ASHOK IT

Learn Here.. Lead Anywhere..!!

LINUX OS

By Mr. Ashok

Phone: +91 9985 39 66 77

Location: H.No - 7-1-413/2, Beside Sonabhai
Temple, Ameerpet, Hyderabad - 500016

What is Operating System?

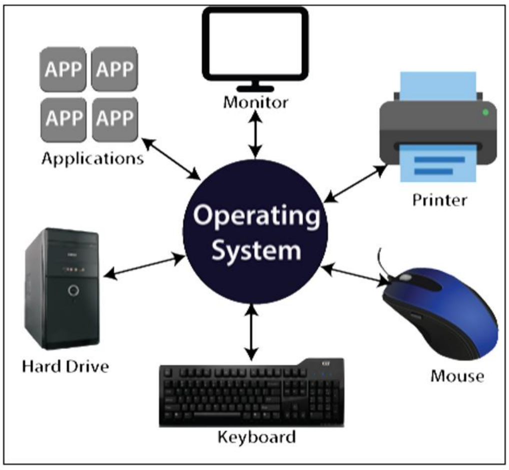
-> Operating System is a software which is used to interact with computer

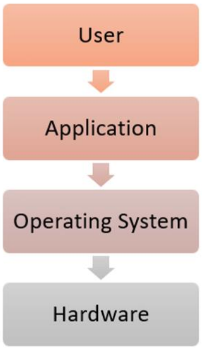
-> Operating System is acting as mediator between users and computer hardware components

-> Operating System is mandatory to use any computer

-> OS provides environment to run other applications

(Browser, Notepad, Paint, Calc)





-> The OS came into market in 1950

-> Microsoft released its first OS in 1981 (MS DOS)

Windows OS

-> Developed by Microsoft Company

-> It is having GUI (Graphical user interface)

-> It is single user based Operating System (only one person can use this at a time)

-> It is commercial (paid)

-> Less Security

-> It is recommended for personal use

- Watch Movies

- Using Internet

- Playing Games

- Attending Online classes

Linux OS

-> Linux is Community Based OS

-> Linux is Free & Open-Source OS

-> Linux is Multi User Based OS

-> Linux provides High Security

-> Linux is recommended to use for Applications, Servers, Databases etc..

Note: In real-time, we will use Linux OS only to setup infrastructure required to run our application

-> Linux OS is not only for Administrators, even developers and testers also will use Linux OS in real-time to monitor our application and application servers and application logs.

History of Linux

-> In 1991, a student 'Linus Torvalds' developed this Linux OS

-> Linux Torvalds identified some challenges in UNIX OS and he suggested some changes

for Unix OS but UNIX OS Team rejected 'Linux Torvalds 'suggestions

-> Linus Torvalds used Minux OS to develop Linux

Linus + Minux

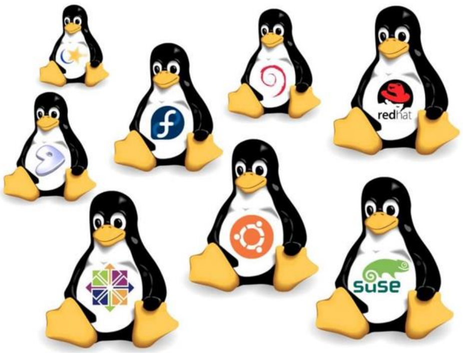
--> First Two letters from his name and last 3 letters from Minux OS

LI + NUX => LINUX

-> Linus Torvalds released LINUX OS with source code into market so that anybody can modify LINUX OS that's why it is called as Open-Source Operating System.

-> As Linux OS is open source, so many people and companies taken that Linux OS and modified according to their requirements and released into market with different names those are called as Linux Distributions.

- RED HAT
- Ubuntu OS
- Cent OS
- Fedora
- Open SUSE
- Kali Linux
- Debian
- Amazon Linux



Note: 200+ Linux Distributions are available in the market.

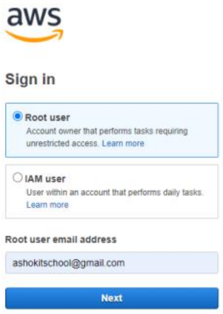
Environment Setup

We can setup Linux machine in multiple ways

1) We can install Linux OS directly in computer
2) We can setup Linux OS in windows machine using Virtual Box
3) We can setup Linux Virtual Machine Cloud Platform like AWS and Azure

Note: In this note we will see how to setup Linux VM in AWS Cloud

Create Linux VM using AWS EC2 service

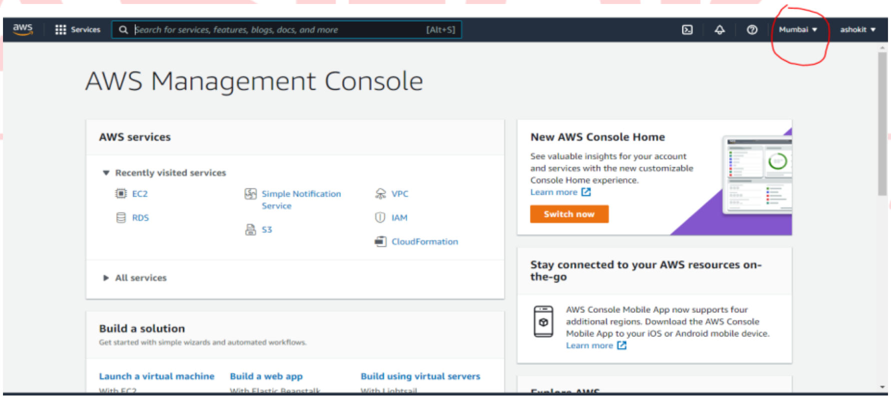


Sign in

● Root user
Account owner that performs tasks requiring unrestricted access. Learn more

○ IAM user
User within an account that performs daily tasks. Learn more

Root user email address: ashokitschool@gmail.com



AWS Management Console

AWS services

Recently visited services: EC2, RDS, Simple Notification Service, S3, VPC, IAM, CloudFormation

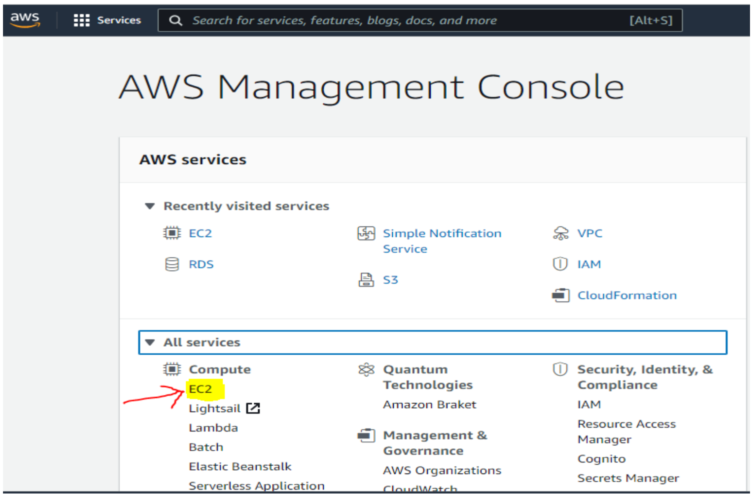
All services

New AWS Console Home
See valuable insights for your account and services with the new customizable Console Home experience. Learn more

Stay connected to your AWS resources on-the-go
AWS Console Mobile App now supports four additional regions. Download the AWS Console Mobile App to your iOS or Android mobile device. Learn more

Build a solution
Get started with simple wizards and automated workflows.

Launch a virtual machine With EC2
Build a web app With Elastic Beanstalk
Build using virtual servers With Lightsail



AWS Management Console

AWS services

Recently visited services: EC2, RDS, Simple Notification Service, S3, VPC, IAM, CloudFormation

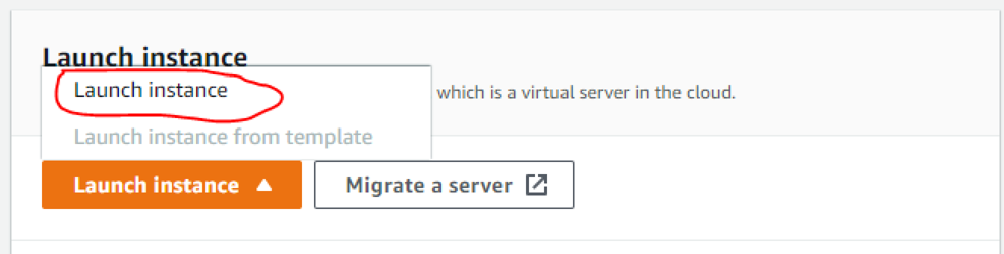
All services

Compute: EC2, Lightsail, Lambda, Batch, Elastic Beanstalk, Serverless Application

Quantum Technologies: Amazon Braket

Management & Governance: AWS Organizations, CloudWatch

Security, Identity, & Compliance: IAM, Resource Access Manager, Cognito, Secrets Manager

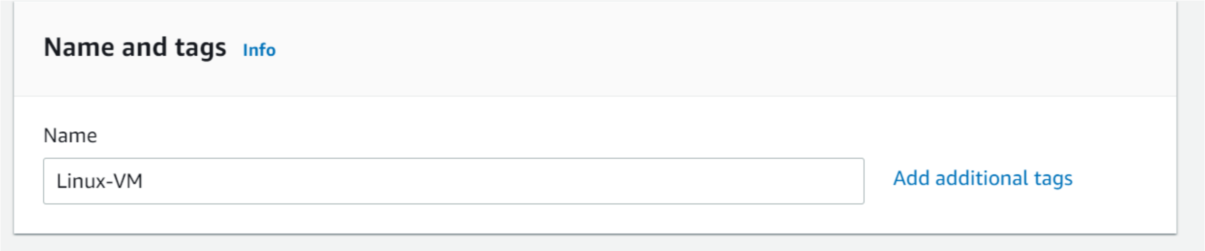


Launch instance

Launch instance

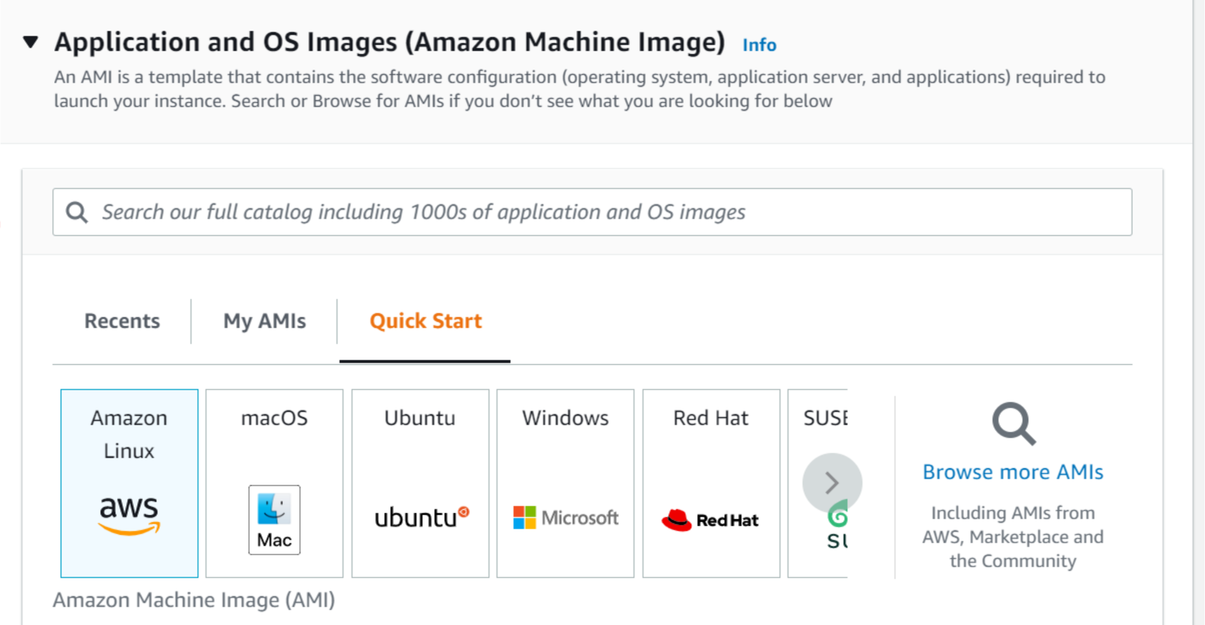
Launch instance from template

which is a virtual server in the cloud.



Name and tags Info

Name: Linux-VM

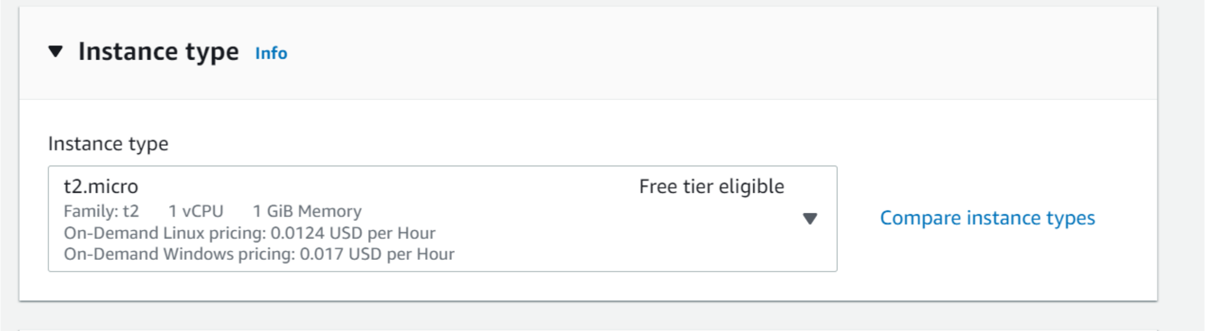


▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images
Recents
My AMIs
Quick Start
Amazon Linux
aws
macOS
Mac
Ubuntu
ubuntu
Windows
Microsoft
Red Hat
Red Hat
SUSE
SU
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

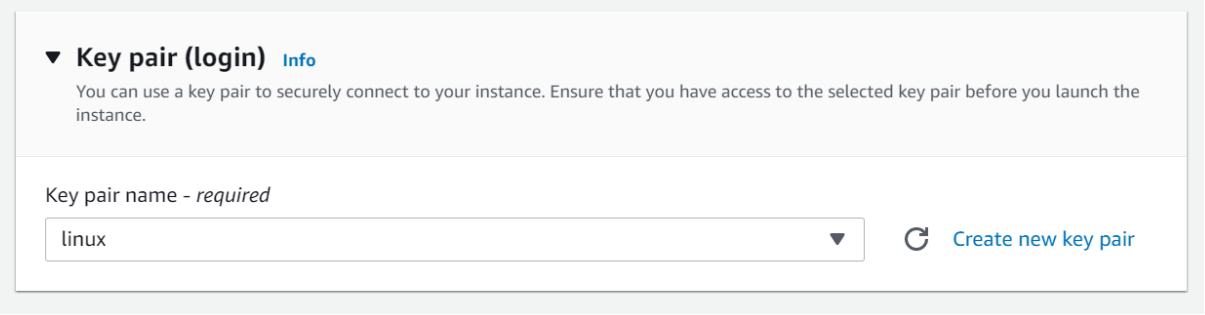
Amazon Machine Image (AMI)



▼ Instance type Info

Instance type

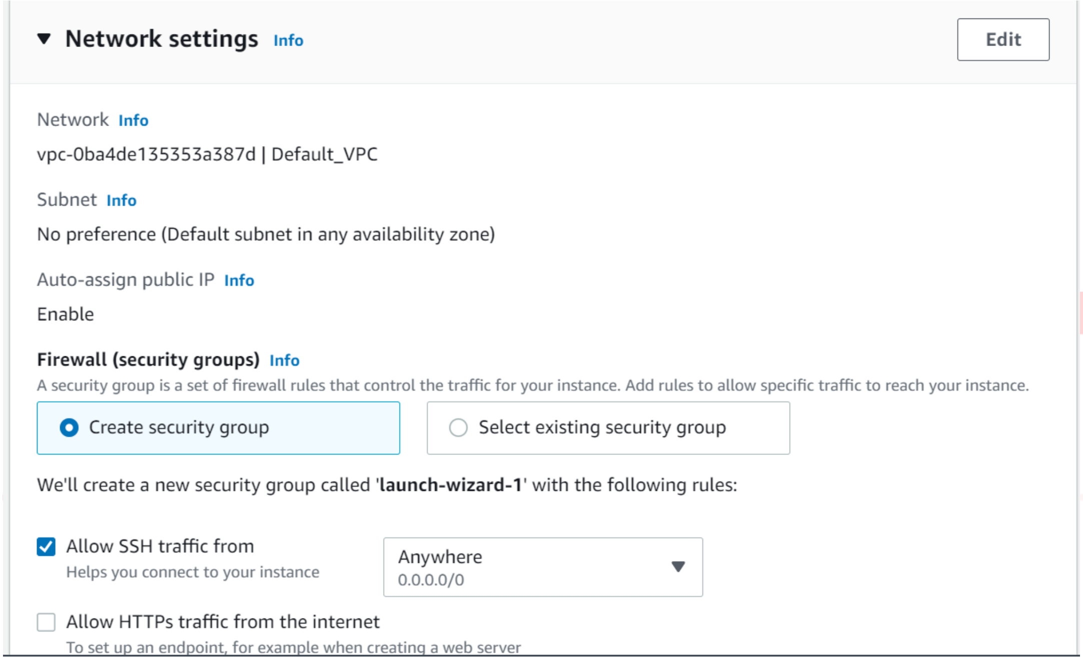
t2.micro Free tier eligible
Family: t2 1 vCPU 1 GiB Memory
On-Demand Linux pricing: 0.0124 USD per Hour
On-Demand Windows pricing: 0.017 USD per Hour



Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*: linux



Network settings Info

Network Info
vpc-0ba4de135353a387d | Default_VPC

Subnet Info
No preference (Default subnet in any availability zone)

Auto-assign public IP Info
Enable

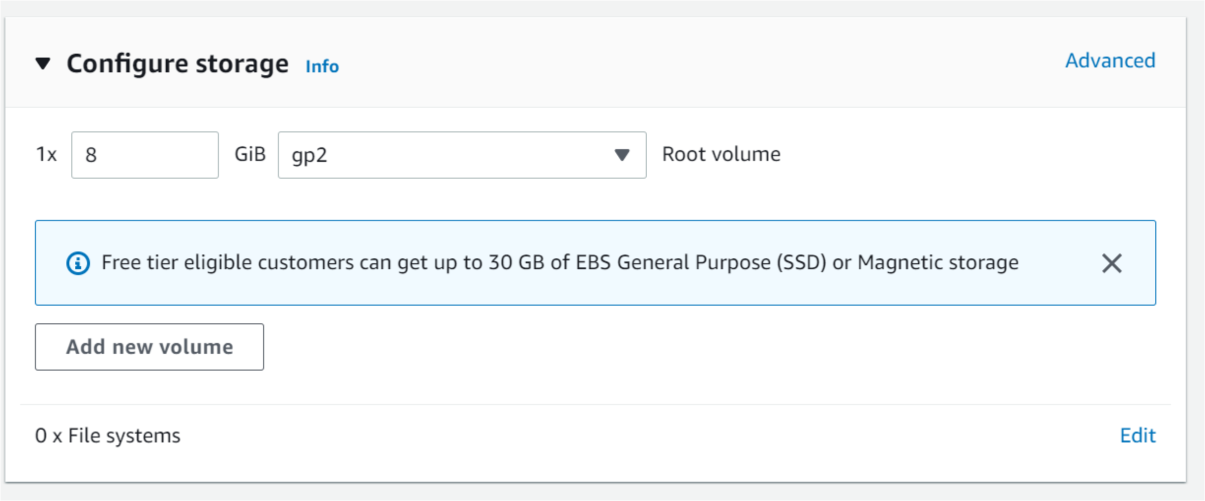
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

● Create security group ○ Select existing security group

We'll create a new security group called '**launch-wizard-1**' with the following rules:

☑ Allow SSH traffic from — Anywhere 0.0.0.0/0
Helps you connect to your instance

☐ Allow HTTPs traffic from the internet
To set up an endpoint, for example when creating a web server

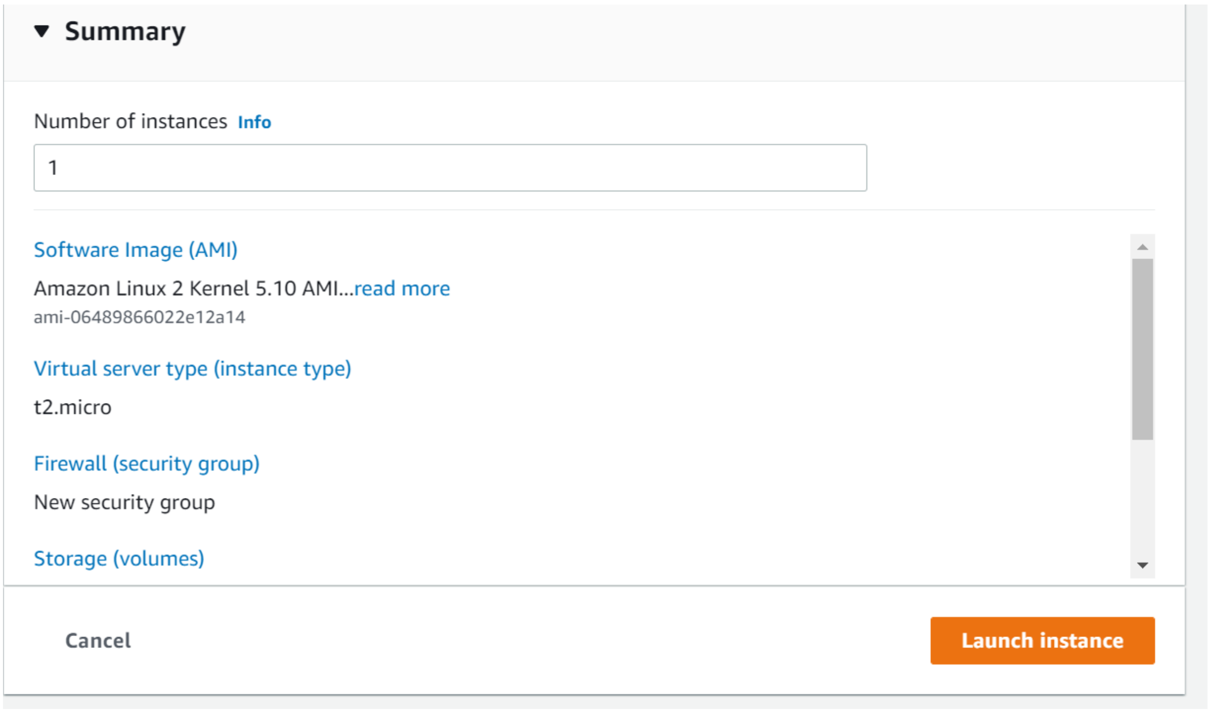


Configure storage Info

1x 8 GiB gp2 Root volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

0 x File systems



▼ Summary

Number of instances Info

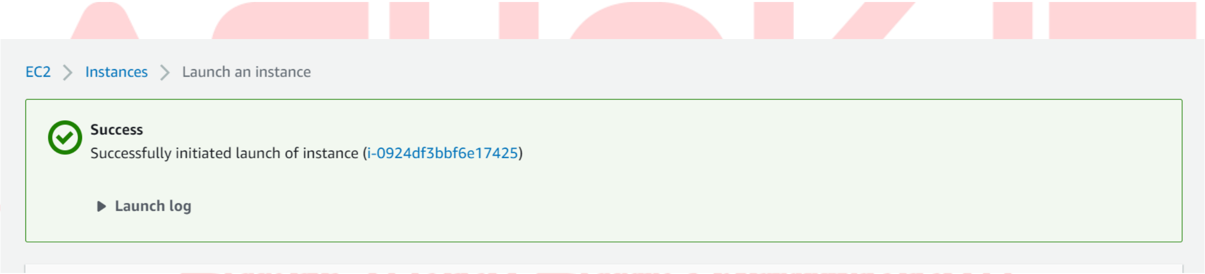
1

Software Image (AMI)
Amazon Linux 2 Kernel 5.10 AMI...read more
ami-06489866022e12a14

Virtual server type (instance type)
t2.micro

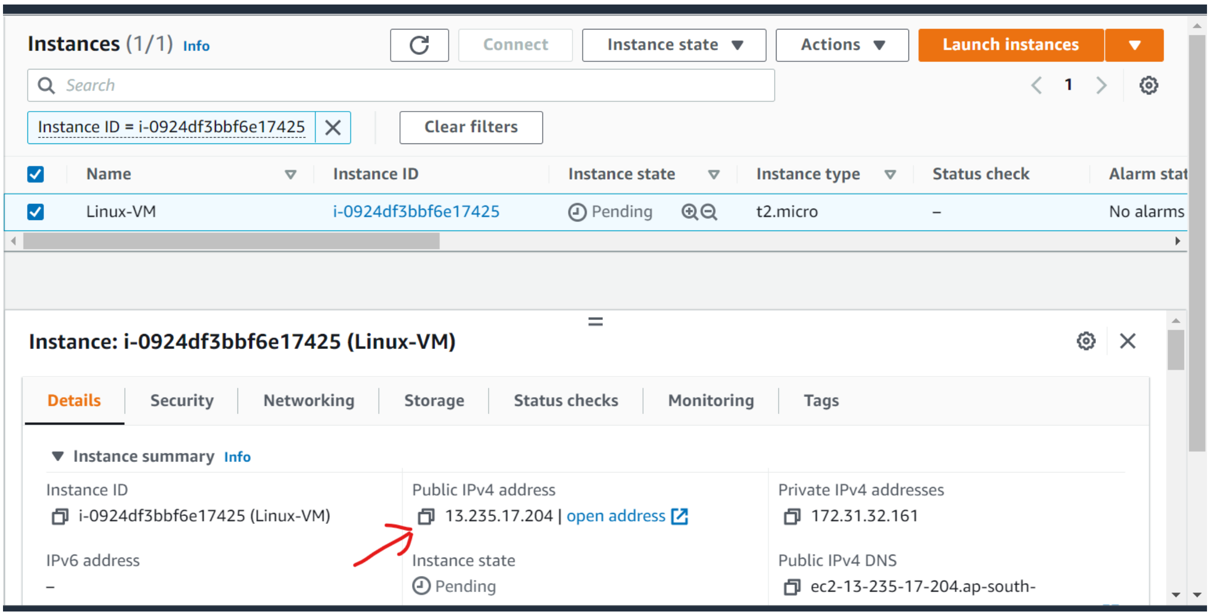
Firewall (security group)
New security group

Storage (volumes)



EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-0924df3bbf6e17425)



Instances (1/1) Info

Instance ID = i-0924df3bbf6e17425

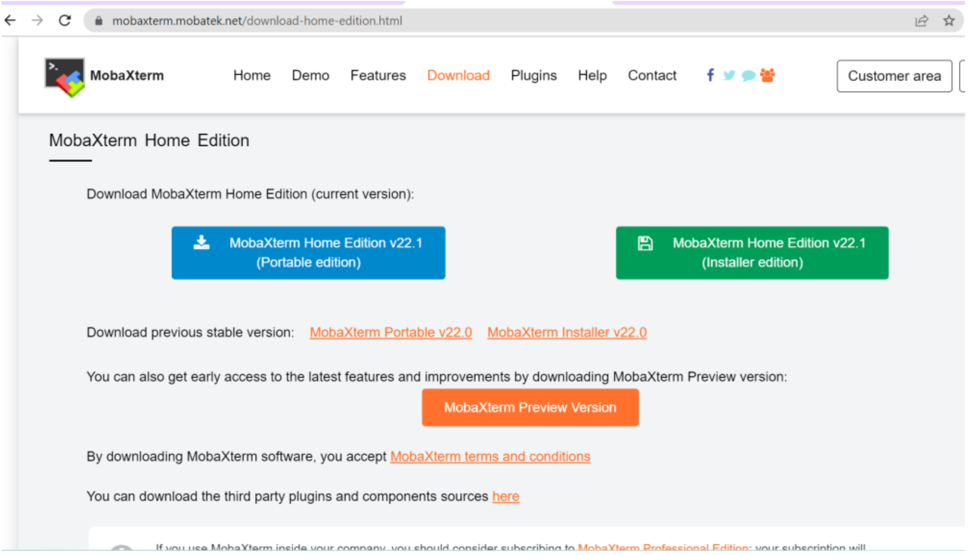
☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm stat
☑	Linux-VM	i-0924df3bbf6e17425	Pending	t2.micro	–	No alarms

Instance: i-0924df3bbf6e17425 (Linux-VM)

▼ Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0924df3bbf6e17425 (Linux-VM)	13.235.17.204 | open address	172.31.32.161
IPv6 address	Instance state	Public IPv4 DNS
–	Pending	ec2-13-235-17-204.ap-south-

Download Mobaxterm software to connect with Linux VM



mobaxterm.mobatek.net/download-home-edition.html

MobaXterm
Home
Demo
Features
Download
Plugins
Help
Contact
Customer area

MobaXterm Home Edition

Download MobaXterm Home Edition (current version):

MobaXterm Home Edition v22.1 (Portable edition)
MobaXterm Home Edition v22.1 (Installer edition)

Download previous stable version: MobaXterm Portable v22.0 MobaXterm Installer v22.0

You can also get early access to the latest features and improvements by downloading MobaXterm Preview version:

MobaXterm Preview Version

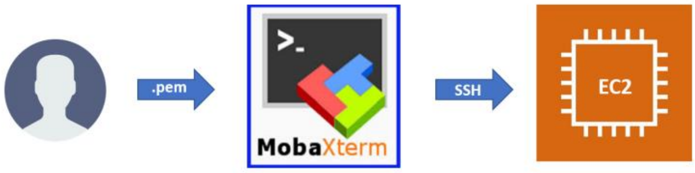
By downloading MobaXterm software, you accept MobaXterm terms and conditions

You can download the third party plugins and components sources here

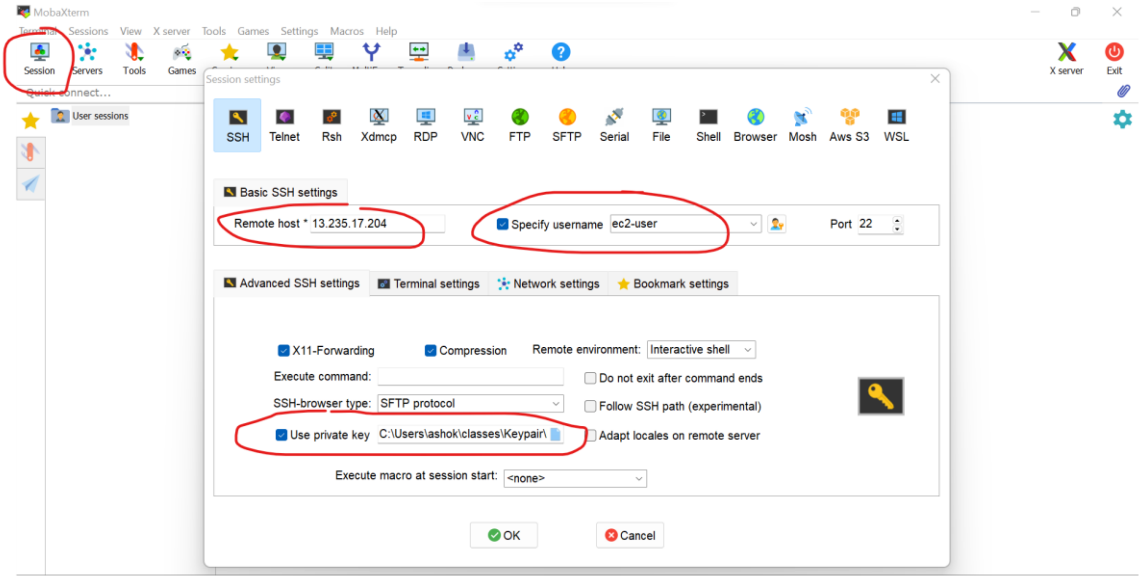
If you use MobaXterm inside your company, you should consider subscribing to MobaXterm Professional Edition: your subscription will

Connect with Linux VM using Public IP and pem file

Note: Once work is completed then stop your Linux VM in EC2 to avoid billing



➔ **Open Mobaxterm software and connect with Linux VM**



Remote host * 13.235.17.204

☑ Specify username ec2-user

Port 22

☑ X11-Forwarding ☑ Compression Remote environment: Interactive shell

Execute command:

□ Do not exit after command ends

SSH-browser type: SFTP protocol

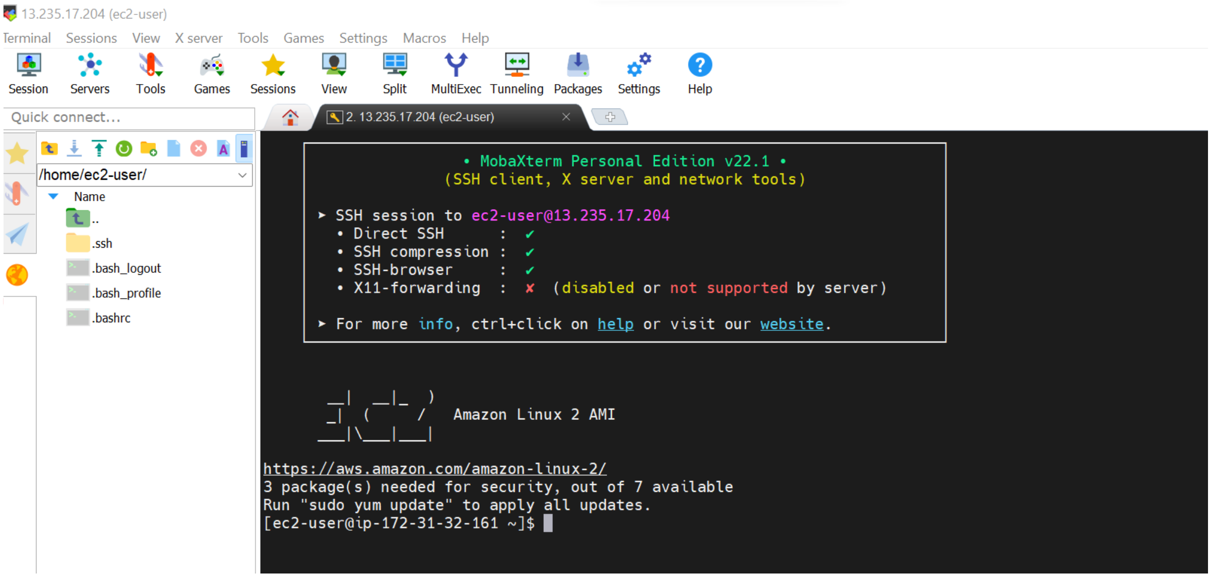
□ Follow SSH path (experimental)

☑ Use private key C:\Users\ashok\classes\Keypair\

□ Adapt locales on remote server

Execute macro at session start: <none>

➔ **Once we connect with EC2 instance using Mobaxterm we can see screen like below**



13.235.17.204 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. 13.235.17.204 (ec2-user)
/home/ec2-user/
Name
..
.ssh
.bash_logout
.bash_profile
.bashrc

```
                • MobaXterm Personal Edition v22.1 •
              (SSH client, X server and network tools)

 ➤ SSH session to ec2-user@13.235.17.204
   • Direct SSH      :  ✓
   • SSH compression :  ✓
   • SSH-browser     :  ✓
   • X11-forwarding  :  ✗  (disabled or not supported by server)

 ➤ For more info, ctrl+click on help or visit our website.


       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
3 package(s) needed for security, out of 7 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-32-161 ~]$
```

Linux Commands

$ whoami : It will display currently logged in username

$ pwd : present working directory

$ date : To display current date

$ cal : To display calendar

Linux File System

-> In Linux everything will be represented as a file

-> We have 3 types of files in Linux

1) Ordinary File / Normal File (it will start with -)

2) Directory File (it will start with d)

3) Link File (it will start with l)

-> The file which contains data is called as ordinary file

-> Directory file is equal to folder (it can contain files and folders)

-> The file which is having linking is called as Link File

-> touch : it is used to create empty file

```
$ touch f1.txt
$ touch f2.txt
$ touch f3.txt f4.txt
```

-> To display files, we will use 'ls' command

```
$ ls
```

-> To create a file with data we will use 'cat' command

```
$ cat > hello.txt
//write data
press CTRL + d (to save and exit)
$ cat hello.txt   (To display file data)
  $ cat >> hello.txt (To append data in the file)
//write data
press CTRL + d (to save and exit)
```

-> To create directory we will use 'mkdir' command

```
$ mkdir dirname
```

-> To remove the file we will use 'rm' command

```
$ rm filename
```

-> To remove empty directory we will use 'rmdir' command

```
$ rmdir dirname
```

-> 'ls' is used to list out all files & directories available in the given directory

Note: we can pass several options for 'ls' commands

- ls : It will display all files in alphabetical order (a to z)
- ls -r : It will display all files in reverse of alphabetical order (z to a)
- ls -l : It will display long listing of files
- ls -t : It will display all files based on last modified date and time. Most recent file will be display at top and old files will display at bottom.
- ls -rt : It will display all files based on reverse of last modified date and time. Old files will display at top and recent files will display bottom.
- ls -a : It will display all files including hidden files (hidden files will start with .)
- ls -li : It will display files with inode numbers.
- ls -lR : It will display all files and directories along with sub directories content

Note: -R represents recursive

Note: We can use several options for ls command at a time. When we are using multiple options order of the options is not important

```
$ ls -ltr

$ ls -tlr

$ ls -l -t -r

$ ls -trl
```

Note: All the above commands will give same output

-> To display content of given directory we can execute like below

```
$ ls -l <dirname>
```

-> To delete a file we will use 'rm' command

```
$ rm <filename>
```

-> To delete empty directory we will use 'rmdir' command

```
$ rmdir <dirname>
```

-> To delete non-empty directory we will use 'rm' command like below

```
$ rm -r dirname
```

-> To display file content we will use 'cat' command

```
$ cat filename
```

-> To display file content with line numbers we will use '-n' option

```
$ cat -n filename
```

-> To display multiple files content at a time, execute command like below

```
$ cat file1 file2 file3
```

-> Copy one file data into another file using 'cat' command

```
$ cat f1.txt > f8.txt
```

-> Copy more than one file data into another file

```
$ cat f1.txt f2.txt > f9.txt
```

-> 'tac' command is used to reverse file content

```
$ tac filename
```

-> 'rev' command is used to reverse each line content of the file

```
$ rev filename
```

-> head command is used to display file data from top (default 10 lines)

```
$ head filename
$ head -n 5 data.log  (first 5 lines data)
$ head -n 20 data.log (first 20 lines data)
```

-> tail command is used to display file data from bottom (default 10 lines)

```
$ tail filename (last 10 lines data)
$ tail -n 100 filename (last 100 lines data)
$ tail +25 filename (it will display data from 25th line to bottom)
```

Note: To see on-growing logs we can use '-f' option

```
$ tail -f data.log   (Live log message we can see)
```

-> It is used to count no.of lines, no.of words and no.of characters in the file

```
$ wc f1.txt
```

-> To copy the data from one file to another file

```
$ cp one.txt two.txt  ( or ) $ cat one.txt > two.txt
$ cp f1.txt f2.txt f3.txt  (invalid syntax)
```

-> We can't copy more than one file data using 'cp' command. To copy multiple files data we should go for 'cat' command

```
$ cat f1.txt f2.txt > f3.txt
```

-> To rename files we will use 'mv' command

```
$ mv f1.txt f1111.txt
$ mv  dirname dirnewname
```

Note: We can use 'mv' command for renaming and moving files

-> To compare files we can use below commands

```
$ cmp f1.txt f2.txt
$ diff f1.txt f2.txt
```

-> cmp command will display only first difference in given 2 two files

-> diff command will display all the differences in the content

Grep Command

-> 'grep' stands for global regular expression print.

-> 'grep' command will process the text line by line and prints any lines which matches given pattern.

Ex: I want to print all lines which contains 'NullPointerException'

```
$ grep -i 'NullPointerException'  *
```

Note: We can install grep using below command

```
$ sudo yum install grep
```

//search for the lines which contains given word in the given filename

```
$ grep 'word' filename
```

//search for the lines which are having exception keyword in server.log file

```
$ grep -i 'exception' server.log
```

//search for the given text in present directory and in sub-directories also

```
$ grep -R 'exception'
```

=> We can pass several options for 'grep' command

-c : This prints only the count of files that matches give pattern

-i : ignore case-sensitivity

-n : Display the matched lines and their line numbers

-l : Displays only file names that matches the pattern

-h : Displays matched lines without file names

Working with Text Editors in Linux

-> The default editor that comes with the LNIX operating system is called vi (visual editor).

-> Using vi editor, we can edit an existing file or we create a new file from scratch

-> We can also use this vi editor to just read a text file.

Modes of Operation in vi editor

There are three modes of operations in vi:

1) command mode

2) insert mode

3) escape mode

Command Mode:

- When vi starts up, it is in Command Mode. This mode is where vi interprets any characters we type as commands and thus does not display them in the window
- This mode allows us to move through a file, and to delete, copy, or paste a piece of text.
- To enter into Command Mode from any other mode, it requires pressing the [Esc] key. If we press [Esc] when we are already in Command Mode, then vi will beep or flash the screen.

Insert mode:

- This mode enables you to insert text into the file.
- Everything that's typed in this mode is interpreted as input and finally, it is put in the file.
- The vi always starts in command mode. To enter text, you must be in insert mode.
- To come in insert mode you simply type i.
- To get out of insert mode, press the Esc key, which will put you back into command mode.

Last Line Mode (Escape Mode):

- Line Mode is invoked by typing a colon [:], while vi is in Command Mode.
- The cursor will jump to the last line of the screen and vi will wait for a command.
- This mode enables you to perform tasks such as saving files, executing commands.

There are following way you can start using vi editor :

- vi filename: Creates a new file if it already not exist, otherwise opens existing file.
- vi -R filename : Opens an existing file in read only mode.
- view filename : Opens an existing file in read only mode.

```
$ vi f1.txt
```

=> After making changes if we want to save those changes then execute :wq

=> After making changes if we don't want to save those changes then execute: q!

SED command:

- SED command in LNIX stands for stream editor and it can perform lots of functions on file like searching, find and replace, insertion or deletion.
- Though most common use of SED command in LNIX is for substitution or for find and replace.
- By using SED you can edit files even without opening them, which is much quicker way to find and replace something in file, than first opening that file in VI Editor and then changing it.
- SED is a powerful text stream editor. Can do insertion, deletion, search and replace (substitution).
- SED command in Linux supports regular expression which allows it perform complex pattern matching.

Example:

```
$ cat > myfile.txt
        unix is great os. unix is opensource. unix is free os.
        learn operating system.
        unix linux which one you choose.
        unix is easy to learn.unix is a multiuser os. Learn unix. unix is a powerful.
```

Replacing or substituting string :

Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word “unix” with “linux” in the file

```
              $ sed 's/unix/linux/' myfile.txt
```

Deleting lines from a particular file

SED command can also be used for deleting lines from a particular file. SED command is used for performing deletion operation without even opening the file

To Delete a particular line say n in this example

```
        $ sed '5d' myfile.txt
```

-> To Delete a last line :

```
        $ sed '$d' myfile.txt
```

-> To Delete from nth to last line :

```
         $ sed '12,$d' myfile.txt
```

Note: By default, SED command changes will not store in file.

=> To make SED command changes to file permanently we will use '-i' option.

```
        $ sed -i 's/unix/linux/g' myfile.txt
```

Note: With above command 'unix' keyword will be replaced with 'linux' keyword in the file permanently.

Linux File permissions:

=> In Linux everything will be represented as a file

=> To protect our data, we need to manage file permissions

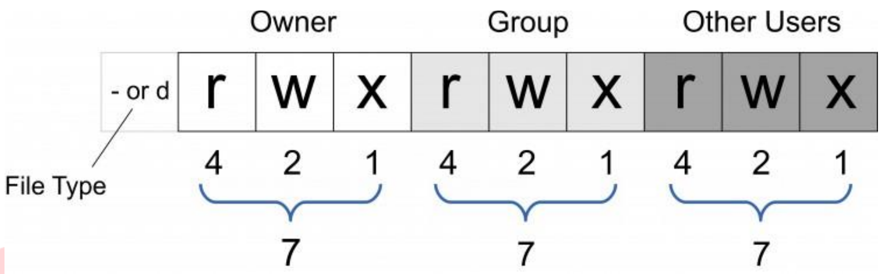
=> We have 3 types of permissions for the files in linux

r ===> read

w ===> write

x ===> execute

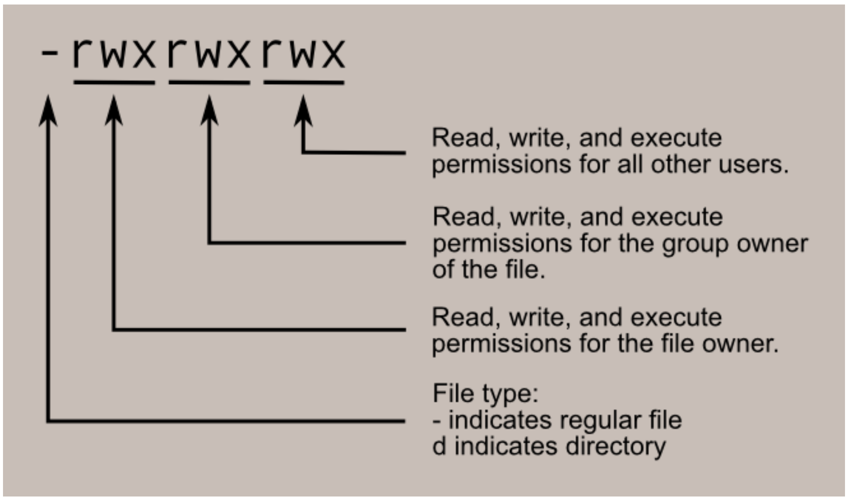
=> Linux file permissions are divided into 3 sections



=> File Permissions contains total 9 characters

- **first 3 characters will represent owner permissions (rwx)**
- **4,5,6 characters will represent group permissions(rwx)**
- **last 3 characters will represent others permissions (rwx)**

Note: If any permission is not available then it will represented with hypen (-)



Read, write, and execute permissions for all other users.

Read, write, and execute permissions for the group owner of the file.

Read, write, and execute permissions for the file owner.

File type:
- indicates regular file
d indicates directory

rw-r----x f3.txt	-w--w-rwx f4.txt	-rw---xr-- f5.txt
user : read + write group : read others : execute	user : write group : write others : read + write + execute	user : read + write group : execute others : read

=> To change file permissions, we will use 'chmod' command

# adding execute permission for user

$ chmod u+x f1.txt

# removing write permission for group

$ chmod g-w f2.txt

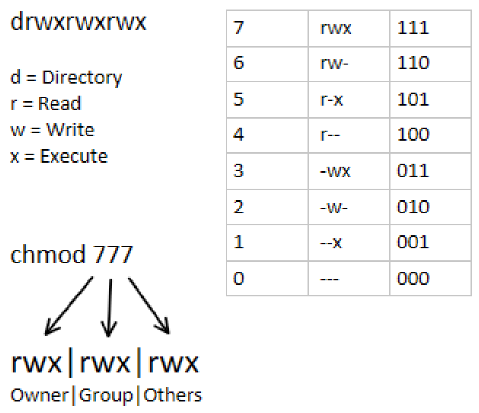
# remove write & execute for user

$ chmod u-wx f3.txt

# add write & execute for others

$ chmod o+wx f4.txt

We can use Symbolic Mode to modify permissions like the following:



drwxrwxrwx

d = Directory
r = Read
w = Write
x = Execute

7	rwx	111
6	rw-	110
5	r-x	101
4	r--	100
3	-wx	011
2	-w-	010
1	--x	001
0	---	000

For example, let's give every permission for all with:

```
$ chmod 777 section.txt
```

Then the permissions will be: -rwxrwxrwx.

Let's remove the execute from the group and the write from other by:

```
$ chmod 765 section.txt
```

Then the permission will be : -rwxrw-r-x

Working with User Accounts

-> Linux is multi user based operating systems

-> Within one Linux machine we can create multiple user accounts

-> Multiple users can access single linux machine and can perform multi-tasking

Note: In every Linux machine by default one 'root' account will be available (super account)

-> When we launch EC2 instance with Amazon Linux OS, we got by default 'ec2-user' user account

# To get user account info we will use below command

```
$ id <username>
```

Note: For every user one home directory will be available in Linux

ec2-user	/home/ec2-user
ashok	/home/ashok
john	/home/john
smith	/home/smith
root	/root

# Switch to root user account

$ sudo su   (it will switch to root account)

$ sudo su - (it will switch to root account & will point to root user home directory)

$ exit (to exit from user account)

Working with Users and Groups

# Switch to root user

```
$ sudo su –
```

# Add new user

```
$ sudo useradd <username>
```

# Set password for user

```
$ sudo passwd <username>
```

# Display users created

```
$ cat /etc/passwd
```

# Switch User account

```
$ su <username>
```

# Delete user

```
$ sudo userdel <username>
```

# Delete user along with user home directory

```
$ sudo userdel <username> --remove
```

# Display all groups available

```
$ cat /etc/group
```

Note: When we create new user then user + user group will be created.

# Create new Group

```
$  sudo groupadd <groupName>
```

# Adding user to group

```
$ sudo usermod -aG <group-name>  <user-name>
```

# Remove user from the group

```
$ sudo gpasswd -d <username> <group-name>
```

# Display users belongs to group

```
$ sudo lid -g <group-name>
```

# display user belongs to how many groups

```
$ id <username>
```

# Delete Group

```
$ sudo groupdel <group-name>
```

# How to change username

```
$ usermod -l <login-name> <old-name>
```

(or)

```
$ usermod  -u UID <username>
```

# How to change user pwd

```
$ passwd <username>
```

# Change group name

```
$ groupmod -n <new-name> <old-name>
```

Chown command:

-> The chown command changes user ownership of a file & directory in Linux

-> Every file is associated with an owning user or group

-> We can check the ownership of a file using 'ls -l'

# Changing owner of a file

```
$ chown NewUser FILE
```

Note: we can change owner using userid also

```
$ chown 1001 filename
```

# changing group of a file

```
$ chown :NewGroup FILE
```

# We can change group using group id also

```
$ chown :1003 sample
```

# Change Owner and the Group

```
$ chown NewUser:NewGroup FILE
```

Working with 'find' and 'locate' commands

-> find and locate commands are used to search files in Linux machines

locate command:

The locate Command find will search for data in local db

```
$ sudo yum install mlocate

$ locate apache

$ locate -c apache

$ locate -c *.txt

$ locate -S (to see locate database)
```

Note: when we create new files it will take some time to update those files in mlocate db

find command:

-> find command will search for the files in entire linux file system.

-> find command providing advanced searching technique

-> Using find command, we can search for the files based on name and type also.

-> Find Files Under Home Directory

```
$ find /home -name f1.txt
```

-> Find Files With 777 Permissions

```
$ find . -type f -perm 0777 -print
```

-> Find all Empty Files inside home directory

```
$ find /home -type f -empty
```

-> Find all Empty Directories inside home directory

```
$ find /home -type d -empty
```

# Display 30 days old files in linux

```
$ sudo find . -mtime 30 -print
```

# Delete 30 days old files in linux inside home directory

```
$ sudo find /home -mtime 30 -delete
```

# Delete 1 hour old files in linux

```
$ sudo find /home -mmin +60 -delete
```

Note: As find command is scanning entire file system, it will take more time to give search results.

-> 'man' command is like a help command. It is used to understand command syntax and options.

```
$ man cat
```

-> To see ip address we will use 'ifconfig' command

```
$ ifconfig
```

-> 'ping' command is used to check connectivity

```
$ ping <ip>
```

-> 'curl' command is used to get response from the server

```
$ curl <url>
```

-> 'wget' command is used to download resources from internet

```
$ wget <url>
```

Working with Zip files in Linux

# check zip is installed or not

```
$ zip --version
```

Note: If not installed then install it

# Create Files for Achieve as zip

```
$ touch file{1..5}.txt
```

# Create zip file

```
$ zip files *.txt
```

# List Zip File Content

```
$ zip -sf files.zip
```

# Add directory to zip file

```
$ zip -r files <directory>
```

# Delete particular file from zip

```
$ zip -d files f5.txt
```

# Create zip file with encryption

```
$ zip -e <zipfilename> *.txt
```

# unzip archived file

```
$ unzip <zipfilename>
```

Memory Related Commands

free: The free command displays information about the amount of physical and swap memory that is available on the system. It also shows how much memory is being used by processes and the kernel.

```
$ free
```

vmstat: The vmstat command displays information about virtual memory usage, including the amount of memory that is being used by processes, the kernel, and the file system cache.

```
$ vmstat
```

top: The top command displays a list of all processes running on the system, along with information about their memory usage.

```
$ top
```

htop: The htop command is a graphical version of the top command. It provides a more detailed view of memory usage and other system resources.

```
$ htop
```

Package Managers in Linux

In Linux, package managers are tools that help users install, update, and manage software packages on their system.

They handle dependency resolution, ensuring that all required libraries and components are installed correctly. Here are some commonly used package managers in Linux:

Advanced Package Tool (APT):

- APT is the package manager used in Debian-based distributions such as Ubuntu.
- Commands: apt-get, apt, apt-cache

dpkg:

- dpkg is the underlying package management system used by APT.
- Commands: dpkg, dpkg-query, dpkg-deb

YUM:

- YUM (Yellowdog Updater, Modified) is the default package manager for Red Hat-based distributions such as CentOS and Fedora.
- Commands: yum

DNF:

- DNF (Dandified YUM) is the next-generation package manager, replacing YUM in recent versions of Fedora and CentOS 8+.
- Commands: dnf

How to install Software in Linux Machine:

-> In Linux we have package manager to install a software

Note: Amazon Linux / CentOS / Red Hat: yum is the package manager

Note: Ubuntu: apt-get is the package manager

# Update existing packages in Linux

```
$ sudo yum update -y
```

# Install git client

```
$ sudo yum install git
```

# check git version

```
$ git --version
```

# Install maven

```
$ sudo yum install maven
```

# check maven version

```
$ mvn –version
```

# install java software

```
$ sudo yum install java
```

# install Java 1.8 version

```
$ sudo yum install java-1.8.0-openjdk
```

Static Website Hosting in Linux VM (Amazon Linux):

1) Connect to EC2 instance

2) Execute below commands to install Apache server (httpd)

```
$ sudo yum update -y

$ sudo yum install httpd

$ service httpd start
```

Note: Check accessing Apache server from browser using EC2 VM public IP

-> If server is not accessible then, enable HTTP port : 80 in Security Group of our EC-2 VM

-> After adding security group try accessing EC2 instance using IP

Ex: http://52.66.101.3/

-> We can modify html page content

```
$ cd /var/www/html

$ sudo vi index.html

<h1> Welcome to Ashok IT - Website</h1>
```

-> Save the content and close the file (press Esc then :wq)

-> Now access public IP in browser

Shell Scripting

What is shell?

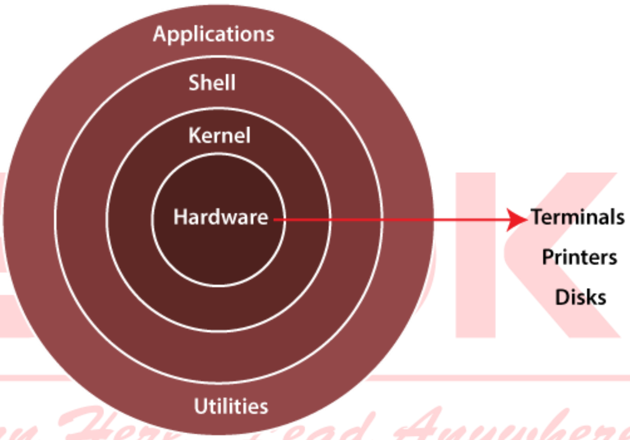
-> Shell is responsible for reading commands given by user

-> Shell will verify command and will give instructions to kernel to process that command

-> If command is invalid shell will give error

-> Kernel will execute our command with System Hard Components

-> Shell acts as mediator between User and Kernel



What is Scripting?

-> Scripting means set of commands mentioned in a file for execution

-> Scripting is used to automate our routine work

-> For example i want to execute below commands every day as a linux user

```
$ date
$ cal
$ whoami
$ pwd
$ ls -l
```

-> Instead of executing all these commands manually we can keep them in a file and we can execute that file.

-> The file which contains set of commands for execution is called as 'Script file'

What is Shell Scripting?

-> Shell Scripting is used to execute set of commands using a script file

-> When we execute script file then shell will read those commands and will verify commands syntax

-> Shell will give instructions to 'Kernel'

-> Kernel will give instructions to hardware components to perform actual operation

Types of Shells

-> There are several shells available in linux OS

1) Bourne Shell
2) Bash Shell
3) Korn Shell
4) CShell
5) TShell
6) ZShell

# Display all the shells of our Linux machines

```
$ cat /etc/shells
```

# Display the default shell of our Linux machine

```
$ echo $SHELL
```

Note: The most commonly used shell in Linux in is 'BASH SHELL'.

Note: Shell Script files will have .sh extension

Working with First Shell Script Program

# Create a script file

```
$ vi task.sh
whoami
pwd
date
```

-> Save the file and close it (ESC + :wq)

# Run the shell script (Option-1)

```
$ sh task.sh
```

Note: If we get permission denied then we should provide 'execute' permission using 'chmod' command

# Run the shell script (Option-2)

```
$ ./task
```

What is sha-bang in shell script?

-> Sha-bang is used to specify which shell should be used to process our script

Syntax :

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

Shell Script - 2

```
#! /bin/bash

echo "Welcome to Scripting"

echo "Scripting is used to automate regular work"

echo "Scripting requires lot of practise"
```

Shell Script - 3

```
#! /bin/bash

echo "Enter your name:"

read name

echo "Good Morning $name"
```

Shell Script - 4

```
#! /bin/bash

a=10

b=20

c=$(($a + $b))
```

```
echo "Sum of $a and $b is =  $c"
```

Shell Script - 5

```
#! /bin/bash

echo "Enter First Number"

read a

echo "Enter Second Number"

read b

c=$(($a + $b))

echo "Sum of $a and $b is =  $c"

END
```

Variables

-> Variables are place-holders to store the value

-> Variables are key-value pairs

-> In Shell Scripting there is no concept of Data Type.

-> Every value will be treated as text/string

Ex:

```
        name=ashok
        age=30
email=ashokitschool@gmail.com
phno=1234
```

-> Variables are divided into 2 types

1) Environment Variables or System variables

2) User Defined Variables

-> The variables which are already defined and using by our system are called as Environment/System variables

Ex:

```
$ echo $USER

$ echo $SHELL
```

-> Based on our requirement we can define our own variables those are called as user defined variables

Ex:

```
name=ashok

age=30

$echo $name $ age
```

Variable Rules

-> We should not use special symbols like -, @, # etc....

-> Variable name should not start with digit

Note: It is recommended to use uppercase characters for variable name

-> we can use 'readonly' for variable so that variable value modification will not be allowed

Command Line Arguments

-> The arguments which will pass to script file at the time of execution

-> Cmd args are used to supply the values dynamically to the script file

Ex:

```
$ sh demo.sh ashokit 30
```

-> We can access cmd args in script file like below

$# - no.of args

$0 - script file name

$1 - First Cmd Arg

$2 - Second Cmd Arg

$3 - Third Cmd Arg

$* - All Cmd Args

-> To comment any single line we can use '#'

```
echo 'hi'

#echo 'hello'
```

-> We can comment multiple lines also in script file like below

```
<<COMMENT

        ....................

COMMENT
```

-> We can hold script execution for some time using 'sleep' command

```
#! /bin/bash

echo $#

echo $0

echo $1

sleep 30s

echo $2

#echo $*
```

Conditional Statements

-> Conditional statements are used to execute commands based on condition

Syntax:

```
if [ conition ]

then

        stmts

else

      stmts

fi
```

-> If given condition satisfied then if statements will be executed otherwise else statements will be executed

```
if [ condition ]

then

        stmts

elif [ condition ]
```

```
then
          stmts
else
          stmts
fi
```

Ex:

```
#!/bin/bash
echo "Enter Your Favorite Color"
read COLOR
if [ $COLOR == 'red' ]
then
      echo "Your are cheerful"
elif [ $COLOR == 'blue' ]
then
      echo "You are joyful"
else
       echo "You are lucky"
fi
```

Working with loops

-> Loops are used to execute stmts multiple times

-> We can use 2 types of loops

1) Range based loop (ex: for loop)

2) Conditional based loop (ex: while loop)

For Loop Example

```
#! /bin/bash

for ((i=1; i<=10; i++))

do

echo "$i"

done
```

While loop Example

```
#! /bin/bash

i=10

while [ $i -ge 0 ]

do

echo "$i"

let i--;

done
```

Infinite Loop

```
#! /bin/bash

while true

do

echo "This is my loop stmt"

done
```

Note: To stop infinite loop we will use 'ctrl + c'

Functions

-> The big task can be divided into smaller tasks using functions

-> Function is used to perform an action / task

-> Using functions we can divide our tasks logically

-> Functions are re-usable

Syntax:

```
function functionaName( ) {
    // commands to execute
}
```

Writing welcome function

```
#! /bin/bash
function welcome(){
  echo "Welcome to functions...";
  echo "I am learning Shell Scripting";
  echo "Shell Scripting is used to automate our regular work";
}
# call the function
welcome
```

Function with Parameters

```
#! /bin/bash
function welcome ( ) {
    echo "$1";
}
# call function
welcome Linux
welcome AWS
welcome DevOps
```

Requirement : Write a function which will take filename as input and print content of that file.

```
#! /bin/bash/

function readFileData(){

  echo "Enter File Name"

  read filename

  cat $filename

}

readFileData
```

Requirement : Check File Exists or not

```
#!/bin/bash

filename=$1

if [ -f "$filename" ]; then

echo "File exists"

else

echo "File does not exist hence creating file"

touch $filename

echo "File Created"

fi
```

Requirement : Check Directory Exists or not

```
#! /bin/bash

dirname=$1

if [ -d "$dirname" ]; then

echo "Directory exist"

else

echo "Directory not exist hence creating directory"

mkdir $dirname

fi
```

Shell Scripting Practice Programs

1) Take a number from user and check given number is prime number or not.

2) Take a number from user and check given number is even or odd number.

3) Take a year from user and check given year is leap year or not.

4) Take a number from user and print table of that number like below

2 * 1 = 2

2 * 2 = 4

2 * 3 = 6

....

2 * 10 = 20;

5) Write a shell script to read user to address, mail subject and mail body then send email with given details.

What is CRON?

Cron is the most useful utility in a Linux or UNIX-like operating system that allows running commands or scripts on a given schedule without any user intervention.

It is mostly used for automating recurring jobs like running scheduled backups, cleaning temporary files, system maintenance, and various other recurring jobs. It is similar to the Task Scheduler in Windows OS.

What is Crond?

Crond is the daemon in the Linux system that runs in the background and checks every minute to see if there is any job scheduled at that time. If there is, it performs that job, else it remains inactive.

Cron Job Syntax

The syntax for cron job is as follows:

```
        * * * * * command/script
```

From the left:

- The first * corresponds to Minutes (0-59)
- The second * corresponds to Hours (0-23)
- The third * corresponds to Day of the month (1-31)
- The fourth * corresponds to the Month of year (1-12)
- The fifth * corresponds to Day of the week (0-6, Sunday to Saturday)

To specify multiple values in a field, use the following operator symbols:

- Asterisk (*): To specify all possible values for a field
- Dash (-): To specify a range of values
- The comma (,): To specify a list of values
- Separator (/): To specify a step value

Editing Crontab File

Crontab is a file that contains the scheduled jobs in a specific syntax. There are two types of crontab files; one for system-specific cron jobs and the other for user-specific cron jobs.

System cron jobs

The system-wide cron jobs are located in the /etc/crontab file and /etc/cron.d directory, and they are run through /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly and /etc/cron.monthly. Only a system administrator can access these files.

A system administrator can define a cron job using the following command:

```
        $ nano /etc/crontab
```

Here is the syntax of the job in the /etc/crontab file:

```
# min hr dayofmonth month dayofweek username command
```

```
        * * * * * user1 ifconfig
```

User-specific cron jobs

The user-specific cron jobs are located in the /var/spool/cron/crontabs directory. Although you can edit these jobs manually, it is recommended to edit these jobs using the crontab -e command.

A standard user can define a cron job using the following command:

```
$ crontab -e
```

For instance, if you are logged in as a "test" user, running the crontab -e command will edit the crontab file for the "test" user. Similarly, if you are logged in as a root user, the crontab -e command will edit the crontab file for the root user.

Crontab Commands

The crontab command is used to edit, list, and remove the cron jobs:

- crontab -e : To edit current user's crontab file
- crontab -l : To display contents of the crontab file
- crontab -u [username] : To edit any other user's crontab file
- crontab -r : To remove the crontab file of the current user's
- crontab -i : To display a prompt before removing the current user's crontab file

Run a cron job every 15 minutes

To schedule a cron job to run every 15 minutes, add the below line in the crontab file:

```
*/15 * * * * command/script
```

Run a cron job at 5 am every day

To schedule a cron job to run at 5 am every day, add the below line in the crontab file:

```
0 5 * * * command/script
```

Run a cron job at 5 pm every day

To schedule a cron job to run at 5 pm every day, add the below line in the crontab file:

```
0 17 * * * command/script
```

Run a cron job at 9 am on the first day of every month

To schedule a cron job to run at 9 am on the first day of every month, add the below line in the crontab file:

```
0 9 1 * * command/script
```

=== Learn Here.. Lead Anywhere..!! ===