**Linux commands for accessing the virtual machine present in server using linux commands**

1. $ whoami :- command used to check the current user logged in to the server
2. $ date :- used to current server date and time
3. $ cal :- used to get the calender for that particular month
4. $ cal 2025 :- used to get the calender for that particular year
5. $ pwd :- command used to get the present working directory
6. $ mkdir <foldername> :- used to create a folder/directory
7. $ ls :- listing the content of that particular directory sorted in alphabetical order
8. $ ls -r :- listing the content of that particular directory sorted in descending alphabetical order or reverse sorting
9. $ ls –l :- longlisting the folder contents based on current date and time
10. $ ls –lr :- longlisting the file names in reverse order
11. $ clear :- command is used for clearing the screen
12. $ rmdir <foldername> :- is used for removing empty directory
13. $ ls –la :- used to get the hidden files in a particular folder
14. $ cd /home/ec2-user/a :- command is used for navigating to another directory folder
15. $ cd .. / cd ~ :- is used for going one folder back
16. $ touch <filename.txt> :- command used for creating a text file
17. $ rm –rf <folder name>:- recursively forcibly remove the folder even when the content is there inside it
18. $ exit :- command used for logout

---------------------------------------------------------------------------------------------------------------------

1. $ mv <existing filename.txt> <expected filename.txt> command used for renaming the file name
2. $ mv f0.text(current file) /home/ec2-user/a (location and the folder name it has to be moved)
3. $ mv present location(file name) to location where it has to moved
4. $ cat > f4.txt enter ctr + C :- creating a file
5. $ cat >> f1.txt

Hello world

Hi

Used for writing contents into the file

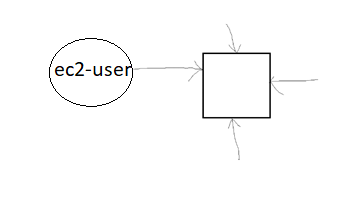
1. $ cat f1.txt :- command used for reading the contents of the file
2. $ cat –n f1.txt :- used for reading the contents of the file including the line numbers
3. $ cat >> f6.txt :- If a file is not present it is used for creating a file as well
4. $ touch f6.txt :- just updates the timestamp of the file it doesn’t alter the content of the file
5. $ tac f1.txt :- is used to read the file in reverse order
6. $ cp f1.txt(source file) f2.txt(destination file) :- used to copy the content from one file to another file
7. $ cat f1.txt f2.txt > f3.txt :- used to copy the contents of file f1, f2 to file f3.txt
8. $ head f1.txt :- used to read the first 10 lines of f1.txt file
9. $ head –n 5 f3.txt :- to read the first 5 lines of f3.txt file
10. $ grep “He” f1.txt :- global regular expression is used to find specific pattern or a word in a particular file
11. $ grep –i “He” f1.txt :- performs search operation based on the case insensitivity comparison
12. $ grep –n –i “He” f1.txt :- here -n is for line number and -i for ignoring the case sensitivity order of i & n doesn’t matter
13. $ grep -v -i -n “He” f1.txt :- here to search other content apart from “He” is used
14. $ grep -i ‘He’ \* :- this command is used for searching from all the files in that particular folder
15. $ grep -i -v ‘He’ \* :- apart from ‘He’ all other data is found and fetched on the screen
16. $ grep –i –v –n ‘He’ \* :- here including the line number if the content has to fetched then this is mainly used
17. $ tail f3.txt | grep –i ‘He’ :- to find the content in the last 10 lines of the file
18. $ tail –n 3 f3.txt | grep –i ‘He’ :- to find the content in the last 3 lines here we are using pipelines for storing multiple commands
19. $ cat f1.txt | nl :- command is used for reading contents of the file in order
20. $ tac f1.txt | nl :- command is used for reading contents of the file in reverse order
21. $ head –n 5 f3.txt | grep -i ‘He’ :- to find the content in first 5 lines of f3.txt
22. $ wc f3.txt :-used for getting the line, word & characters count in that particular file
23. $ wc –l f3.txt :- used to get only the number of lines present in the file
24. $ wc –w f3.txt :- used to get the count of words in the file
25. $ wc –m f3.txt :- gives us the number of characters present in that particular file
26. Use the cp command to copy the contents from file1.txt into the file2.txt
27. Use head to display the first 15 lines of logfile.log
28. Use grep to search all the occurrences of the word “error” in logfile.log
29. Use tail to show the last 20 lines of access.log
30. Use WC to count the number of words in the sample.txt

---------------------------------------------------------------------------------------------------------------------

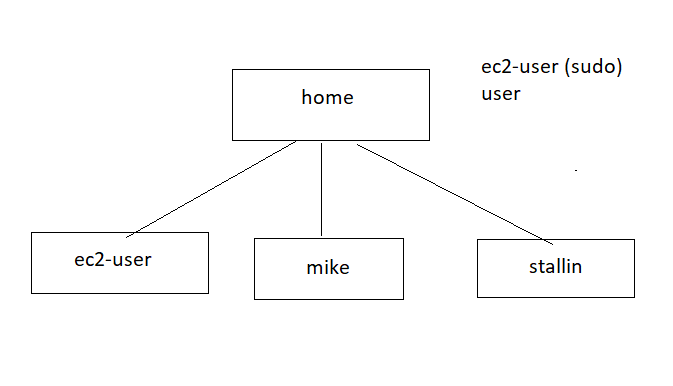
1. $ vi f3.txt :- Is a Visual Editor for Linux Os just like Notepad editor for Windows Os
2. $ vi f3.txt :- after entering the text content just type I for insert mode after entering the text press esc key and for saving and quit use :wq command
3. :w :- for just saving
4. :q :- for quit option
5. :q! :- for quit without saving
6. $ sed :- here sed stands from Stream Editor just like visual editor here stream editor can do text processing (substitution, deletion, printing and insertion). Performing operations without opening them
7. $ sed ‘s\Hello\World\’ f1.txt :- this command is used for replacing/substituting all the first occurences of Hello with World
8. $ sed ‘s\Hello\World\2’ f1.txt :- is used for replacing all the second instances of Hello in a line in the f1.txt file
9. $ sed ‘s\world\hello\g’ f1.txt :- is used to replace or substitute all the occurrences of world to hello
10. $ sed -i 's/world/kushal/g' f1.txt :- is used for permanently changing the content
11. $ sed -i '1d' f1.txt :- command is used for deleting the first line of the text file
12. $ sed -i ‘$d’ f1.txt :- command is used for deleting the last line
13. $ sed -i ‘1,5d’ f1.txt :- command is used for deleting range of lines
14. $ sed -i ‘1, $d’ f1.txt :- command for deleting all the lines in a text files

---------------------------------------------------------------------------------------------------------------------

1. Difference b/w vi editor & stream editor is for vi editor we need to open the file and make changes whereas in stream editor without opening we can edit, substitute & delete the changes
2. $ sed –n ‘1,3p’ f1.txt :- command is used for printing the contents of the file(-n reading the file)
3. $ sed -i '2i\hello world' f1.txt :- command is used for adding content in the middle of the file
4. $ sed -i '$a\last line' f1.txt :- command is used for adding content at the last line



* If this is a server multiple users are getting connected to the server as EC2-User with admin (sudo) In linux there is no admin it’s called sudo user super user do(sudo).
* If a Linux server is created in AWS using EC2 it’s going to create a EC2-User sudo account by default
* By using aws we can create multiple users for ec2 by default there is linux vm created by user which will have a home directory and user will be ec2-user \home\ec2-user
* If another user is created called mike all his work will be stored in \home\mike directory under home folder
* If a third User is created Stallin under \home\stallin directory get’s created all these users are independent of each other



1. $ sudo useradd mike :- here admin is sudo user ec2-user responsible for adding user
2. $ sudo passwd mike :- here for creating password of user mike here after that need to give new password and retype the new password
3. $ grep mike /etc/passwd :- here grep command is used for searching a particular command all users can be seen in etc directory and under sub directory passwd under home all these folders are created
4. $ sudo useradd stallin
5. $ sudo passwd stallin (add password and confirm password)
6. $ grep stallin /etc/passwd :- grep command is for searching the stallin user
7. $ su mike :- here su is switch user and we must give the password for mike
8. $ pwd :- after logging in as mike it is still in { /home/ec2-user }
9. $ ls :- it shows cannot open directory ‘.’ : permission denied
10. $ cd ~ :- here it returns to the mike directory
11. $ ls
12. $ touch f1.txt
13. $ ls = touch f1.txt
14. $ exit :- back to the root account to sudo user /home/ec2-user
15. $ cd ..
16. $ ls = ec2-user mike stallin
17. $ su stallin :- for switching to stallin after enter hit password
18. $ sudo userdel mike --remove :- Is used for deleting the account in linux os
19. $ cd ..
20. $ ls = ec2-user stallin
21. $ cd ~
22. $ sudo useradd alice
23. $ sudo passwd alice :- here it asks for password and retype the same pwd for confirmation post that If want to update the password
24. $ sudo passwd alice :- then enter the new pwd for updating the password
25. Creating user groups putting person while creating a group manage the users by putting a permission on that user groups automatically users added to that user groups has all the permissions.

**[Commands for user groups]**

1. $ cat /etc/group :- all user groups are displayed
2. $ sudo groupadd <groupname> :- command for adding the group name or creating a user group

---------------------------------------------------------------------------------------------------------------------

1. $ sudo useradd stallin
2. $ sudo passwd stallin
3. $ su stallin

The sudoers file controls which user can execute communicate commands as superusers

1. $ sudo visudo //this is for opening up the sudoers file

Scroll down here I want to make stallin as sudo user (admin user)

stallin ALL = (ALL:ALL) ALL //this command is used for giving all the access to the stallin to have the admin access here both ec2-user, stallin users has sudo user access details present in sudoers file

1. For exit(quit) : - ctrl + x + y + enter
2. Here permission for stallin has been set to a super user

Enable password Authentication

By Default all the users password authentication is not there

1. $ sudo vi /etc/ssh/sshd\_config
2. Scroll down search for (PasswordAuthentication yes)
3. Esc and then :wq
4. After modification of sshd\_config after setting PasswordAuthentication to yes restart the SSH Service
5. $ exit
6. Connect from ec2 to current device through ec2 commands copy paste those 2 lines in EC2 instance
7. $ sudo systemctl restart sshd //make sure that after the modifications everything has to be restarted
8. $ ssh mike@Ip4adress (copy it from ec2 dasboard after connecting to server).
9. It asks for password If these are given to others anyone can access this using these credentials
10. $ pwd
11. $ ssh stallin@Ipaddress
12. $ clear
13. $ id mike //each user will have a unique identifier associated with him
14. $ id ec2-user

---------------------------------------------------------------------------------------------------------------------

1. $ sudo groupadd developers //here if & only if user is a sudo user he can create a group

Add Users to the group “developers”

1. $ sudo usermod –aG developers alice //here after creating a group called developers adding the user alice to developers group
2. $ sudo useradd alex
3. $ sudo passwd alex
4. $ clear
5. $ sudo usermod –aG developers alex :- append –a for Group G is the exact meaning

Comma/nd for displaying the number of users in the user groups

1. $ sudo lid –g developers <groupname> :- here the groupname is developers

Will get the list of all the users present in the user group developers

1. $ sudo lid –g developers :- command for checking the list of users in the user group

//mike (uid = 1001)

//stallin (uid = 1002)

Command for removing/deleting a particular user from that group

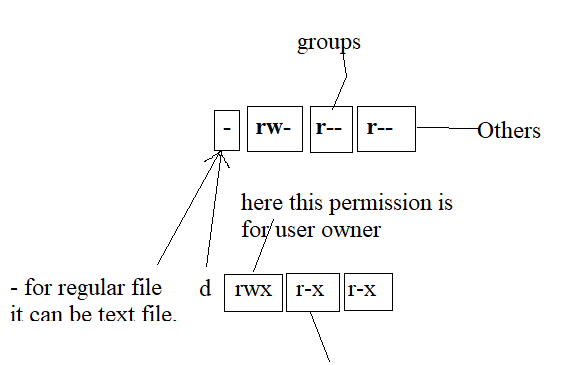
1. $ sudo gpasswd –d mike developer :- -d for remove or delete the user mike from the user group developer
2. $ sudo lid –g developers
3. $ id mike :- command is used for checking which all the groups mike belong to

Delete the user group

1. $ sudo groupdel <groupname> :- is used for deleting the group
2. $ sudo groupmod –n team team1 :- command used for modifying the group name here the first is the team name that has to be renamed and team1 is the existing group

Permissions in groups

1. $ touch file.txt
2. $ ls –l



Here first 3 characters :- User/Owner permissions

Middle 3 characters :- Group permissions

Last 3 characters :- Others permissions

In symbolic permissions are modified using operators:

+ : Add a permission, - Remove a permission, = set specific permission/overriding

1. $ chmod u+x f1.txt :- u represents execute permission for user
2. $ chmod g+w f1.txt :- g represents write permission for group
3. $ chmod o+x f1.txt :- o represents execute permission for others
4. $ chmod u+x f1.txt :- provides the execute permission for user
5. $ chmod g+x f1.txt :- provides the execute permission for the file
6. $ chmod g+w f1.tx :- provides the write permission for the file present in group
7. $ ls –l
8. $ chmod g-w f1.txt :- for removing the write permission from the file
9. $ chmod o+w f1.txt :- for removing the o + w f1.txt
10. $ chmod
11. Permissions with numeric format

Here Read(r) = 4, (w) = 2, (x) = 1

1. $ chmod 111 file1.txt :- command is owner, groups, others has only only for execute permission
2. $ chmod 444 file1.txt :- command is owner, groups, others has only for read permission
3. $ ls -l
4. $ chmod 222 file1.txt :- command is owner, groups, others only for write permission
5. $ ls –l
6. $ chmod 777 file1.txt :- command is owner, groups, others has all permissions
7. $ ls –l
8. $ chmod 741 file1.txt :- user group can read, write & execute, group has permission only for read and others has the permission only to execute
9. $ chmod 742 file1.txt
10. $ls –l
11. $ chmod 000 file1.txt :- for revoking all permissions

Chown command in linux

Change owner of a file i.e changing the owner of a particular file

1. $ sudo chown mike psa.txt
2. $ sudo chown :mike psa.txt :- changing the owner from ec2-user to mike
3. $ sudo chown stallin:stallin psa.txt

Finding files in Linux operating system

$ find [path] [expression]

1. $ sudo find /home –name f1.txt :- find files in home directory based on specific name
2. $ sudo find /home –type f –empty :- finds empty files
3. $ sudo find /home –type d –empty :- finds empty directories

Find files modified 30 days ago

1. $ sudo find /home –mtime 1 –print :- prints files which were created 1 day ago
2. $ sudo find /home –mtime 1 –delete :- deletes the file which were created 1 day ago

---------------------------------------------------------------------------------------------------------------------

Working with zip files

Zip is a utility used for archiving and compression. It allows users to combine multiple files into a single archive while reducing the total size

1. $ touch f1.txt, f2.txt, f3.txt, f4.txt
2. $ zip example \*.txt :- here example is the folder name and \* indicates zipping all the files present in the folder
3. $ ls –l :- shows all the files and folders in that directory and to unselect all the files
4. $ rm \*.txt :- selects all the files and delete the text files
5. $ ls –l :- it has only the zip file
6. $ unzip example.zip :- this extracts the contents of example.zip into the current directory

Networking commands in linux

1. Ping :- used to check the connectivity between your system and another host

* ping <Ip-address>

1. $ ping [www.facebook.com](http://www.facebook.com) :- connection b/w the local system and Facebook host gets established the response and time details along with ip address and fetched

Ctr + c enter

1. $ ping –c 4 :- after it works for 4 times it stop restricting the response
2. Wget command used to download the files from internet no installation
3. $ wget <url-of the file that has to be downloaded> paste the url :- it downloads all the files from online //can’t install the software but just can download the software
4. $ ls –l
5. $ unzip <file-name> :- copy the file name from ls –l
6. $ clear
7. $ cd eclipse/
8. $ cd ~ :- for reverting back to root directory
9. $ curl <http-address> :- used to send HTTP requests to a server and fetch responses in the form of a json format

Package managers in linux

* Amazon linux/ Red Hat Linux/ CentOs : yum
* Ubuntu / Debian: apt

1. Check if a package is installed or not:

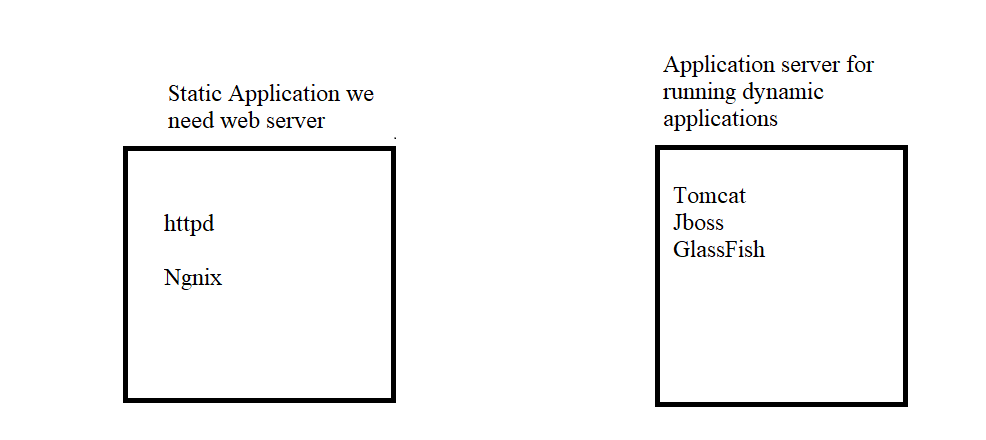
<package-name> --version ex:- git --version

1. Install a Package

$ sudo yum install <package-name> ex:- $ sudo yum install git

1. After install check $ git –version
2. $ whereis get :- to check the path where the git is getting installed
3. $ sudo yum remove git :- command is for uninstalling a particular file or directory
4. $ git –version :- it doesn’t show the software once its being removed/uninstalled
5. $ sudo yum install git –y :- for downloading a particular software
6. $ git –version
7. $ sudo yum install java –y :- command for installing a software
8. $ sudo yum remove java :- command is for uninstalling a software
9. $ sudo yum install maven –y :- command for installing maven
10. $ sudo yum update maven :- command is for updating a software

---------------------------------------------------------------------------------------------------------------------



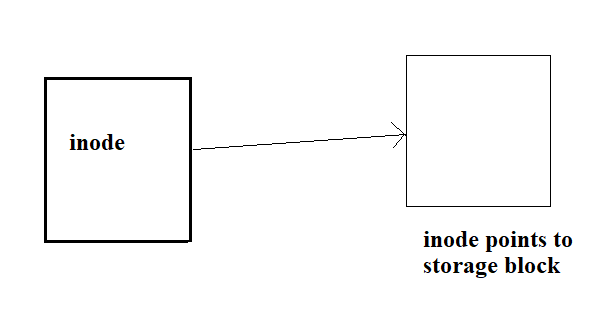
1. Steps for installing a static web server httpd

$ sudo yum install httpd -y

1. $ sudo service httpd status :- to check the server status
2. $ sudo service httpd start
3. $ sudo service httpd status :- it becomes active
4. In AWS there is no concept of firewall where the firewall filters the incoming requests and responses basically firewall is for monitoring the incoming requests for basically protecting the system from incoming threats through the incoming urls or through local internets etc.
5. If I want to access httpd server running on port 80
6. In AWS EC2 dashboard after selecting the EC2 virtual machine
7. In the bottom there is option for security tab
8. Select the security-groups url (like a firewall)
9. So security groups is very important in Linux os like it acts as a firewalls which is very much similar to firewall in windows
10. We can configure the port number that can be accessed here by default port 22 has the access whereas other ports it can’t be accessed by using security groups we can edit rule add port numbers and the protocols for the access
11. On installing httpd server $ cd /var/www/html is by default is created when httpd web server is installed
12. $ cd /var/www/html
13. $ sudo vi index.html
14. What is systemctl in lunix ?

systemctl is a command line utility used to manage and control system services on linux systems

1. $ sudo systemctl start httpd (starting a service)
2. $ sudo systemctl stop httpd (stoping the httpd service)
3. To start a service stop a service to enable or disable a service systemctl is used in linux command
4. $ sudo systemctl restart httpd (restart the httpd service) :- stops the service and restarts the serviec
5. $ sudo systemctl reload httpd (reloads the service without stopping and starting) the service doesn’t affect the existing users
6. $ top :- gives the usage of ram memory
7. If a file for ex : $ touch A.txt



1. $ ls –li :- here i is added to check the inode number
2. to check where the file data is present it points to that area of storage where file data is present
3. Create a new file and make sure it contains the same inode number through that it can take a backup
4. $ touch A.txt
5. $ cat >> A.txt
6. Hello world (save)
7. $ ls –li
8. $ ln A.txt B.txt :- link A.txt to B.txt both the data has same inode
9. $ rm A.txt

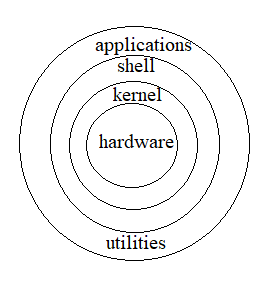
**Soft Links (creating desktop shortcuts)**

1. Files present in other folders can be created as easily accessible shortcut deleting the file at the original location makes the shortcut un accessible as it’s original file is deleted
2. $ mkdir f1
3. $ cd f1
4. $ touch A.txt
5. $ cat >> A.txt //Hello World
6. $ cd ~
7. $ ln –s /home/ec2-user/f1/A.txt B.txt
8. $ cat B.txt

**General Commands**

1. For accessing the task manager in windows we have ctrl + shift + esc key
2. $ ps aux :- its like task manager in linux os in vm
3. $ sleep 1000 & :- command for creating a process
4. $ ps aux | grep sleep
5. $ kill <Process-Id> :- killing the process
6. $ ps aux | grep sleep
7. $ kill -9 <Process-Id> :- forcefully killing the process
8. $ sleep 2000 & :- here & is background returns the process and processId
9. $ free –m :- how much % of ram utilization

---------------------------------------------------------------------------------------------------------------------Linux Architecture



Applications means gui based, when a request is sent from the browser application to the shell here shell is interpreter shell converts it to the format kernel can understand is a mediator b/w hardware and os

Shell is executing the commands it acts like a interpreter command is given to shell allows user to interact with system it’s going to convert it to the format which kernel can understand

Vice versa kernel going to convert it to a format and translate it such that hardware can understand

1. Change hostname in amazon linux
2. $ sudo hostname new-hostname :- this for temporary name change
3. $ exit :- trying to reconnect
4. $ sudo reboot :- for disconnecting the hostname
5. $ sudo vi /etc/hostname
6. Click on insert and change the content to

dev –env-1

1. $ sudo reboot
2. $ sudo vi /etc/hostname :- for permanent hostname changes
3. Change the env from dev-env-1 to qa-env-1
4. $ sudo reboot
5. Hostnames are mainly to check which services are running in which server



Scripts are used for automating the tasks like taking backups, automating testing in browser etc.

* Automate like take backup of files every night 12pm
* Writing all my commands in the script file
* Automatically run the scripts at 12 AM
* $ vi script.sh
* Cal
* Date
* Whoami
* Pwd
* $ sh script.sh //command for executing
* We do shell scripting with bash
* Shebang: #!/bin/bash :- is used for selecting the flavor of scripting that is required
* #!/bin/bash :- used for selecting the flavors of scripts
* We will use bash flavor

1. Variables
2. $ vi f1.sh

Name = “mike” //here double quotes is optional

echo hello world //echo is a case sensitive it’s a reserved key word

1. $ sh f1.sh

name = mike

echo $name

max\_value = 100

echo $max\_value

average = 10.3

echo $average

* Here local variables are accessible only within the files outside not possible
* Environment variables are accessible across the sessions
* Read only variables //values can’t be altered

Here expression evaluation is done differently

$ echo $((1+2))

$ echo $((2\*2))

$ echo $((4/2))

$ echo $((2.3 + 4.3)) //adding decimal values doesn’t work here

Here bc for decimal division

Bc is a command line calculator that supports floating point arithmetic

1. Decimal division

echo “scale=2; 2.3 + 4.3” | bc //so here for 2 or 3 its for decimal numbers and bc command after pipeline is used for

echo “scale=3; 2.3+4.3” | bc

echo “scale=3; 20.3/4.3” | bc

1. Indentation

num=1

if [ 2>3 ] ;then

echo “welcome”

fi

//after if there should be a space

//there should be space after square bracket as well

//if there is a number comparison or string comparison

For arithmetic operators this doesn’t work in that way

If [ $((20 > 3))] ;then

Echo “welcome”

Fi

If ((20>3)) ;then

Echo “Welcome”

Fi

If [ 20 -ge 3 ] :then

Echo “Welcome”

Fi

//-ge for >= -gt for > -lt for < -le for <=

For string comparison

If [ “welcome” > “hello” ] ;then

Echo”welcome” //1st way

Fi

* Use blank lines to separate logical sections

# Define a variable

my\_var=”Hello world”

# Print the variable

echo $my\_var

* Loops

for i in 1 2 3; do

echo $i

done

$ vi s1.sh

echo Enter your age

read age

if (( age > 21)); then

echo "Eligible"

else

echo"Not Eligible"

fi

$ sh s1.sh

$ sh s1.sh

Enter your age

22

Eligible

---------------------------------------------------------------------------------------------------------------------

1. Check whether the entered number is even or odd

**echo Enter the number**

**read** number

**if** (**(** number % **2** **==** **0** **));** **then**

**echo "Even number"**

**else**

**echo "odd number"**

**fi**

Output

$ sh s1.sh

Enter the number

3

odd number

1. Example for if-elif ladder

**echo Enter tihe number**

**read** x

**if** (**(** x **>** **0** **));** **then**

**echo "Positive number"**

**elif** **((** x **<** **0** **));** **then**

**echo "Negative number"**

**else**

**echo "zero"**

**fi**

**Output:**

$ sh s1.sh

Enter tihe number

0

zero

1. Print a variable using for loop

for **((i**=**0;i<5;i++))**

**do**

**echo $i**

**done**

1. Print a series of number using for loop reverse

for **((i**=**10;i>=1;i--))**

**do**

**echo $i**

**done**

1. Program to print the even and odd

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

**do**

**if** (**(** i % **2** **==** **0));** **then**

**echo $i is even**

**else**

**echo $i is odd**

**fi**

**done**

1. Example for while loop

**num**=**3**

**while ((** num **>** **0** **))**

**do**

**echo $num**

**((** num-- **))**

**Done**

1. **Program** for checking pin number

for **((i**=**0;i<3;i++))**

**do**

**echo "Enter the pin number"**

**read** pin\_number

**if** (**(** pin\_number **==** **1234));** **then**

**echo "welcome"**

**break**

**else**

**echo "Invalid pin number"**

**if** (**(** **i**=**=2** **));** **then**

**echo "card is blocked"**

**fi**

**fi**

**done**

---------------------------------------------------------------------------------------------------------------------

Functions

1. Example for function

function test() {

echo "From test 1"

echo 100

echo "Hello World"

}

Test

1. function test() {

echo $1

echo $2

echo $\*

echo $#

}

test "mike" 100

//here $1 is called as 1st argument & $2 is second argument

1. function with return type

**function** **add\_numbers(){**

**res**=**$((** **$1** + **$2** **) )**

**echo $res**

**}**

**sum**=**$( add\_numbers 10 20 )**

**echo $sum**

**//here sum is 30**

1. function for multiplication of 2 numbers

**function multiply(){**

**res=$(( $1 \* $2 ))**

**echo $res**

**}**

**prod=$( multiply 3 5 )**

**echo $prod**

**CRON jobs**

* automatically running some files at a particular times is called as cron jobs
* like example everyday at night 12’0 Clock taking backup of db
* deleting files at night 10 pm automating such tasks are done by cron jobs

synatx: \* \* \* \* \* file-name.sh

* 1st \* defines minutes (0 to 59)
* 2nd \* defines hours (0 to 23)
* 3rd \* defines days (1 to 31)
* 4th \* defines month (1 to 12)
* 5th \* defines week (0 to 6 or sun - mon)

Example 1 :- run the file everyday at 7am

0 7 \* \* \* file-name.sh

Example 2 :- run file everday at 3 pm

0 15 \* \* \* file-name.sh

Example 3 :- run every month 1st at 10 am

0 10 1 \* \* file-name.sh

Example 4 :- run the file every 15 minutes

\*/15 \* \* \* \* file-name.sh

Example 5:- run the file Monday to Friday 5 pm

0 17 \* \* 1-5 file-name.sh

Example 6:- run the file mon-sat 5:25 pm

25 17 \* \* 1-6 file-name.sh

Step 1 need to install cron jobs in linux

* Step-1 $ sudo yum install cronie
* Step-2 $ sudo systemctl start crond #(systemctl is used to start a service & stop a service)
* $ sudo systemctl enable crond
* Step-3 $ sudo systemctl status crond
* Step-4 $ crontab -e #(it will open a file \*/1 \* \* \* \* /home/ec2-user/script.sh)
* Step-5 $ chmod +x /home/ec2-user/script.sh
* Step-6 $ crontab -l

Cloud Service Infrastructure

* Taking infrastructure on rent
* Low maintenance cost
* Cost effective
* Easy to scale
* Quick
* Security

Cloud service providers

* Aws
* Azure
* Gcp
* IBM cloud
* Oracle cloud
* Alibaba clod
* Salesforce
* DigitalOcean
* Linode (now Akamai cloud)

Popular cloud service models :

Infrastructure as a service (Iaas) – provides virtualized computing resources eg. (AWS, EC2, Google Compute Engine) deploy application (select the server) – OS -Network -Storage

-platform + deploy we need to install softwares explicitly to run our application in the server after deployment

Platform as a service (Paas) – Server – OS -Network- Storage – platform

Getting platform along with resources EBS – elastic bean stalk

Need to take care of deployment

Software as a service (SAAS) :- best example is Salesforce is a crm application like zoho

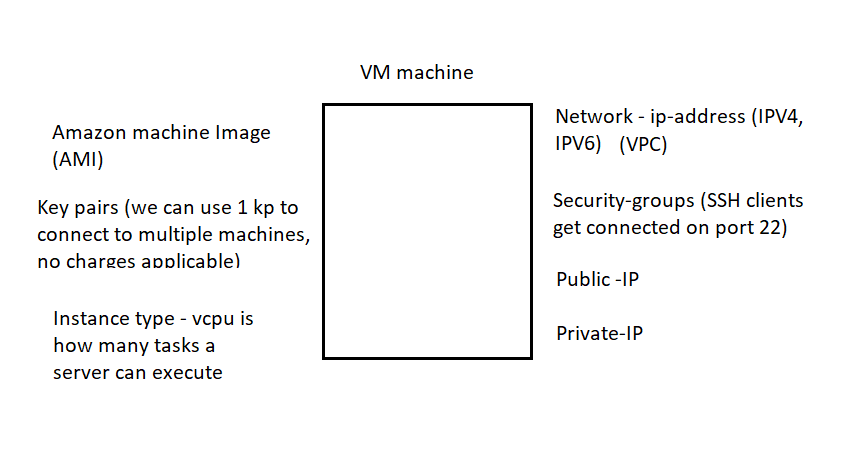
Server -Os -network -harware -platform -deployment

Everything is taken care this is called as software as a service mainly for customer relationship management (CRM)

For easy scaling we can make use of saas

EC2 Service

* Elastic cloud compute
* Helps us to create virtual servers /machines /instances
* Linux servers (8gb of storage) by default
* Window server (30 gb of storage) by default
* EBS(Elastic Block Storage) is used to provide some storage memory to the server
* Max of 16TB of storage can be given to a server through EBS



Ip-address

-> private-ip

-> public-ip

-> elastic-ip

Elastic-Ip address provides a fixed Ip address

Create an elastic ip click on the IP address add the VM’S required for this elastic-ip

EBS- Elastic Block Storage (HDD / SSD)

-> when a instance is created EBS volume for that is automatically created. This EBS Volume is called as a root volumes

EBS has 2 types of volumes

1. Root Volume – for linux 8GB and for windows 30GB
2. Additional Volume – per instance max 16TB (16000 GB) can be given

Go to folder in local system open git bash and connect the system with the AWS instance

$ lslbk

$ sudo mkfs –t ext4 /dev/xvdb :- make file system –t type ext4 file

$ mkdir psa

$ sudo mount /dev/xvdb psa //mounting the additional volume

$ cd psa

$ pwd

$ touch A.txt B.txt C.txt

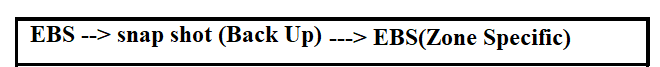
$df –h //command to check external disk is mounted or not

$sudo fsck /dev/xvdb //to check whether file system is intact

--------------------------------------------------------------------------------------------------------------------

Snapshots in AWS is like taking backup of the data in EBS like more frequently

1. Create a snapshot and copy the volume to same or a different region if needed
2. Once snapshot is available in the target region /AZ, you can create a new EBS volume from the snapshot in the desired AZ and attach it to the EC2 instance in that AZ



1. LifeCycle Manager (Scheduling the backups)
2. Volumes –elastic Block Storage gives us external storage to store data in EBS
3. Types of EBS