

AI-02022 Ubuntu Server Admin

■ Owner	Ⓒ Rigel Arcayan
■ Tags	Project
■ Created time	@August 27, 2025 4:13 PM

I'm new to Ubuntu Server but I'm familiar with Windows Server OS. I was able to deploy Ubuntu Server on my Hyper-v . My objectives are the following:

1. Be an expert Ubuntu Server Administrator
2. Be an expert in Openemr,, Apache, MariaDB, PHP
3. Be an expert in Wordpress

My requests are the following:

1. Create a top level tasks in relation to my objectives
2. List top Ubuntu applications and utilities to help me in my objectives
3. List all the sudo scripts I need for deployment, maintenance, backup, file copy, file rename, permissions, remote ssh, Remote Desktop.

Here are the top-level tasks, essential applications, and sudo scripts for your Ubuntu Server administration goals.


Top-Level Tasks

Your objectives can be broken down into three major phases: foundational server administration, application stack mastery, and specific application deployment.

- Phase 1: Master Ubuntu Server Administration
 - Learn the Linux command line and file system hierarchy.
 - Understand user and group management, and the sudo command.
 - Configure network settings, including static IP addresses.
 - Implement security best practices (firewall configuration, SSH hardening, etc.).
 - Set up and automate system maintenance tasks (updates, backups).

- Phase 2: Master the LAMP Stack (OpenEMR, WordPress)
 - Install and configure the Linux, Apache, MariaDB, and PHP stack.
 - Secure your web server with SSL/TLS certificates (e.g., Let's Encrypt).
 - Learn to manage databases with MariaDB (user creation, database management, backups).
 - Master PHP configuration and its role in web applications.
 - Understand how Apache handles virtual hosts to serve multiple websites.
- Phase 3: Deploy and Manage Applications
 - Deploy OpenEMR by following its specific installation guide, which often involves setting up the database and configuring the web server.
 - Deploy WordPress by creating a new database, user, and configuring the web server to serve the site.
 - Learn to troubleshoot common issues for both applications (e.g., permissions errors, database connection problems).
 - Implement regular, automated backup strategies for both the application files and their databases.

Top Ubuntu Applications and Utilities

Becoming an expert involves a lot of command-line work. These are the tools you'll use constantly.
- System and Performance Monitoring:
 - htop: An interactive process viewer that gives you a detailed, real-time look at your system's performance. 
 - UFW (Uncomplicated Firewall): A user-friendly front-end for iptables that makes firewall management straightforward.
 - Cockpit: A web-based graphical interface for server administration, great for monitoring and managing a server remotely.
- Text Editors:
 - nano: A simple, easy-to-use text editor for quick file edits. Perfect for beginners.

- vim or emacs: More advanced and powerful editors that are essential for long-term Linux administration, though they have a steep learning curve.
- Package and File Management:
 - apt: The primary command-line tool for installing, updating, and managing software packages on Ubuntu.
 - tar and gzip: Essential utilities for creating and extracting compressed archive files, crucial for backups.
 - rsync: A powerful utility for fast, incremental file transfers and synchronization, perfect for backups and file copies.
 - netstat or ss: Used to display active network connections, routing tables, and a wide range of network interface statistics.
- Database Management:
 - mysql-client or mariadb-client: Command-line clients for interacting with your MariaDB databases.
 - phpMyAdmin: A web-based GUI tool for managing MariaDB databases, useful if you prefer a graphical interface over the command line.

Essential sudo Scripts and Commands

Here are some essential command-line snippets and scripts you'll use frequently, all requiring sudo for elevated privileges.

Deployment and Maintenance

- Update and Upgrade System:

```
#!/bin/bash
```

```
sudo apt update -y
```

```
sudo apt upgrade -y
```

```
sudo apt autoremove -y
```

This script updates the list of available packages, upgrades all installed packages to their latest versions, and removes any unused dependencies.

- Restart a Service:

```
sudo systemctl restart apache2
```

```
sudo systemctl restart mariadb
```

Use this to restart services after making configuration changes.

Backup and File Operations

- Backup a Directory (e.g., your WordPress site):

```
#!/bin/bash
```

```
sudo tar -czvf /path/to/backup/wp-site-$(date +%Y-%m-%d).tar.gz  
/var/www/html/wordpress
```

This script creates a compressed archive (.tar.gz) of your WordPress directory, naming the backup with the current date.

- Backup a MariaDB Database:

```
#!/bin/bash
```

```
sudo mysqldump -u root -p[your_root_password] wordpress_db >  
/path/to/backup/wordpress_db-$(date +%Y-%m-%d).sql
```

This command uses mysqldump to create a backup of your wordpress_db database. Important: It's best practice to avoid putting the password directly in a script. For automation, you should use a password file.

- Copy a File/Directory:

```
sudo cp -r /source/path /destination/path
```

The -r flag is for recursive copying, which is necessary for directories.

- Rename a File/Directory:

```
sudo mv /path/to/old_name /path/to/new_name
```

Permissions

- Change File Ownership:

```
sudo chown -R www-data:www-data /var/www/html/wordpress
```

This is one of the most common commands for web servers. It recursively (-R) changes the owner and group of the WordPress files to www-data, the user that the Apache web server runs as.

- Change File Permissions:

```
sudo chmod 755 /path/to/directory
```

```
sudo chmod 644 /path/to/file
```

This is another critical command. 755 (rwx r-x r-x) for directories and 644 (rw-r-- r--) for files are common and secure permissions for web server content.

Remote Access

- Remote SSH Access:
 - First, install the OpenSSH server: `sudo apt install openssh-server`.
 - Then, you can connect from another machine using: `ssh your_username@server_ip_address`.
 - To harden security, you should configure SSH to use key-based authentication and disable password logins.
- Remote Desktop Access:
 - Install a desktop environment and a VNC/RDP server, such as `xrdp`.
 - `sudo apt install ubuntu-desktop xrdp`
 - Once installed, you can connect using a Windows Remote Desktop client to `server_ip_address`.

There were more xrdp commands I saw such as creating users, folders, permissions

That's a great observation. While the initial commands I provided get you started with `xrdp`, they are just the first step. You're correct that a complete remote desktop setup, especially in a multi-user environment, requires additional commands for user management, home directory setup, and permissions. Here are the more detailed `sudo` scripts and commands you would use to fully configure a multi-user `xrdp` environment on Ubuntu, building upon the basic installation.

1. User and Group Management for `xrdp`

When you have multiple users connecting via RDP, it's essential to create separate accounts for them.

- Create a New User for RDP Access:

```
sudo adduser new_rdp_user
```

This command creates a new user account, prompts you to set a password, and creates a home directory for them.

- Add a User to the `sudo` Group (Optional, but useful for admin users):

```
sudo usermod -aG sudo new_rdp_user
```

This command adds the `new_rdp_user` to the `sudo` group, giving them the ability to run commands with elevated privileges. The `-aG` flag means "append to group."

1. Configure `xrdp` to Use the Correct Desktop Environment

By default, `xrdp` might try to use a different desktop environment than the one you installed (e.g., GNOME). You often need to tell it explicitly which one to use.

- Create a `.xsession` file for a user:

This script creates a file in a user's home directory that tells `xrdp` what desktop environment to load. This is a crucial step for a clean RDP session.

```
#!/bin/bash
```

This script configures the `.xsession` file for the specified user

```
if [ -z "$1" ]; then
echo "Usage: $0 <username>"
exit 1
fi

USERNAME=$1
HOME_DIR="/home/$USERNAME"

if [ ! -d "$HOME_DIR" ]; then
echo "Error: Home directory for user '$USERNAME' does not exist."
exit 1
fi
```

Check if a `.xsession` file already exists and back it up

```
if [ -f "$HOME_DIR/.xsession" ]; then
sudo mv "$HOME_DIR/.xsession" "$HOME_DIR/.xsession.bak"
fi
```

Write the new .xsession file content

```
sudo sh -c "echo 'gnome-session' > $HOME_DIR/.xsession"
```

Ensure the user owns the new .xsession file

```
sudo chown "$USERNAME:$USERNAME" "$HOME_DIR/.xsession"
```

```
echo "Created .xsession file for user '$USERNAME'."
```

How to use: `sudo ./configure_xrdp_desktop.sh new_rdp_user`

3. Permissions and File System Access

Proper permissions are vital for security and functionality, especially when users need to access specific files or folders.

- Grant Read/Write Access to a Shared Folder:
Let's say you have a folder at `/var/www/shared_projects` that multiple RDP users need to access.

Create a new group for shared access

```
sudo addgroup shared_rdp_users
```

Add the users to the new group

```
sudo usermod -aG shared_rdp_users new_rdp_user
```

```
sudo usermod -aG shared_rdp_users another_user
```

Change the group ownership of the directory

```
sudo chown -R :shared_rdp_users /var/www/shared_projects
```

Set permissions so the group has read/write/execute access

```
sudo chmod -R 775 /var/www/shared_projects
```

This is a more robust approach than simply giving everyone 777 permissions, which is a major security risk.

4. Firewall Configuration for xrdp

Just as with SSH, you must open the correct port in your firewall for RDP connections.

- Allow RDP (xrdp) Connections:

```
sudo ufw allow 3389/tcp
```

```
sudo ufw reload
```

The xrdp service listens on port 3389 by default. This command allows inbound connections to that port through your firewall.

1. xrdp Configuration File Editing

For more advanced settings, you might need to edit the xrdp.ini configuration file.

- Edit the xrdp.ini file:

```
sudo nano /etc/xrdp/xrdp.ini
```

In this file, you can adjust settings like the listening port, logging level, and default session type. This is where you would fine-tune the xrdp behavior. For example, if you wanted to change the default port, you'd change the port=3389 line.

These commands provide a more comprehensive set of tools for deploying and managing a production-ready xrdp environment with multiple users, covering not just the installation but also user, file, and security management.

I would like to download and install Nautilus, Microsoft Edge, VS Code, Git, can you show me how to do this. It seems simple in Windows environment but in Linux, it seems everything is done in command line which I think a lot simpler once I will have a list of commands to learn and practice.

That's an excellent way to think about it. The command line in Linux is incredibly powerful and efficient once you get the hang of it. You're right that it simplifies a lot of tasks once you know the commands.

Here are the commands to download and install Nautilus, Microsoft Edge, VS Code, and Git on Ubuntu. Since you're using Ubuntu Server, you'll need a desktop environment to use graphical applications like Nautilus, Edge, and VS Code. The

commands below assume you've already installed a desktop environment like GNOME or XFCE on your server.

1. Install a Desktop Environment

If you haven't already, you need to install a desktop environment to run graphical applications. The most common one for Ubuntu is GNOME.

- Install the GNOME desktop:

```
sudo apt update
sudo apt install ubuntu-desktop
```

This command downloads and installs the full desktop environment, which can take some time. Once complete, you can reboot your server to start using the graphical interface.

1. Install Nautilus (GNOME Files)

Nautilus is the default file manager for the GNOME desktop. It's usually installed as part of the desktop environment.

- Install Nautilus:

```
sudo apt install nautilus
```

This command ensures that Nautilus is installed. If you've already installed `ubuntu-desktop`, it's likely already present.

1. Install Microsoft Edge

Microsoft provides an official repository for installing Edge on Linux, which makes it easy to keep the browser updated.

- Add the Microsoft GPG key:

```
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --
dearmor > packages.microsoft.gpg
sudo install -o root -g root -m 644 packages.microsoft.gpg
/etc/apt/trusted.gpg.d/
```

This step adds a key that verifies the authenticity of packages from Microsoft's repository.

- Add the Microsoft Edge repository:

```
sudo sh -c 'echo "deb [arch=amd64]
https://packages.microsoft.com/repos/edge stable main" >
```

```
/etc/apt/sources.list.d/microsoft-edge.list'  
sudo rm packages.microsoft.gpg
```

This adds the repository to your system's sources list, so the apt package manager knows where to find the Edge browser.

- Install Microsoft Edge:

```
sudo apt update  
sudo apt install microsoft-edge-stable
```

First, you update the package list to include the new repository, then you install the stable version of Microsoft Edge.

1. Install Visual Studio Code (VS Code)

Like Edge, VS Code is a Microsoft product with a dedicated repository for Linux.

- Add the VS Code GPG key:

```
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --  
dearmor > packages.microsoft.gpg  
sudo install -o root -g root -m 644 packages.microsoft.gpg  
/etc/apt/trusted.gpg.d/
```

This is the same key you used for Edge, but it's good practice to re-add it or verify it's there.

- Add the VS Code repository:

```
sudo sh -c 'echo "deb [arch=amd64,arm64,armhf]  
https://packages.microsoft.com/repos/vscode stable main" >  
/etc/apt/sources.list.d/vscode.list'  
sudo rm packages.microsoft.gpg
```

This command adds the VS Code repository to your sources.

- Install VS Code:

```
sudo apt update  
sudo apt install code
```

You update the package list again, then use the apt install code command to install VS Code.

1. Install Git

Git is a fundamental tool for developers and is almost always available directly from the standard Ubuntu repositories, making the installation very simple.

- Install Git:

```
sudo apt update
```

```
sudo apt install git
```

This command updates the local package index and then installs the latest version of Git available in the official Ubuntu repositories.

- Verify the installation:

```
git --version
```

You can run this command to check that Git was installed correctly and to see its version number.

Tell more of other sudo commands I need to jumpstart my learning

Jumping into Ubuntu administration requires a good handle on fundamental sudo commands. Here are more commands beyond the initial set, categorized to help you jumpstart your learning.

1. File System and Directories

These commands are crucial for navigating and managing files and directories.

- `sudo mkdir /var/www/myproject`: Creates a new directory. This is often needed for setting up a web project or an application's folder.
- `sudo rm -rf /var/www/oldproject`: Removes a directory and all its contents recursively (-r) and forcefully (-f). Use this with extreme caution, as it can delete critical system files without a warning.
- `sudo find / -name "apache2.conf"`: Searches the entire file system (/) for a file or directory with a specific name. This is invaluable for locating configuration files.

1. Service Management

Managing system services is a core part of server administration.

- `sudo systemctl status apache2`: Checks the current status of a service. You'll use this to see if a service is running, stopped, or has an error.

- `sudo systemctl stop apache2`: Stops a running service.
- `sudo systemctl start apache2`: Starts a stopped service.
- `sudo systemctl enable apache2`: Ensures a service starts automatically after a reboot.
- `sudo systemctl disable apache2`: Prevents a service from starting automatically after a reboot.

1. User and Group Management 👤

These commands are essential for creating and managing user accounts and their permissions.

- `sudo adduser newuser`: Creates a new user account. This command is more user-friendly than `useradd` as it prompts you to set a password and creates a home directory.
- `sudo deluser olduser`: Deletes a user account. You can add the `--remove-home` flag to also delete their home directory.
- `sudo usermod -aG groupname username`: Adds an existing user to a new group. For example, `sudo usermod -aG sudo myadmin` adds the `myadmin` user to the `sudo` group.
- `sudo chown -R newuser:newuser /home/newuser`: Changes the ownership of a file or directory. The `-R` flag is for recursive ownership change, useful for entire home directories or project folders.

1. Networking 🌐

Managing network configurations and diagnosing connectivity issues are critical skills.

- `sudo ip a`: Displays all network interfaces and their IP addresses. This is the modern replacement for the older `ifconfig` command.
- `sudo ss -tulpn`: Shows listening TCP and UDP ports and the processes that are using them. This is the modern replacement for `netstat` and is invaluable for troubleshooting port conflicts.
- `sudo ufw status`: Shows the current status of the Uncomplicated Firewall. You'll use this to see if your firewall is active and which rules are in place.

1. System Information and Utilities

These commands help you gather information about your system and manage background processes.

- `sudo df -h`: Shows disk space usage in a human-readable format.
- `sudo du -sh /var/log`: Summarizes the disk usage of a specific directory. The `-s` flag provides a summary, and `-h` makes it human-readable.
- `sudo reboot`: Reboots the system.
- `sudo shutdown now`: Shuts down the system immediately. You can also schedule a shutdown with a time, for example: `sudo shutdown +15` for a shutdown in 15 minutes.

Is there an acronym list and meaning I can use? It's always a good learning technique if I know what the command stands for such as `mkdir` to make directory, `cp` for copy, `mv` for move . But I'm still new like the background of `sudo` or `apt` or `aG`, tell more

That's an excellent approach to learning the command line. Understanding the etymology of commands makes them much easier to remember. Here is a list of common Linux command acronyms and their meanings, including the ones you mentioned and some others you'll encounter frequently.

Basic Commands

- `sudo` (SuperUser Do): This command allows a permitted user to execute a command as the superuser (root) or another user, as specified by the security policy.
- `cd` (Change Directory): Moves you from one directory to another in the file system.
- `ls` (List): Lists the files and directories in the current directory.
- `pwd` (Print Working Directory): Shows you the absolute path of your current location in the file system.
- `cp` (Copy): Creates a copy of a file or directory.
- `mv` (Move): Moves a file or directory from one location to another. It's also used to rename files.

- `rm` (Remove): Deletes files.
 - `mkdir` (Make Directory): Creates a new directory.
- Package Management
- `apt` (Advanced Package Tool): A command-line utility for handling packages in Debian-based systems like Ubuntu. It's a successor to older tools like `apt-get` and `apt-cache`.
 - `dpkg` (Debian Package): The low-level backend tool used to install, build, and remove software packages. `apt` uses `dpkg` in the background.
 - `aG` (Append Group): This is a combination of two options used with commands like `usermod`. The `-a` stands for "append", and the `-G` stands for "groups". So, `usermod -aG sudo newuser` means "append the sudo group to newuser's list of secondary groups."
- System & Service Management
- `df` (Disk Free): Reports the amount of free disk space on your file systems.
 - `du` (Disk Usage): Estimates and displays the disk usage of files in a directory.
 - `ps` (Process Status): Displays information about the currently running processes.
 - `top` (Table of Processes): Provides a dynamic, real-time view of running processes and system resource usage.
 - `htop`: An interactive and user-friendly version of `top`.
 - `ss` (Socket Statistics): A modern utility that replaces `netstat` for displaying network socket information.
 - `UFW` (Uncomplicated Firewall): A user-friendly command-line interface for managing firewall rules.
- Networking & Remote Access
- `ip` (Internet Protocol): A powerful utility for showing and configuring network interfaces and routing. It has largely replaced `ifconfig`.
 - `ssh` (Secure Shell): A cryptographic network protocol for secure data communication, remote command-line login, and other secure network services.

- `scp` (Secure Copy): A command-line tool for securely copying files between a local host and a remote host. It uses the `ssh` protocol for data transfer.
- `xrdp`: An open-source implementation of the Microsoft Remote Desktop Protocol (RDP) for Linux, allowing you to connect to a Linux desktop from a Windows machine.

Knowing these meanings will help you build a mental model of how the Linux command line operates, making it a much more intuitive and powerful tool.

In windows, there are `ipconfig -release`, `ping` etc. tell more in Linux

You're right, Windows has `ipconfig` and `ping`. Linux has similar commands, but with different names and more powerful options. The most important commands for network administration in Linux are `ip` (the modern standard) and `ping`.

`ip` Command (The `ipconfig` Equivalent)

The `ip` command is the modern replacement for `ifconfig` and is a comprehensive tool for configuring and displaying network settings.

- Display IP Addresses:

```
ip a
```

This command, short for `ip address show`, shows the IP addresses, network interfaces (like `eth0` or `enp0s25`), and their current state. This is the direct equivalent of `ipconfig`.

- Bring an Interface Up/Down:

```
sudo ip link set dev enp0s25 down
```

```
sudo ip link set dev enp0s25 up
```

These commands deactivate and activate a specific network interface. This is useful for troubleshooting or making configuration changes.

- Add/Remove an IP Address:

```
sudo ip a add 192.168.1.100/24 dev enp0s25
```

```
sudo ip a del 192.168.1.100/24 dev enp0s25
```

You can manually add a static IP address to an interface or remove an existing one.

- Show Routing Table:

```
ip r
```

This command, short for `ip route`, displays the system's routing table, showing how network traffic is routed to different destinations. This is comparable to Windows' `route print`.

ping Command

The ping command works the same way as its Windows counterpart but has a few key differences and more options.

- Basic Ping:

`ping google.com`

Unlike in Windows where ping sends four packets and stops, in Linux it sends packets continuously until you manually stop it with `Ctrl+C`.

- Send a Specific Number of Pings:

`ping -c 5 google.com`

The `-c` (count) option tells ping to send a specific number of packets, making it behave like the Windows version.

Other Useful Network Commands

- `ss` (Socket Statistics): This command is a modern, faster replacement for `netstat`.

`ss -tulpn`

This command is a lifesaver for troubleshooting. The options break down as follows:

- `t`: Show TCP sockets.
 - `u`: Show UDP sockets.
 - `l`: Show listening sockets.
 - `p`: Show the process using the socket.
 - `n`: Show numerical addresses instead of hostnames.
- `traceroute` (The `tracert` Equivalent):
`traceroute google.com`

This command traces the path a packet takes to a destination, listing each hop (router) along the way. In Windows, the same command is called `tracert`.

- dig (Domain Information Groper): This is a powerful command for performing DNS lookups.

`dig google.com`

It provides much more detailed information than nslookup (the older Windows equivalent), including DNS records like A, MX, and CNAME.

- hostname:
`hostname`

Displays the hostname of the server. This is the Linux equivalent of the `hostname` command in Windows.

Explain Bash, is it same as the basic shell

No, Bash isn't the same as the basic shell, but it's a type of shell. Think of a shell as a program that takes commands from your keyboard and gives them to the operating system to execute. It's the command-line interface.

Bash, which stands for Bourne-again shell, is a specific and very popular type of shell. It's an improved version of the original "sh" (Bourne shell) that was used in early Unix systems.

Key Differences

- **Interactive Features:** Bash includes features that make it easier to use, such as command-line completion (pressing the Tab key to automatically complete a command or file name) and command history (using the up and down arrow keys to recall previous commands). The basic Bourne shell didn't have these.
- **Scripting Language:** Bash is also a scripting language. You can write complex scripts to automate tasks, set variables, and use control structures like loops (for, while) and conditionals (if/else). While basic shells have some scripting capabilities, Bash offers a richer and more powerful set of features.
- **Portability:** The original shells were designed for specific systems. Bash was developed to be compatible with a wide range of operating systems, including Linux, macOS, and even Windows via the Windows Subsystem for Linux (WSL).

In modern Linux distributions like Ubuntu, Bash is the default shell for most users. When you open a terminal and type commands, you are almost always using Bash. The basic shell (sh) still exists, but it's often a symbolic link to

Bash, or a stripped-down version, used for system scripts that need to be highly portable and don't require the advanced features of Bash.

I once used a Windows terminal to ssh to my Ubuntu server, tell more

It's great that you've already used SSH. That's a fundamental skill for server administration. The ability to connect from a Windows terminal to your Ubuntu server is thanks to the OpenSSH client, which is now built into modern versions of Windows 10 and 11.

How It Works

- **SSH Protocol:** The connection is made using the Secure Shell (SSH) protocol. This protocol provides a secure, encrypted tunnel between your local machine (the client) and the remote server. All the data you send, including your password and commands, is encrypted, making it safe to use over public networks.
- **SSH Client:** Your Windows terminal (like Command Prompt or PowerShell) acts as the SSH client. It initiates the connection request.
- **SSH Server:** Your Ubuntu machine runs an SSH server (specifically, the `sshd` daemon). It listens for connection requests on a specific port (usually port 22) and authenticates the user before granting access to a shell session.

The Command

The command you used is the most basic form of an SSH connection:

```
ssh username@ip_address
```

- `ssh`: The command that invokes the SSH client.
- `username`: The account you want to log in as on the Ubuntu server.
- `@`: A separator that connects the username and the server's address.
- `ip_address`: The IP address or hostname of your Ubuntu server.

More on SSH

You can do a lot more with SSH than just log in. Here are some of the most common and powerful uses:

- **Remote Command Execution:** You can run a single command on the server without starting a full interactive shell session. This is useful for scripting and automation.

```
ssh username@ip_address "ls -l /var/www/html"
```

- Secure File Transfer: You can copy files and directories between your Windows machine and the Ubuntu server using SCP (Secure Copy Protocol) or rsync. Both use SSH for the secure data transfer.

Copy a file from your Windows machine to the Ubuntu server

```
scp C:\path\to\local\file.txt username@ip_address:/path/to/remote/
```

Copy a file from the Ubuntu server to your Windows machine

```
scp username@ip_address:/path/to/remote/file.txt C:\path\to\local\
```

- Port Forwarding (Tunnelling): This allows you to securely access a network service on the remote server through your local machine. For example, you can create a secure tunnel to access a web server on a port that isn't publicly accessible.

```
ssh -L 8080:localhost:80 username@ip_address
```

This command forwards local port 8080 to remote port 80, allowing you to access the web server by navigating to <http://localhost:8080> in your browser.

- SSH Keys for Password-less Login: Instead of using a password every time, you can set up SSH keys for more secure and convenient authentication. You generate a pair of keys (a public and a private key). The private key stays on your Windows machine and is never shared, while the public key is uploaded to your Ubuntu server. The server uses the public key to verify your identity without needing a password.

Wow too much information but it's really helpful. How to transfer files from my Windows box to my Ubuntu server using a gui? Possible?

Yes, it is very possible and quite common to transfer files using a graphical user interface (GUI). The most common method involves a client on your Windows machine that uses the SFTP (Secure File Transfer Protocol). SFTP is built on top of the SSH protocol, so it's secure and uses the same port (22) that you use for your

SSH command-line connections.

You don't need to install any additional software on your Ubuntu server, as the SSH server you've already installed handles SFTP. You just need to install an SFTP client on your Windows machine.

Recommended GUI Tools

Two of the most popular and reliable free tools for this purpose are WinSCP and FileZilla. Both provide a user-friendly, drag-and-drop interface similar to a file manager.

- WinSCP: This is a powerful, Windows-only SFTP client that's perfect for this task. It has a two-panel interface, with your local Windows file system on one side and the remote Ubuntu file system on the other.
- FileZilla: A very popular, open-source FTP client that also supports SFTP. It's available for Windows, macOS, and Linux. It has a similar dual-pane interface to WinSCP.

How to Transfer Files with WinSCP

- Download and Install WinSCP: Go to the official WinSCP website and download the installer.
- Launch the Application: Open WinSCP, and a "Login" window will appear.
- Enter Connection Details:
 - File Protocol: Select SFTP.
 - Host name: Enter the IP address of your Ubuntu server.
 - Port number: This should be 22 by default.
 - User name: Enter your Ubuntu server username.
 - Password: Enter your password.
- Connect: Click the Login button. WinSCP might show a security alert about the host key; this is normal for a new connection. Click Yes to continue.
- Drag and Drop: Once connected, you will see your Windows folders on the left pane and your Ubuntu server's file system on the right. You can now simply drag and drop files and folders between the two panels to copy them.

List the FileZilla file transfer instructions

Sure, here are the instructions for transferring files using the FileZilla GUI.
FileZilla File Transfer Instructions

- Download and Install FileZilla: Download the FileZilla Client from the official FileZilla website and install it on your Windows machine.
- Launch FileZilla: Open the application once the installation is complete.
- Enter Server Details: At the top of the FileZilla window, you'll see a quick connect bar. Enter your server's connection details in the following fields:
 - Host: Enter the IP address of your Ubuntu server. You can also use a hostname if you have one.
 - Username: Enter your Ubuntu server username.
 - Password: Enter your user password.
 - Port: Enter 22. This is the default port for SFTP.
- Connect: Click the Quickconnect button. FileZilla will attempt to establish a secure connection to your server using SFTP (Secure File Transfer Protocol), which is built into the SSH server. If you're connecting for the first time, you may see a warning about an unknown host key. This is normal. Check the box to "Always trust this host" and click OK.
- Transfer Files: Once connected, the FileZilla interface is divided into four main sections:
 - Local site (Left pane): Your computer's file system.
 - Remote site (Right pane): Your Ubuntu server's file system.
 - Transfer Queue: The bottom pane shows the status of ongoing file transfers.
 - Log Window: The top pane displays command and status messages from the server.To transfer files, simply drag and drop them from the Local site pane to the desired directory in the Remote site pane. The transfer progress will be visible in the queue at the bottom. You can also right-click on a file and select "Upload" or "Download."

Bash Cheat Sheet

A cheat sheet for bash commands.

Command History

```
``bash
```

```
!!          # Run the last command
```

```
touch foo.sh
```

```
chmod +x !$ # !$ is the last argument of the last command i.e. foo.sh
```

```
...
```

Navigating Directories

```
``bash
```

```
pwd          # Print current directory path
```

```
ls           # List directories
```

```
ls -a|--all  # List directories including hidden
```

```
ls -l        # List directories in long form
```

```
ls -l -h|--human-readable # List directories in long form with human readable sizes
```

```
ls -t        # List directories by modification time, newest first
```

```
stat foo.txt # List size, created and modified timestamps for a file
```

```
stat foo     # List size, created and modified timestamps for a directory
```

```
tree         # List directory and file tree
```

```
tree -a      # List directory and file tree including hidden
```

```
tree -d      # List directory tree
```

```
cd foo       # Go to foo sub-directory
```

```
cd           # Go to home directory
```

```
cd ~         # Go to home directory
```

```
cd -         # Go to last directory
```

```
pushd foo    # Go to foo sub-directory and add previous directory to stack
```

```
ack
```

```
popd         # Go back to directory in stack saved by `pushd`
```

...

Creating Directories

```
``bash
mkdir foo                # Create a directory
mkdir foo bar            # Create multiple directories
mkdir -p|--parents foo/bar # Create nested directory
mkdir -p|--parents {foo,bar}/baz # Create multiple nested directories

mktemp -d|--directory    # Create a temporary directory
...
```

Moving Directories

```
``bash
cp -R|--recursive foo bar # Copy directory
mv foo bar                 # Move directory

rsync -z|--compress -v|--verbose /foo /bar # Copy directory, overwrite
destination
rsync -a|--archive -z|--compress -v|--verbose /foo /bar # Copy directory, without
overwriting destination
rsync -avz /foo username@hostname:/bar # Copy local directory to
remote directory
rsync -avz username@hostname:/foo /bar # Copy remote directory
to local directory
...
```

Deleting Directories

```
``bash
rmdir foo                # Delete empty directory
rm -r|--recursive foo    # Delete directory including contents
rm -r|--recursive -f|--force foo # Delete directory including contents, ignore
non-existent files and never prompt
```

```

...

## Creating Files

``bash
touch foo.txt      # Create file or update existing files modified timestamp
touch foo.txt bar.txt # Create multiple files
touch {foo,bar}.txt # Create multiple files
touch test{1..3}   # Create test1, test2 and test3 files
touch test{a..c}   # Create testa, testb and testc files

mktemp            # Create a temporary file
...

## Standard Output, Standard Error and Standard Input

``bash
echo "foo" > bar.txt    # Overwrite file with content
echo "foo" >> bar.txt   # Append to file with content

ls exists 1> stdout.txt # Redirect the standard output to a file
ls noexist 2> stderr.txt # Redirect the standard error output to a file
ls 2>&1 > out.txt       # Redirect standard output and error to a file
ls > /dev/null         # Discard standard output and error

read foo           # Read from standard input and write to the variable foo
...

## Moving Files

``bash
cp foo.txt bar.txt          # Copy file
mv foo.txt bar.txt         # Move file

rsync -z|--compress -v|--verbose /foo.txt /bar # Copy file quickly if not changed

```

```
rsync z|--compress -v|--verbose /foo.txt /bar.txt # Copy and rename file quickly if not changed
...
```

Deleting Files

```
``bash
rm foo.txt      # Delete file
rm -f|--force foo.txt # Delete file, ignore nonexistent files and never prompt
...
```

Reading Files

```
``bash
cat foo.txt      # Print all contents
less foo.txt     # Print some contents at a time (g - go to top of file, SHIFT+g, go to bottom of file, /foo to search for 'foo')
head foo.txt     # Print top 10 lines of file
tail foo.txt     # Print bottom 10 lines of file
open foo.txt     # Open file in the default editor
wc foo.txt      # List number of lines words and characters in the file
...
```

File Permissions

#	Permission	rwx	Binary
-	-	- -	
7	read, write and execute	rwx	111
6	read and write	rw-	110
5	read and execute	r-x	101
4	read only	r--	100
3	write and execute	-wx	011
2	write only	-w-	010
1	execute only	--x	001
0	none	---	000

For a directory, execute means you can enter a directory.

User	Group	Others	Description
-	-	-	-
6	4	4	User can read and write, everyone else can read (Default file permissions)
7	5	5	User can read, write and execute, everyone else can read and execute (Default directory permissions)

- u - User
- g - Group
- o - Others
- a - All of the above

```
```bash
ls -l /foo.sh # List file permissions
chmod +100 foo.sh # Add 1 to the user permission
chmod -100 foo.sh # Subtract 1 from the user permission
chmod u+x foo.sh # Give the user execute permission
chmod g+x foo.sh # Give the group execute permission
chmod u-x,g-x foo.sh # Take away the user and group execute permission
chmod u+x,g+x,o+x foo.sh # Give everybody execute permission
chmod a+x foo.sh # Give everybody execute permission
chmod +x foo.sh # Give everybody execute permission
```
```

Finding Files

Find binary files for a command.

```
```bash
type wget # Find the binary
which wget # Find the binary
whereis wget # Find the binary, source, and manual page files
es
```

...

`locate` uses an index and is fast.

```
``bash
updatedb # Update the index

locate foo.txt # Find a file
locate --ignore-case # Find a file and ignore case
locate f*.txt # Find a text file starting with 'f'
...

```

`find` doesn't use an index and is slow.

```
``bash
find /path -name foo.txt # Find a file
find /path -iname foo.txt # Find a file with case insensitive search
find /path -name "*.txt" # Find all text files
find /path -name foo.txt -delete # Find a file and delete it
find /path -name "*.png" -exec pngquant {} # Find all .png files and execute pngquant on it
find /path -type f -name foo.txt # Find a file
find /path -type d -name foo # Find a directory
find /path -type l -name foo.txt # Find a symbolic link
find /path -type f -mtime +30 # Find files that haven't been modified in 30 days
find /path -type f -mtime +30 -delete # Delete files that haven't been modified in 30 days
...

```

## Find in Files

```
``bash
grep 'foo' /bar.txt # Search for 'foo' in file 'bar.txt'
grep 'foo' /bar -r|--recursive # Search for 'foo' in directory 'bar'
grep 'foo' /bar -R|--dereference-recursive # Search for 'foo' in directory 'bar'

```

and follow symbolic links

```
grep 'foo' /bar -l|--files-with-matches # Show only files that match
grep 'foo' /bar -L|--files-without-match # Show only files that don't match
grep 'foo' /bar -i|--ignore-case # Case insensitive search
grep 'foo' /bar -x|--line-regexp # Match the entire line
grep 'foo' /bar -C|--context 1 # Add N line of context above and below
each search result
grep 'foo' /bar -v|--invert-match # Show only lines that don't match
grep 'foo' /bar -c|--count # Count the number lines that match
grep 'foo' /bar -n|--line-number # Add line numbers
grep 'foo' /bar --colour # Add colour to output
grep 'foo|bar' /baz -R # Search for 'foo' or 'bar' in directory 'baz'
grep --extended-regexp|-E 'foo|bar' /baz -R # Use regular expressions
egrep 'foo|bar' /baz -R # Use regular expressions
...

```

### ### Replace in Files

```
``bash
sed 's/fox/bear/g' foo.txt # Replace fox with bear in foo.txt and output
to console
sed 's/fox/bear/gi' foo.txt # Replace fox (case insensitive) with bear in
foo.txt and output to console
sed 's/red fox/blue bear/g' foo.txt # Replace red with blue and fox with bea
r in foo.txt and output to console
sed 's/fox/bear/g' foo.txt > bar.txt # Replace fox with bear in foo.txt and sav
e in bar.txt
sed 's/fox/bear/g' foo.txt -i|--in-place # Replace fox with bear and overwrite f
oo.txt
...

```

### ## Symbolic Links

```
``bash
ln -s|--symbolic foo bar # Create a link 'bar' to the 'foo' folder
ln -s|--symbolic -f|--force foo bar # Overwrite an existing symbolic link 'bar'

```

```
ls -l # Show where symbolic links are pointing
...
```

## ## Compressing Files

### ### zip

Compresses one or more files into \*.zip files.

```
``bash
```

```
zip foo.zip /bar.txt # Compress bar.txt into foo.zip
zip foo.zip /bar.txt /baz.txt # Compress bar.txt and baz.txt into foo.zip
zip foo.zip /{bar,baz}.txt # Compress bar.txt and baz.txt into foo.zip
zip -r|--recurse-paths foo.zip /bar # Compress directory bar into foo.zip
...
```

### ### gzip

Compresses a single file into \*.gz files.

```
``bash
```

```
gzip /bar.txt foo.gz # Compress bar.txt into foo.gz and then delete bar.txt
gzip -k|--keep /bar.txt foo.gz # Compress bar.txt into foo.gz
...
```

### ### tar -c

Compresses (optionally) and combines one or more files into a single \*.tar, \*.tar.gz, \*.tpz or \*.tgz file.

```
``bash
```

```
tar -c|--create -z|--gzip -f|--file=foo.tgz /bar.txt /baz.txt # Compress bar.txt and baz.txt into foo.tgz
tar -c|--create -z|--gzip -f|--file=foo.tgz /{bar,baz}.txt # Compress bar.txt and baz.txt into foo.tgz
```

```
tar -c|--create -z|--gzip -f|--file=foo.tgz /bar # Compress directory b
ar into foo.tgz
...

```

## ## Decompressing Files

### ### unzip

```
``bash
unzip foo.zip # Unzip foo.zip into current directory
...

```

### ### gunzip

```
``bash
gunzip foo.gz # Unzip foo.gz into current directory and delete foo.gz
gunzip -k|--keep foo.gz # Unzip foo.gz into current directory
...

```

### ### tar -x

```
``bash
tar -x|--extract -z|--gzip -f|--file=foo.tar.gz # Un-compress foo.tar.gz into cur
rent directory
tar -x|--extract -f|--file=foo.tar # Un-combine foo.tar into current dire
ctory
...

```

## ## Disk Usage

```
``bash
df # List disks, size, used and available space
df -h|--human-readable # List disks, size, used and available space in a huma
n readable format

```

```
du # List current directory, subdirectories and file sizes

```

```
du /foo/bar # List specified directory, subdirectories and file sizes
du -h|--human-readable # List current directory, subdirectories and file sizes i
n a human readable format
du -d|--max-depth # List current directory, subdirectories and file sizes wit
hin the max depth
du -d 0 # List current directory size
...

```

## ## Memory Usage

```
``bash
free # Show memory usage
free -h|--human # Show human readable memory usage
free -h|--human --si # Show human readable memory usage in power of 100
0 instead of 1024
free -s|--seconds 5 # Show memory usage and update continuously every f
ive seconds
...

```

## ## Packages

```
``bash
apt update # Refreshes repository index
apt search wget # Search for a package
apt show wget # List information about the wget package
apt list --all-versions wget # List all versions of the package
apt install wget # Install the latest version of the wget package
apt install wget=1.2.3 # Install a specific version of the wget package
apt remove wget # Removes the wget package
apt upgrade # Upgrades all upgradable packages
...

```

## ## Shutdown and Reboot

```
``bash
shutdown # Shutdown in 1 minute

```

```
shutdown now "Cya later" # Immediately shut down
shutdown +5 "Cya later" # Shutdown in 5 minutes
```

```
shutdown --reboot # Reboot in 1 minute
shutdown -r now "Cya later" # Immediately reboot
shutdown -r +5 "Cya later" # Reboot in 5 minutes
```

```
shutdown -c # Cancel a shutdown or reboot
```

```
reboot # Reboot now
reboot -f # Force a reboot
```

## ## Identifying Processes

```
``bash
```

```
top # List all processes interactively
htop # List all processes interactively
ps all # List all processes
pidof foo # Return the PID of all foo processes
```

```
CTRL+Z # Suspend a process running in the foreground
bg # Resume a suspended process and run in the background
fg # Bring the last background process to the foreground
fg 1 # Bring the background process with the PID to the foreground
```

```
sleep 30 & # Sleep for 30 seconds and move the process into the back
ground
jobs # List all background jobs
jobs -p # List all background jobs with their PID
```

```
lsdf # List all open files and the process using them
lsdf -itcp:4000 # Return the process listening on port 4000
...

```

## ## Process Priority

Process priorities go from -20 (highest) to 19 (lowest).

```
``bash
nice -n -20 foo # Change process priority by name
renice 20 PID # Change process priority by PID
ps -o ni PID # Return the process priority of PID
...

```

### ## Killing Processes

```
``bash
CTRL+C # Kill a process running in the foreground
kill PID # Shut down process by PID gracefully. Sends TERM signal.
kill -9 PID # Force shut down of process by PID. Sends SIGKILL signal.
pkill foo # Shut down process by name gracefully. Sends TERM signal.
pkill -9 foo # force shut down process by name. Sends SIGKILL signal.
killall foo # Kill all process with the specified name gracefully.
...

```

### ## Date & Time

```
``bash
date # Print the date and time
date --iso-8601 # Print the ISO8601 date
date --iso-8601=ns # Print the ISO8601 date and time

time tree # Time how long the tree command takes to execute
...

```

### ## Scheduled Tasks

```
``pre
* * * * *
Minute, Hour, Day of month, Month, Day of the week
...

```

```

```bash
crontab -l          # List cron tab
crontab -e          # Edit cron tab in Vim
crontab /path/crontab # Load cron tab from a file
crontab -l > /path/crontab # Save cron tab to a file

* * * * * foo      # Run foo every minute
*/15 * * * * foo   # Run foo every 15 minutes
0 * * * * foo      # Run foo every hour
15 6 * * * foo     # Run foo daily at 6:15 AM
44 4 * * 5 foo     # Run foo every Friday at 4:44 AM
0 0 1 * * foo      # Run foo at midnight on the first of the month
0 0 1 1 * foo      # Run foo at midnight on the first of the year

at -l              # List scheduled tasks
at -c 1            # Show task with ID 1
at -r 1            # Remove task with ID 1
at now + 2 minutes # Create a task in Vim to execute in 2 minutes
at 12:34 PM next month # Create a task in Vim to execute at 12:34 PM next
month
at tomorrow        # Create a task in Vim to execute tomorrow
...

```

HTTP Requests

```

```bash
curl https://example.com # Return response body
curl -i|--include https://example.com # Include status code and HT
TP headers
curl -L|--location https://example.com # Follow redirects
curl -o|--remote-name foo.txt https://example.com # Output to a text file
curl -H|--header "User-Agent: Foo" https://example.com # Add a HTTP heade
r
curl -X|--request POST -H "Content-Type: application/json" -d|--data '{"fo
o":"bar"}' https://example.com # POST JSON

```

```

curl -X POST -H --data-urlencode foo="bar" http://example.com
POST URL Form Encoded

wget https://example.com/file.txt . # Download a file to the current directory
wget -O|--output-document foo.txt https://example.com/file.txt # Output to a file with the specified name
...

Network Troubleshooting

``bash
ping example.com # Send multiple ping requests using the ICMP protocol
ping -c 10 -i 5 example.com # Make 10 attempts, 5 seconds apart

ip addr # List IP addresses on the system
ip route show # Show IP addresses to router

netstat -i|--interfaces # List all network interfaces and in/out usage
netstat -l|--listening # List all open ports

traceroute example.com # List all servers the network traffic goes through

mtr -w|--report-wide example.com # Continually list all servers the network traffic goes through
mtr -r|--report -w|--report-wide -c|--report-cycles 100 example.com # Output a report that lists network traffic 100 times

nmap 0.0.0.0 # Scan for the 1000 most common open ports on localhost
nmap 0.0.0.0 -p1-65535 # Scan for open ports on localhost between 1 and 65535
nmap 192.168.4.3 # Scan for the 1000 most common open ports on a remote IP address
nmap -sP 192.168.1.1/24 # Discover all machines on the network by ping'ing

```

```
them
...
```

## ## DNS

```
``bash
host example.com # Show the IPv4 and IPv6 addresses

dig example.com # Show complete DNS information

cat /etc/resolv.conf # resolv.conf lists nameservers
...
```

## ## Hardware

```
``bash
lsusb # List USB devices
lspci # List PCI hardware
lshw # List all hardware
...
```

## ## Terminal Multiplexers

Start multiple terminal sessions. Active sessions persist reboots. `tmux` is more modern than `screen`.

```
``bash
tmux # Start a new session (CTRL-b + d to detach)
tmux ls # List all sessions
tmux attach -t 0 # Reattach to a session

screen # Start a new session (CTRL-a + d to detach)
screen -ls # List all sessions
screen -R 31166 # Reattach to a session
```

```
exit # Exit a session
...
```

## ## Secure Shell Protocol (SSH)

```
``bash
ssh hostname # Connect to hostname using your current user name
over the default SSH port 22
ssh -i foo.pem hostname # Connect to hostname using the identity file
ssh user@hostname # Connect to hostname using the user over the de
fault SSH port 22
ssh user@hostname -p 8765 # Connect to hostname using the user over a c
ustom port
ssh ssh://user@hostname:8765 # Connect to hostname using the user over a
custom port
...
```

Set default user and port in `~/.ssh/config`, so you can just enter the name ne  
xt time:

```
``bash
$ cat ~/.ssh/config
Host name
 User foo
 Hostname 127.0.0.1
 Port 8765
$ ssh name
...
```

## ## Secure Copy

```
``bash
scp foo.txt ubuntu@hostname:/home/ubuntu # Copy foo.txt into the specified
remote directory
...
```

## ## Bash Profile

```
- bash - `bashrc`
- zsh - `zshrc`
```

```
``bash
```

```
Always run ls after cd
```

```
function cd {
 builtin cd "$@" && ls
}
```

```
Prompt user before overwriting any files
```

```
alias cp='cp --interactive'
alias mv='mv --interactive'
alias rm='rm --interactive'
```

```
Always show disk usage in a human readable format
```

```
alias df='df -h'
alias du='du -h'
...
```

## ## Bash Script

### ### Variables

```
``bash
```

```
#!/bin/bash
```

```
foo=123 # Initialize variable foo with 123
```

```
declare -i foo=123 # Initialize an integer foo with 123
```

```
declare -r foo=123 # Initialize readonly variable foo with 123
```

```
echo $foo # Print variable foo
```

```
echo ${foo}_bar # Print variable foo followed by _bar
```

```
echo ${foo:-'default'} # Print variable foo if it exists otherwise print default
```

```
export foo # Make foo available to child processes
```

```
unset foo # Make foo unavailable to child processes
...

```

### ### Environment Variables

```
``bash
#!/bin/bash

env # List all environment variables
echo $PATH # Print PATH environment variable
export FOO=Bar # Set an environment variable
...

```

### ### Functions

```
``bash
#!/bin/bash

greet() {
 local world = "World"
 echo "$1 $world"
 return "$1 $world"
}
greet "Hello"
greeting=$(greet "Hello")
...

```

### ### Exit Codes

```
``bash
#!/bin/bash

exit 0 # Exit the script successfully
exit 1 # Exit the script unsuccessfully
echo $? # Print the last exit code
...

```

### ### Conditional Statements

#### #### Boolean Operators

- `foo` - Is true
- `!foo` - Is false

#### #### Numeric Operators

- `eq` - Equals
- `ne` - Not equals
- `gt` - Greater than
- `ge` - Greater than or equal to
- `lt` - Less than
- `le` - Less than or equal to
- `-e` foo.txt - Check file exists
- `-z` foo - Check if variable exists

#### #### String Operators

- `=` - Equals
- `==` - Equals
- `-z` - Is null
- `-n` - Is not null
- `<` - Is less than in ASCII alphabetical order
- `>` - Is greater than in ASCII alphabetical order

#### #### If Statements

```
```bash
```

```
#!/bin/bash
```

```
if [[foo = 'bar']]; then
```

```
    echo 'one'
```

```
elif [[foo = 'bar']] || [[foo = 'baz']]; then
```

```
    echo 'two'
elif [[ $foo = 'ban' ]] && [[ $USER = 'bat' ]]; then
    echo 'three'
else
    echo 'four'
fi
...

```

Inline If Statements

```
``bash
#!/bin/bash

[[ $USER = 'rehan' ]] && echo 'yes' || echo 'no'
...

```

While Loops

```
``bash
#!/bin/bash

declare -i counter
counter=10
while [ $counter -gt 2 ]; do
    echo The counter is $counter
    counter=counter-1
done
...

```

For Loops

```
``bash
#!/bin/bash

for i in {0..10..2}
do

```

```

    echo "Index: $i"
done

for filename in file1 file2 file3
do
    echo "Content: " >> $filename
done

for filename in *;
do
    echo "Content: " >> $filename
done
...

#### Case Statements

``bash
#!/bin/bash

echo "What's the weather like tomorrow?"
read weather

case $weather in
    sunny | warm ) echo "Nice weather: " $weather
    ;;
    cloudy | cool ) echo "Not bad weather: " $weather
    ;;
    rainy | cold ) echo "Terrible weather: " $weather
    ;;
    * ) echo "Don't understand"
    ;;
esac
...

```