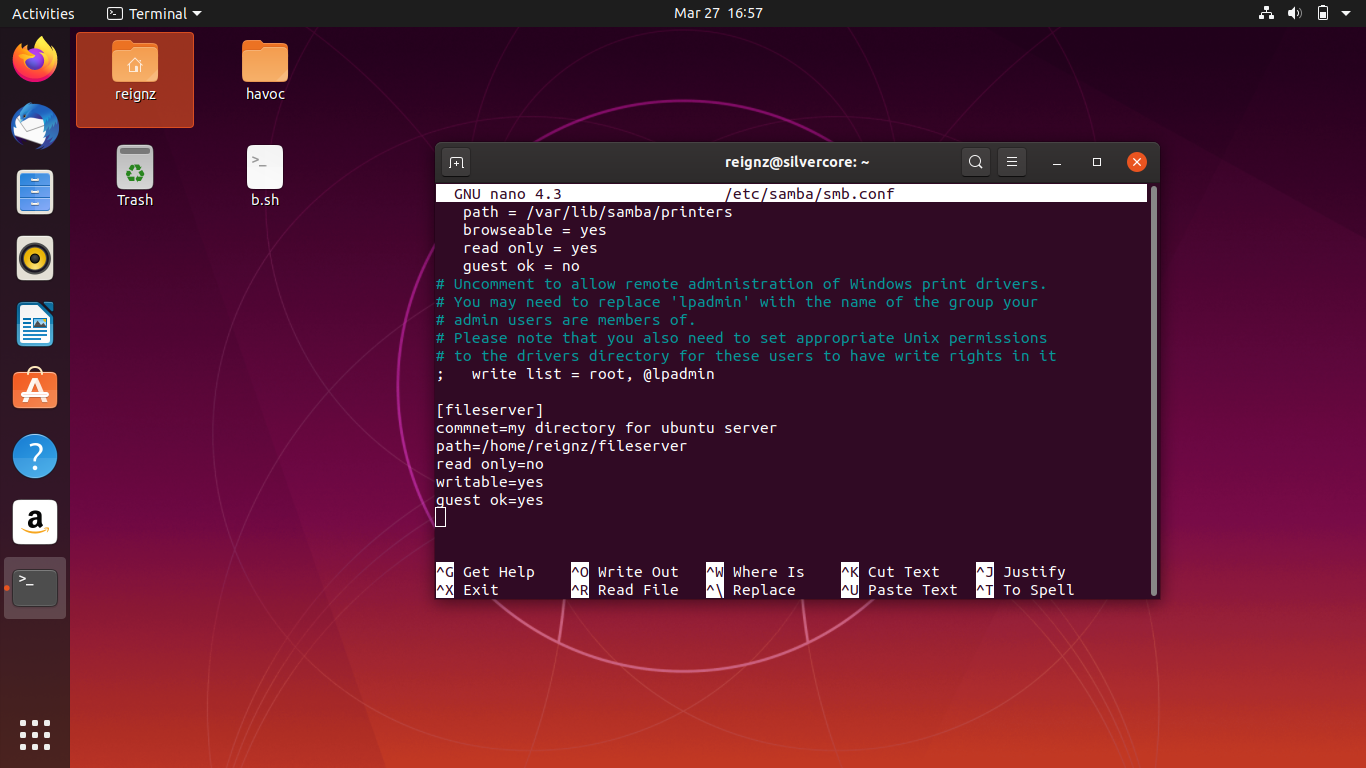
Step 7: Now install samba using the following command sudo apt-get install samba.

After installing samba now create a directory call fileserver in which we are going to share a file on the server.

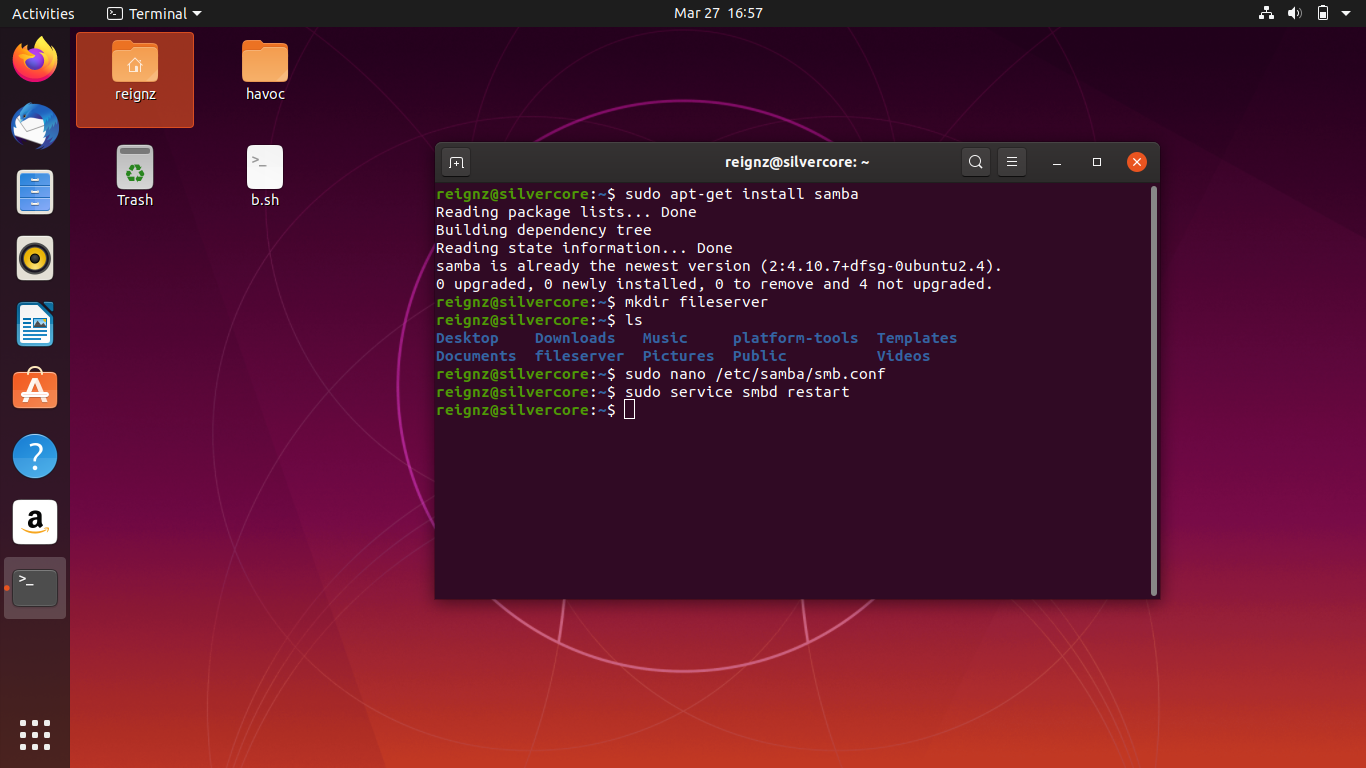
Now open samba configuration file using the following command sudo vi /etc/samba/smb.conf

Once the file is open press key i an go to last line and add the following lines show in the output of smb.conf file once it done save the file and close the editor.

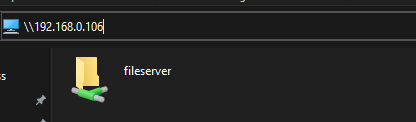


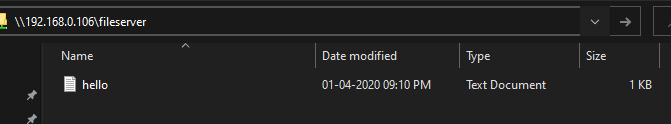


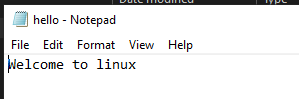
Step 8: Now restart the samba server using the following command sudo smbd restart

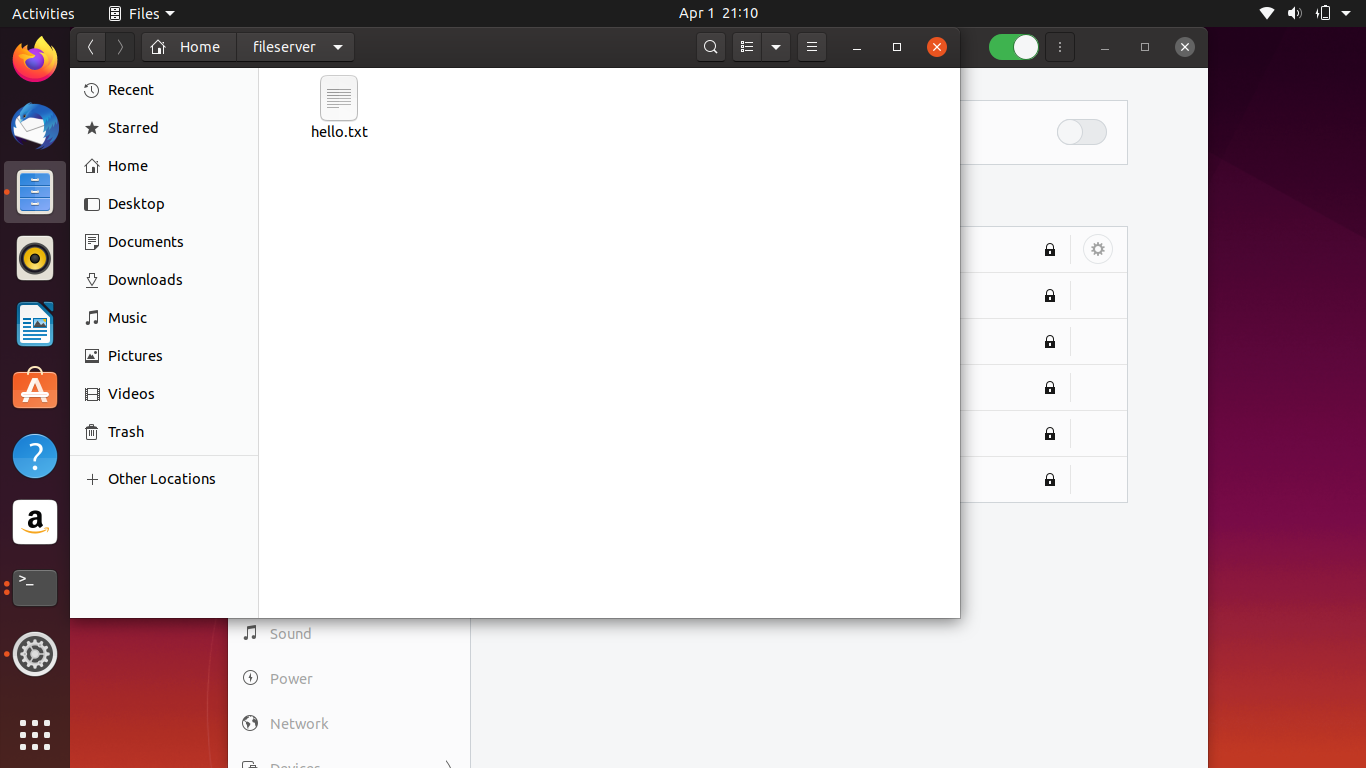


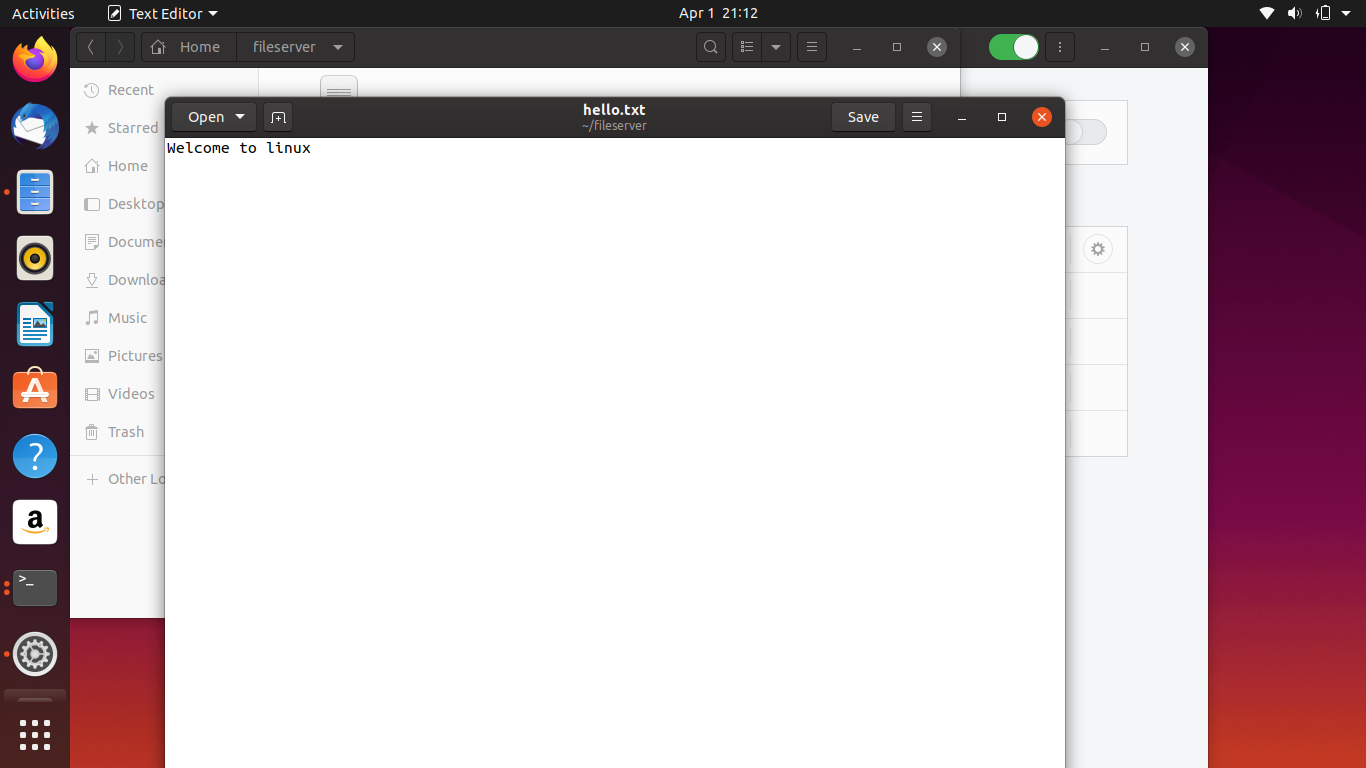
Step 9: Now we are going to access that file on windows using the staic ip we setup above to see if it working or not.











**Conclusion: -**

  We have described the operation, performance, and convenience of a transparent, adaptive mechanism for file system discovery and replacement. The adaptiveness of the method lies in the fact that a file service client no longer depends solely on a static description of where to find various file systems, but instead can invoke a resource location protocol to inspect the local area for file systems to replace the ones it already has mounted.