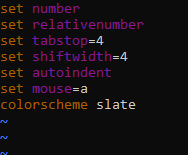
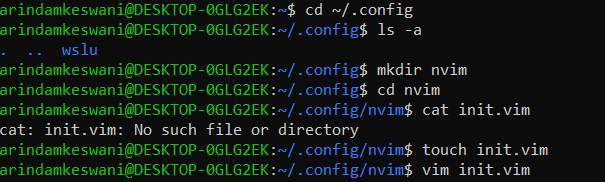
**Vim**

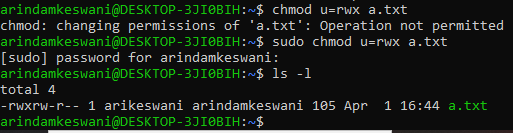
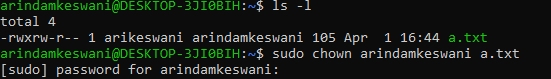
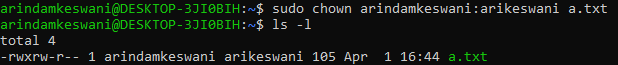
* i - insert (go into text editor) (Explore a,o,A,O,I)
* Esc - come out of editing mode
* :x / :wq - Save and exit
* :q - Exit without saving (if it gives an error, :q!)
* :set number - give line numbers
* :set relativenumber - give relative line numbers
* <number> followed by arrow key- Move ‘number’ units in that direction
* h - move left
* j - move down
* k - move up
* l - move right
* :set mouse=a - allows pointing with mouse
* :set tabstop=<number> - Moves tab whitespaces
* :set shiftwidth=<number> - Indentation using shift
* colorscheme <option?> - change theme
* Settings change to default after exiting file
  + To make changes permanent, run “~/.vimrc”
  + 
  + Save and exit
* Cheat sheet
* 
* u (in normal mode) - Undo
* Ctrl + r (in normal mode)= Redo
* You can use numbers with undo and redo to commit the operation <number> times
* There is a third mode: Visual. Use “v” to enter it
  + Arrow keys select a text area in visual mode
  + d - deletes the selected text
  + y - for yanking/copying the selected text
  + p - paste
* w - Jump to the next word (E.g. d2w - Delete 2 words, diw - Deletes the word where the cursor currently is)
* b - Jump to prev word
* e - jumps to end of word
* 0 - Beginning of the line
* $ - End of the line
* In a line which says e.g.
  + print(“hello world”)
  + If I run **ci”** (change inner quotation marks) or **yi”** (yank inner quotation marks) it will remove/copy everything between the quotation marks if the cursor in between them
  + Same for parenthesis
* % - (Cursor is on parenthesis) To jump between brackets
* d% - Delete everything inside parenthesis (along with brackets)
* t<char> - jump to right before character (first occurrence from cursor)
* f<char> - jump to character (first occurrence from cursor)
* T and F do the respective options in reverse order (search backwards)
* gg+shift+g - Jump to end of file (check this once)
* :<number> - Jump to particular line number
* >> - Indent right
* << - Indent left
* Select a part of text + > (or <) - Indent
* Shift+v + arrow (up or down) - Select full lines
* Ctrl + v + arrow - Select row/column wise
* = - Auto indent
* /<search term>+Enter = Search for a term. ‘n’ to move to next occurrence, ‘N’ to move to prev occurrence
* Can do the same with **?**
  + **n** will move up, **N** will move forward
* m<char> - Set checkpoint
  + ‘<char> - Visit this checkpoint
* zz - Get curr line in centre of screen
* :%s/<original>/<new>/g - Search and replace in the entire file
* :s/<original>/<new>/g - Search and replace in selected area
* . - Repeat last command
* :reg - view register
* Deleting in vim means cutting
* “<number>p - Get command at <number> from register and paste it
* “0p - Paste what we copied last
* qa - Record your own macro/key binding
* sudo apt install neovim - Install neovim
* Or
* sudo apt-get update

sudo apt-get install neovim

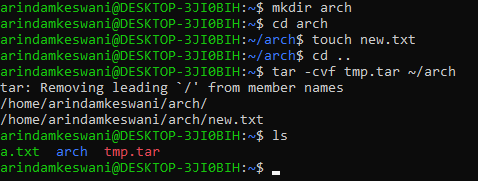
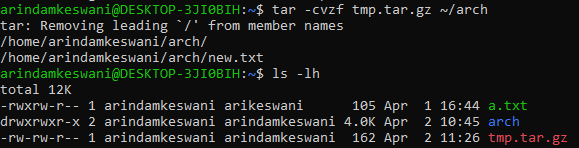
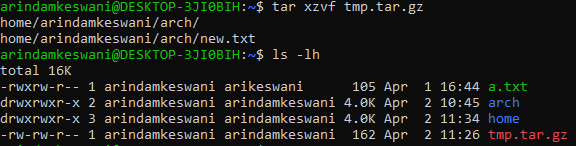
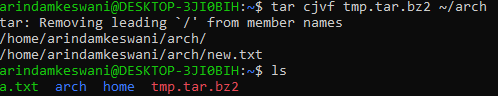
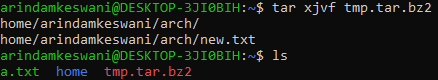
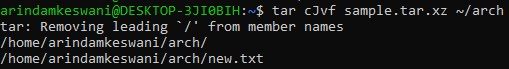
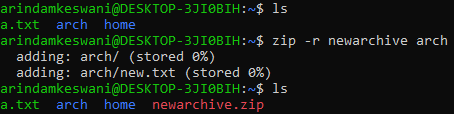
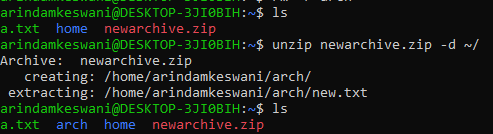
* + To add plugins, go [here](https://github.com/junegunn/vim-plug)
  + Steps to configure nvim
  + 
  + Set config as we did in .vimrc file
  + :PlugInstall - installs all plugins inside the init.vim file

**Sudo, Users, Groups, Permissons**

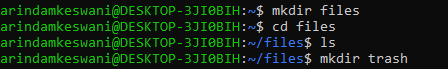
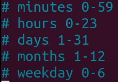
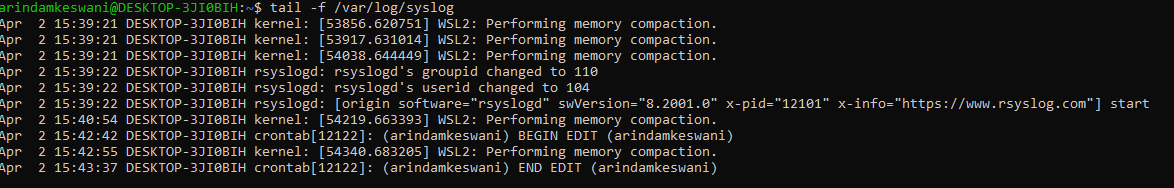
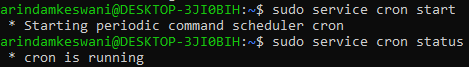
When you install Linux, you would need to create a user, who has a username and a password

* Linux has many users by default
* Root user is an important one
  + Referred to as the super user or super admin, as it has all the privileges to perform any kind of task such as config, add or remove softwares etc.
  + Recommended to do day to day work without root user privileges
* sudo - Substitute user do
  + Allows any user(for eg. one that may have restricted access) to execute any command with root user’s privileges
* To check which user is currently logged in, use **whoami** command
  + Now try the same with **sudo whoami**
* Users and groups are managed in config files inside the **etc** folder which also has other system-wide configurations
* **cat etc/passwd/** shows all the users on the system (you might want to go the the home directory using “cd /”)
  + 
  + Each row has the username, placeholder for the password (“x”)
  + The two numbers that follow are user ID and group ID respectively
    - Group ID tells us which group this user is a part of.
    - By default Linux creates a group of the same name and ID as the user ID when a user is created
  + Next column is the comment column
  + After that is the home folder for that specific user
    - Notice how root user has its home directory in the “/root” directory
    - Other users have their home folder in the “/home” directory
  + Last column is the shell that the user will use by default
  + It is not recommended to edit this file directly
* Add a new user
  + **useradd** command
  + This command will be executed with root user’s privileges so use it with **sudo**
  + 
  + -m : creates home directory
  + -s: specify the shell followed by the shell name itself
  + -c: Comment
  + Lastly, put the username
  + Following is the output in the psswd file:
  + 
* Change password
  + Use **passwd** to change the password of a user .
* Switch users
  + **su username** - switch to another user
    - Enter the password and hit enter
  + To go to the home folder for a user, use **cd ~**
* Groups
  + Groups are stored in /etc/group/ file
  + 
  + 1st column - Group name
  + 3rd column is the group number
  + Last column (after the last colon) has the group member names
  + This means that user “arindamkeswani” can use the commands related to “audio” group
  + Try to log in using a user that is not a part of the sudo group
    - Run **sudo whoami**
    - Check the output
    - 
    - To implement this we will make use of **usermod**  command
      * You can modify a lot of attributes
      * Log-in as a user who has root privileges
      * **sudo usermod -aG sudo username**
      * This will add the specified group (sudo) to the list of groups that the user with the name “username” is a part of
      * Log in using the other user again and try **sudo whoami** . It should now work as intended
      * [Execute sudo without password](https://askubuntu.com/questions/147241/execute-sudo-without-password)
    - **sudo groupadd <group name>** - Adds a new group with the specified name
      * 
      * Group added successfully
      * Try to add a couple of users to this group
      * Output: 
    - Delete users and group
      * Make sure that you have logged off from the user who you wish to delete
      * **sudo userdel username** - Deletes the user
        + Notice that the user has been removed from all of its groups too
      * **sudo groupdel groupname** -Deletes the group
* **Permissions**
  + **ls -l** - To view files and other details in the current directory
  + 
  + 
    - These are file permissions which describe what users and groups can do with the particular file
    - First character indicated file type
      * ‘-’ means that it is a file
      * ‘d’ means that it is a directory
    - Next three characters specify file permissions for the file owner
    - r: Read, w: Write, x: Execute
      * Here, the file owner does not have execute permissions
    - Next three characters specify file permissions for the group owner in the same order of **rwx**
    - Next three chars are permissions for all other users in the Linux OS that are not the file owner or group member
    - Note: Root user has all three permissions irrespective of whether or not it is mentioned
    - If you try to edit a file via a user who does not have permissions for it, it will display a similar message
    - 
  + 
    - File owner and Group Owner respectively
  + 
    - File name for which we are viewing these details
  + **chmod** - change file permissions
    - 2 variations:
      * Text mode
      * Options: **u**(file owner), **g**(group), **o**(other users)
      * 
      * Say I want to give all permissions to the file owner
      * chmod u=rwx a.txt
      * Output:
      * 
      * To remove a permission, simple don’t mention the relevant alphabet
        + **chmod u=rw a.txt** will revoke execute right for the file owner
        + **chmod u+x a.txt** will add execute permissions and leave the rest of the permissions as it is. Using ‘-’ instead would remove the permission
      * Numeric mode
      * r=4, w=2, x=1. Permission for an entity (user, group, or other user) is the sum of the relevant permissions
      * **sudo chmod 761 a.txt** will give the following permissions:
        + rwx (4+2+1) to file owner
        + rw (4+2) to group
        + x (1) to other users
  + **chown**
    - Change ownership of a file
    - 
    - 
    - To change group:
    - 
    - Specify group name after colon
    - To change ownership of all files inside a folder add the “-R” flag after chown

**Compressing and Archiving files**

* tar
  + It is an archival tool. Allows us to create an archived file
  + Look at the following steps to create a tar file
  + 
    - Created a new directory, and a new file in this directory
    - Used **tar -cvf tmp.tar ~/arch** to make this file, where the flags mean the following
      * **c** - create
      * **v** - verbosely list the files processed
      * **f** - specify name of new tar file (given as tmp.tar)
    - After this, you can see that the tar file is now in the directory
* **xz**
  + Very good compression algorithm, but resource intensive
  + Compression syntax
  + 
  + Decompression syntax
  + 
  + Explore -e flag for compression
* **gzip**
  + Compression algorithm
  + Compression syntax
    - Current size
    - 
    - After compression
    - 
    - 
  + Decompression syntax
  + 
  + Explore –fast and –best flags
* **bzip2**
  + Compression algorithm
  + Compression syntax
  + 
  + 
  + Decompression syntax
  + 
* Why was an archived file created before we compressed it?
  + The idea is to use the two techniques together(archiving and compression)
  + We can directly compress the file while archiving it by adding flags in the **tar** command
    - Syntax:
    - 
    - This will compress the file using gzip (as specified by the z flag)
    - Decompression syntax
    - 
    - **x** flag is used to specify that we wish to perform extraction
    - **z** flag is used to specify that we wish to decompress using gzip’s decompression algo
  + To do the same process using bzip2
    - Compression
    - 
    - Decompression
    - 
  + To do the same process using xz
    - Compression
    - 
    - Decompression
    - 
  + You can download utilities for zip
    - Usage
    - Zipping
    - 
    - -r flag is used as we are zipping a directory
    - Unzipping
    - 

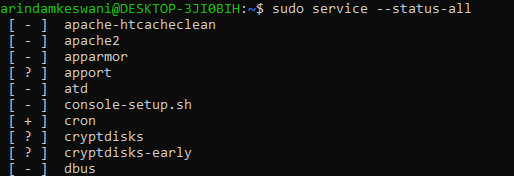
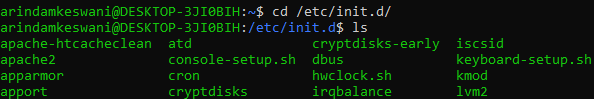
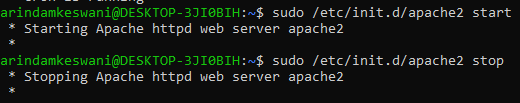
**CRON Jobs**

* Objective: Schedule a task, script, or process to run
* CRON is a time-based scheduling utility that allows us to run tasks periodically or even on events like system reboot
* **crontab -e**
  + This will launch an editor (an empty file with some comments), allow us to control tabs which are used to create these periodic jobs
  + It will ask you to choose a default editor, choose the appropriate number
    - The default editor can be changed using **select-editor** command
  + There are 5 “\*”, like “\* \* \* \* \*” that you have to check, which represent minute, hour, day of month, month, week. You can place numbers in
* It is not advisable to use CRON jobs for tasks that include interactivity or tasks involving graphical apps, e.g. opening a terminal.
* For this demo, make the following directory structure
* 
* Make a random file inside the trash folder
* In the cron file, write the following:
  + 
  + It will check the trash folder every minute and continuously clear it
  + First “\*” is the number of minutes we wish to wait before executing the command (between 0 and 59). Same is the case with the other “\*”
  + 
  + 
    - This would run the script every sunday
  + 
    - This would run the script every January
  + Note: Hours follow military time
  + After exiting, it will give a similar message
  + 
  + To check if it is working properly,
  + 
  + 
  + In the last three rows, you will notice the logs of the CRON job that we just scheduled
  + In case you are not able to find the syslog file, look [here](https://answers.microsoft.com/en-us/windows/forum/all/location-of-syslog-file-in-windows-10-ubuntu-bash/ae40a2d2-005e-4130-b830-d90e768e6fb5) and [here](https://stackoverflow.com/questions/10979435/where-does-linux-store-my-syslog)
  + In case your script is not working properly, make sure that it is executable using chmod
    -  is the folder in which your crontab is
  + 
  + [In case your cron job does not run properly, check if the service is running](https://linuxhint.com/check_working_crontab/#:~:text=Method%20%23%201%3A%20By%20Checking%20the%20Status%20of%20Cron%20Service&text=Running%20the%20%E2%80%9Csystemctl%E2%80%9D%20command%20along,working%20perfectly%20well%2C%20otherwise%20not.)
  + 
  + 
  + Sample input of CRON job
  + 
  + Sample output (using **tail -f /var/log/syslog**)
  + 
  + Output in destination file:
  + 
  + Instead of using “\*”, we can even use the following
  + 
  + Example:
  + . This will delete the files in the specified directory once every hour
  + You can even run bash scripts using CRON jobs
    - 
  + **crontab -l** shows you the contents of the crontab file
  + **crontab -r** removes the crontab file
  + **crontab -u username -e** opens crontab for a specific user

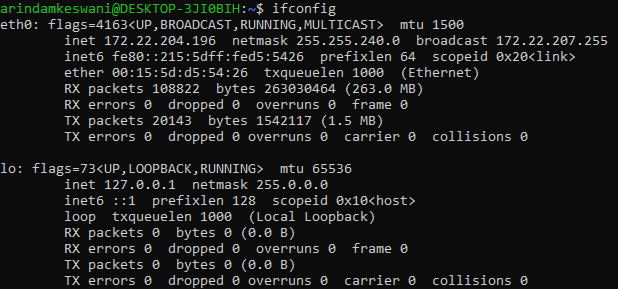
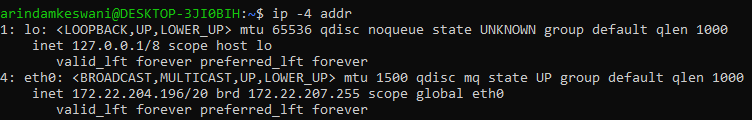
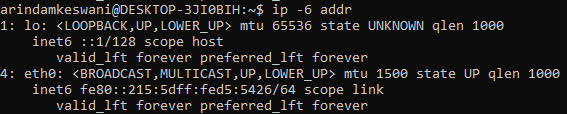
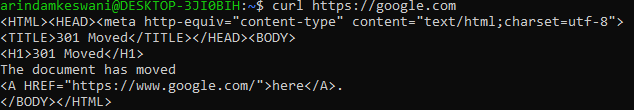
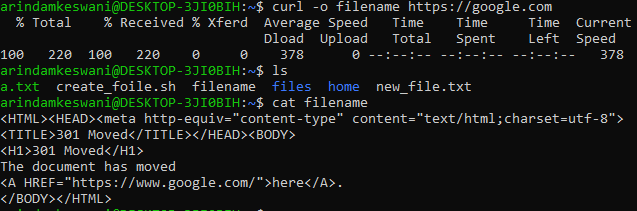
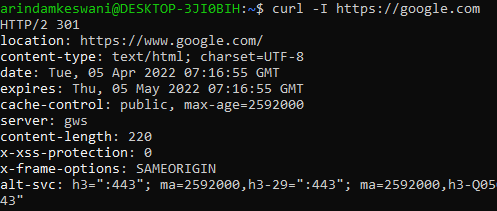
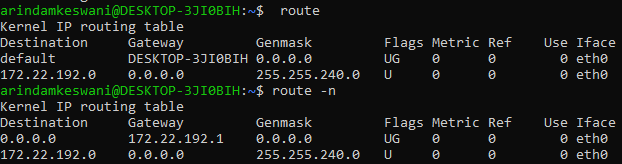
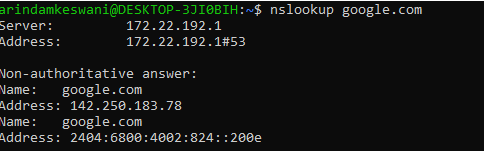
**Managing Processes: systemd, systemctl, package**

**Management**

* Systemd, systemctl, service, init.d
  + Helps you manage running services, i.e. start, stop, and restart services
  + E.g. on an OS like windows, you cannot use an application e.g. Word without actually opening it. If the app is installed, but it is not open/running, we cannot open it. Similarly, a lot of times when we encounter a problem with an application, it turns out that the service is not running
  + If we change the configuration of an application, the changes will not be reflected in the service till we restart it
  + Services are basically applications that run on the server
  + **systemd** is one of the ways to control services
  + Another way is to use the **service** command
  + Third way is via **init.d** scripts, though it is deprecated
  + Services are referred to as units

* **Service** command
  + sudo service –status-all
    - Displays all services
    - 
    - + means that the service is running
    - - means that the service is not running
    - ? means that service does not have a status command, so **service** command cannot determine whether the service in question is running or not
  + To check a particular service, e.g. cron
  + 
  + To stop a service
  + 
* **init.d**
  + They are deprecated
  + Go to the script
    - cd /etc/init.d/
    - ls (check out the contents)
    - 
  + To see the details of a service, e.g. apache2
  + 
  + To start and stop a service
  + 
* **Package management (Debian, Fedora, Arch)**
  + Ubuntu uses **apt** command for package management
  + Before managing packages, refresh the package repository index, which can be done with:
    - **sudo apt update** (Debian)
    - pacman -Sy (Arch)
  + Debian uses “apt”
  + Fedora uses “dnf”. It used “yum” before. Same for CentOS.
  + Arch Linux uses “pacman”
  + Installing a package
    - To install a package, say apache, run the following
      * sudo apt install apache2 (Debian)
      * dnf install httpd (Fedora)
      * pacman -S apache
  + Removing a package
    - **sudo apt remove apache2** (Debian)
    - **dnf remove httpd** (Fedora) (Dependencies are removed too)
    - **pacman -R apache**
    - Some dependencies that are installed with the package are left behind while removing the package. To remove them, use **apt autoremove** (Debian)
      * This will all dependencies that are no longer required, as their packages are removed
    - On arch, **pacman -Qdt** lists the orphaned packages
  + Update the server
    - Updates can be varied, e.g. security patches
    - **sudo apt upgrade** (Debian)
    - **dnf upgrade** (Fedora)
  + To search for a package that we wish to install
    - Debian
      * **apt-cache search apache** (apache is the keyword we are searching for)
      * To limit the results, add “ | more” to the previous command
    - Fedora
      * **dnf search apache**
    - Arch
      * **pacman -Ss apache**

**Networking related commands**

* **ifconfig**
  + Gives info about network interface devices
  + 
  + “eth0” is a hardware device (the physical ethernet connection), its IPv4 Address is  and IPv6 address is 
  + “lo” is a software/virtual device (loopback)
* **ip** 
  + To just find the IP Address of the machine
    - **ip -4 addr** (IPv4 address)
    - 
    - **ip -6 addr** (IPv6 address)
    - 
* **netstat**
  + Shows detailed info of network activity
  + To show all ports listening to TCP/UDP connections
    - **netstat -a**
  + To show all the TCP connections
    - **netstat -at**
  + All active ports
    - **netstat -l**
  + Ports listening to UDP
    - **netstat -u**
* **curl**
  + Used to test an endpoint or send a req to a URL
  + By default, GET requests are sent
  + 
  + Post request
    - **curl -X POST <url>**
      * A method is expected after “-X”
  + Send data w post request
    - curl -X POST --data "p1=value1&p2=value2" <https://google.com>
    - curl -X POST -d parm1=value <https://google.com>
  + Save server response to a file
    - 
  + To see the header
    - curl -I <url>
    - 
  + Send header with request
    - Example
    - **curl –header “Content-type:application/json” -X POST -d param1=x** [**https://google.com**](https://google.com)
* **ping**
  + Sends packets to an endpoint and tells us whether we are getting a response for it. Used to check whether a domain is online
  + **ping <url>**
* **route**
  + Shows us how traffic is routed from our computer to the outside world
  + **route -n**
    - Shows us IP Addresses of the gateways
    - 
* **dhclient**
  + used to obtain, renew, and release a clients IP address, subnet mask, default gateway, and DNS servers from a DHCP server
  + **sudo dhclient -v -r**
    - Releases DHCP address
  + **sudo dhclient -v** 
    - Obtains the DHCP address
* **nslookup** (deprecated)
  + “Name server lookup”
  + Command for getting information from the DNS server, i.e. is queries the DNS to get domain name/IP address
  + 
  + [Usage](https://geek-university.com/nslookup-command/)

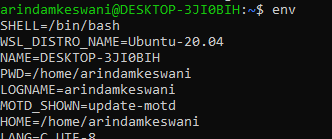
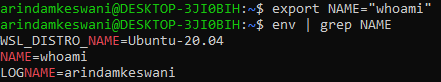
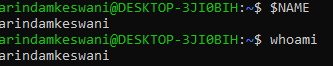
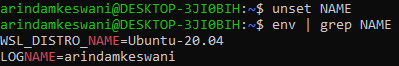
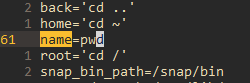
[**Environment and Shell Variables**](https://www.youtube.com/watch?v=NNp0SN5x1wM&ab_channel=Pedagogy)

Variable: A string of characters to which a value is assigned

This value could be anything ranging from a string, filepath, command, etc.

Any process can create and use variables to re-use values

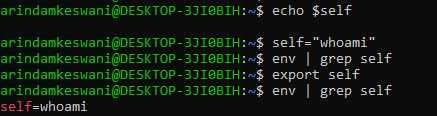
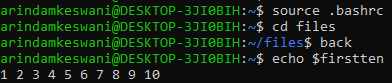
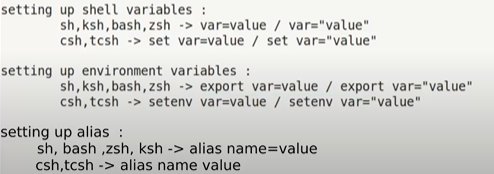
Two types of variables in Linux:

* Environment variables
  + Say there are 4 processes running in the following order:
  + P1 -> P2 -> P3 -> P4
  + If a variable was defined in P2, and if this variable was an environment variable, then all subsequent processes (P3 and P4) would be able to use it
  + Shell processes can also create an environment variable
  + Hands-on:
  + To print all environment variables in use:
    - 
    - (“printenv” can also be used for the same purpose)
  + To set your own env variable:
    - **export NAME=”VALUE”**
      * Single quotes can also be used
      * If the value does not have spaces, quotes can be omitted
      * Make the variable and check in the following ways:
      * 
      * 
      * Using the name with preceding “$” sign, replaces the variable name with the value that we have assigned to it
      * 
  + To remove the env variable
    - **unset VARNAME**
      * 
* Shell variables
  + Can only be created by shell processes
  + Cannot be used by subsequent processes
    - E.g. if the process is bash (shell process), any variable created in it cannot be accessed by any other process
  + Hands-on:
  + To set the shell variable
    - **name=”value”**
    - ****
    - To view all the commands that have been set-up, use ”set” command (but this gives a lot of information that may not be useful to us)
    - A better idea is to open the output in a text editor and analyse the top few lines, which shows the variables. You can try searching for the variable that you just created
      * ****
  + To remove the env variable
    - **unset VARNAME**
    - ****

**Environment variables use case:**

* To send data to subsequent processes
* Set some values
  + E.g. You want to make an ssh connection to a remote host. This usually requires a unique ID to be entered, which looks something like this
  + 
  + Note that we are assuming “3456” to be the unique ID, so we can just keep it as an environment variable
  + 
  + The use that variable so that we won’t have to remember the ID everytime we want to use it
  + 
  + A preferable way to do it is by using curly braces
  + 
* You can find out the user in the following way
  + 
* Check your default shell
  + 

Note: Another way of creating an env variable is by creating a shell variable and export it

* 
* Whenever an env var is created, it also becomes the shell variable
* On restarting the terminal, shell, environment variables and aliases are removed
  + To make it persistent, edit .bashrc file
  + 
  + Save the file
  + To reflect the changes, either restart the terminal or run “source .bashrc”
  + 
  + Various shells do the above tasks using the following syntax:
  + 
  + The process for saving these variables is different for different shells as well

**Working with remote server**

SSH Protocol

* Secure Shell Protocol
* Designed as a replacement of Telnet and other shell protocols
  + Telnet was not very secure
* Uses encryption
* Connecting with a machine remotely over the internet means that the request is passed over many networks, leaving it open to be intercepted, viewed or modified
* SSH prevents this by creating a secure connection
* Therefore, SSH is used for securing the transfer of data over an unsecured network by establishing a secure connection between client and server remotely, using commands
* Files are transferred over the network with the help of security protocols such as SFTP and SCP
  + SFTP (Secure File Transfer Protocol) authenticates the server and the client
  + After authentication, SFTP establishes a secure channel between client and host computer, after which data can be transmitted
  + This data is protected using hashing

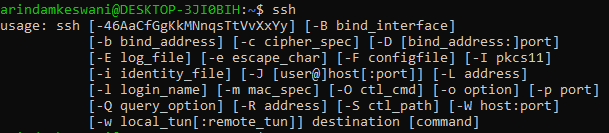
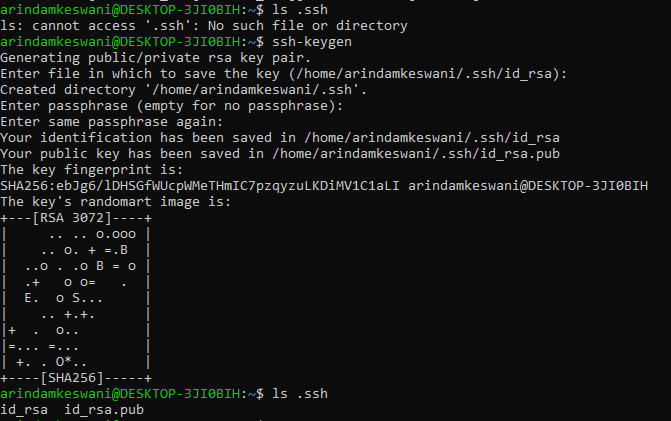
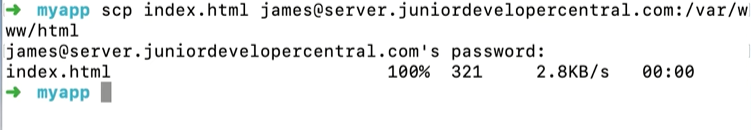
SCP

* Secure Copy Protocol
* It’s role is to transfer the data and to maintain its confidentiality

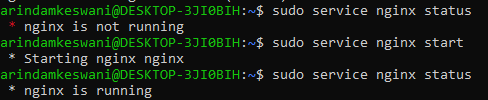
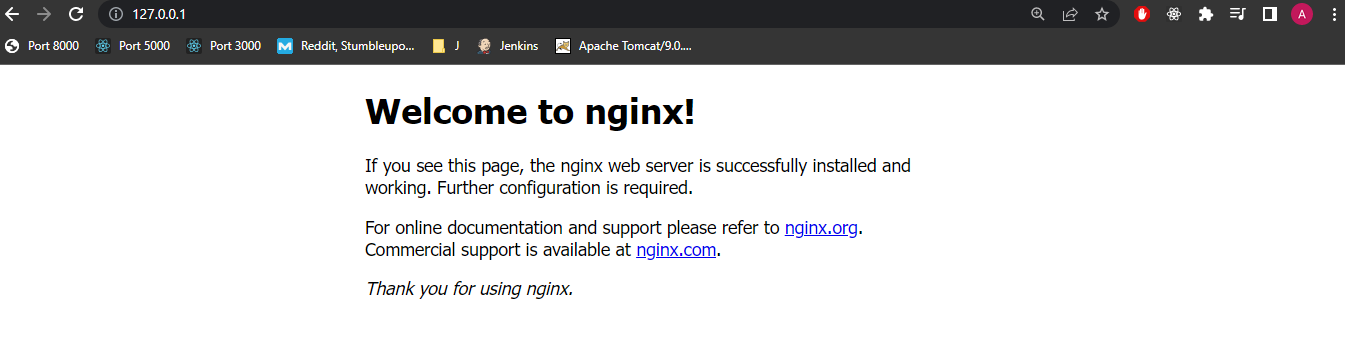
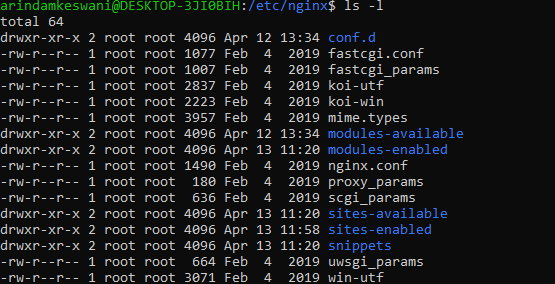
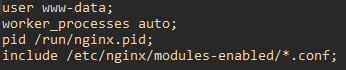
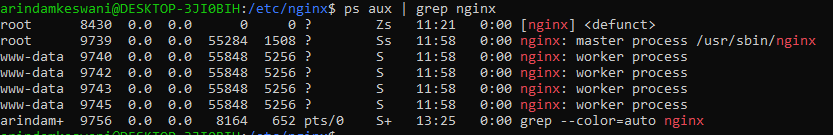
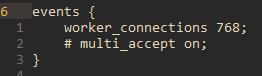
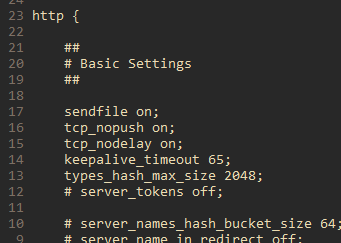
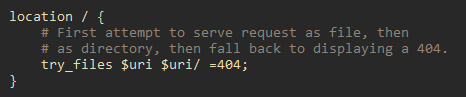
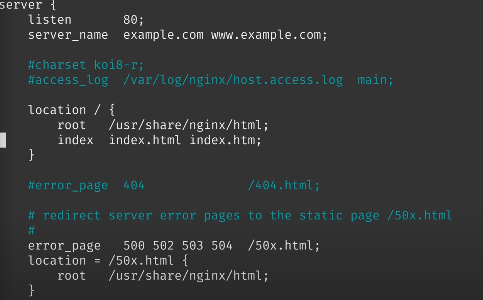
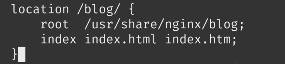
SCP and SSH together ensure that the authenticity of the client and server is checked, and confidentiality of the data is maintained.

SCP uses SSH services to transfer data and for authentication.

Hands-on:

* 
* To make an SSH connection, we need to know 2 things:
  + Server host (e.g. IP Address, domain name)
  + Username (user on the remote server that we wish to connect to)
  + Password/Key can be used for authentication as well
  + Example connection:
  + 
  + 
  + After this it may ask for a password, after which you will get access to the server and we can use any commands that the server has permitted us to use
  + If you have any ssh keys, it can be found in the “.ssh” file
  + You can generate a key using “ssh-keygen” command
  + 
  + To provide a public key to a server, check the following usage
  + 
  + Once the server has the public key, it can make the same SSH connection as before
  + Common usage of SSH is to copy files from a local machine to a remote server, e.g. GitHub
  + 
  + Note: SCP is outdated, consider using SFTP

[**Configuring NGINX on remote servers**](https://www.youtube.com/watch?v=MP3Wm9dtHSQ&ab_channel=Linode)

* [Install NGINX](https://phoenixnap.com/kb/how-to-install-nginx-on-ubuntu-20-04)
* Check if nginx is running or not
* 
* In case you see Apache webpage, check how to fix it [here](https://askubuntu.com/questions/642238/why-do-i-still-see-an-apache-site-on-nginx/642288#642288)
* If the installation has been done properly, this is what your localhost would look like
* 
* The location of this file is 
* To check out NGINX’s configuration files, go to “/etc/nginx” folder and list the contents
* 
* Check out nginx.conf file
* 
* The above 4 lines are part of the main context. These lines are not for any server in particular. They are configurations for nginx itself.
* **worker \_process auto** means that the number of worker processes running in the background will be decided automatically. It is recommended to set the number of processes=number of cores
* E.g. You can see that the worker process has the user “www-data” associated with it, which is the same user in the above screenshot
* 
* NGINX Logs can be accessed here
* 
* Try printing the contents of the access.log file. Refresh the nginx html page and print the contents again. Observe the changes
* Back in .conf file, this block shows the number of connections allowed per worker
* 
* The HTTP block is the block which is customised (if necessary) if we wish to make changes to the web server
* 
* There are other configuration files as well and these are inside the “sites-available” and “sites-enabled” folders
  + These files contain information that helps us configure the nginx server
  + E.g. In the “default” file, it mentions the port which nginx is supposed to listen
  + 
  + The “server\_name” field in this file allows us to perform name-based virtual hosting.
  + If we have more than one websites hosted on our server, we will have different configuration files for each
  + 
  + The above “location” block is served on “/” route. If anything is being accessed at the domain mentioned in the “server\_name” field, with no sub-path, then the files mentioned will be served to the user. E.g.
  + example.com will be served on the “/” route
  + 
  + We could have another one, like 
  + Note: the route is case sensitive and the content will not be served if the name does not match
  + Go to 
  + You will see an “index.html” file here
  +  <- this domain maps to ->  based on the image given above
  + Even “www.example.com” would be enough as we have already specified index.html as our default page
* To restart nginx
  + 