**LINUX COMMANDS JOEL**

-$ Lsb\_release –a Use to check the Version of the Distribution & Release Date

-$ uname –r Use to check the Kernel Version

-$ ls Provides a list of files one has created in home directory

-$ ls –version Use to show the version of the GNU core utility

-$ sudo X –version Use to show the X-Server version (Doesn’t allow me to ask Sai)

-$ uname -a prints the name, version and other details about the current machine and the operating system running on it

$ usermod -aG wheel ec2-user Adding user into wheel group for root privileges

$ sudo su - ec2-user getting back to ec2-user without passwdord asking

$ sudo chown -R ec2-user:ec2-user /var/lib/Jenkins (give ownership of Jenkins file to ec2-user)

**$ sudo systemctl restart service**

**$ timedatectl Checking time zone details**

**$ timedatectl set-timezone America/New\_York Setting timezone to EST**

**Connecting to Remote Server in Linux**

**Key Things to know**

* Identify your cloud user Ip address (AWS Instance Ip Address)
* User Name
* Password

Command to connect to the remote server

-$ ssh cloud\_user@IP

You will be prompted to agree if you are the cloud user, simply click yes

Lastly you will prompted for password

Enter your online cloud password

You are G2G

Note: -$ is unprivileged user

-£ is privileged user

**Commands**

LS – List of Files

* -$ ls Shows list of files
* -$ ls – a Show all file list (A- All)
* -$ ls –all shows list of all files
* -$ ls –ls shows list of all files with their corresponding permissions , file ownership and date created (Note L- Long, S- Size)

.files shown are considered hidden file in linux

* -$ ls –alsh it’s a more advanced –ls with all list of iles and their permissions ( H- Human Readable)
* -$ ls –al Formatted listing with hidden files

Variable – Linux can declare and use variables like any other programming language. Bourne Again Shell (BASH) is also a Programming Language

Examples

* -$ var1=”integer”
* -$ var1= 6
* -$ var2= “double”
* -$ var2= 7.6
* echo -$ var1 (6)
* echo -$ var2 (7.6)
* -$ pwd (Present Working Directory) Shows the current directory
* -$ cd dir change directory to dir
* -$ cd change directory to home
* -$ mkdir dir create directory dir
* -$ mkdir -p create directory dir recursively
* -$ rm file remove file
* -$ rm – r dir remove/delete directory dir
* -$ rm –f file force remove file
* -$ rm –rf dir force remove directory dir
* -$ cp file1 file2 copy file1 to file2
* -$ cd –r dir1 dir2 copy dir1 to dir2. Create dir2 if it doesn’t exist
* -$ mv file1 file2 rename or move file1 to file 2. If file2 is an existing directory move file1 into directory file2
* -$ ln –s file link create symbolic link link to file
* -$ touch file create or update file
* -$ ls –af list all the files
* -$ cat > file places standard input into a file
* -$ more file show/otput the contents of a file
* -$ head file display/ output top 10 lines of a file
* -$ tail file output last 10 lines of a file
* -$ tail –f file output the contents of a file as it grows starting with the last 10 line

COPYING FROM REMOTE

* $ scp “filename” “remoteusername”@”remote publicip”:/PATH/

Example copy file name joel to a remote machine ec2-user inside the path /home/ec2-user/

$ scp joel ec2-user@192.68.0.7: /home/ec2-user/

Process Management

* -$ ps Display current active processes
* **- $ ps –ef | grep process\_name**(application name) Example $ ps –ef | grep Jenkins (searching for a specific process)
* -$ top Diplay all running processes
* -$ Q Getting out of top
* -$ kill pid kill process id pid
* -$ killall proc kill all processes named proc \*
* -$ bg list stopped or backroung jobs
* -$ fg brings the most recent jobs to the foreground
* -$ fg a brings jobs a to the foreground

**MAIN NOTE: TO CLOSE OR EXIT ANY LINUX COMMAND SIMPLY TYPE Ctrl + C**

**It will display something similar to Example**

**At> ^C$**

**File Permissions**

**Permissions in Linux is done using numbers:**

**-Read Permission is given value – 4 (r)**

**-Write Permission is given value – 2 (w)**

**-Execute Permission is given value- 1 (X)**

**Note: There are three main categories of Entities that interact with the system in Linux which are User, Group and Other. So in granting permission we specify the permission value for each entity. Below is a summarized table of permission values in Linux**

|  |  |  |  |
| --- | --- | --- | --- |
| **Permissions** | **User** | **Group** | **Other** |
| **Read** | **4** | **4** | **4** |
| **Write** | **2** | **2** | **2** |
| **Execute** | **1** | **1** | **1** |
| **Full Permissions** | **7** | **7** | **7** |

**Permission Command**

* **-$ chmod (permissions values)**

**Example 1 Granting Full Permission using Linux for all entities**

* **-$ chmod 777**

**Example 2 Granting User Full permission, Group no permission and Other (read and write permission only)**

* **-$ chmod 706**

**Example 3 Granting User (Read and Write permissions), Group (Execute) and Other (No permission)**

* **-$ chmod 610**

**Searching**

* -$ grep pattern files search for patterns in files
* -$ grep –r pattern dir search recurring for pattern in dir
* -$ command | grep pattern search for pattern in the output of a command
* -$ locate file find an instance of a file

**System Info**

* -$ date shows the current date
* -$ cal show this month calendar
* -$ uptime displays the current time
* -$ w shows who is online
* -$ whoami show who you are logged in as
* -$ finger user display information about user
* -$ cat /proc /cpuinfo cpu information
* -$ cat /proc /meminfo memory information
* -$ df shows disk usage
* -$ du shows directory space usage
* -$ free show memory and swap usage
* -$ whereis app show possible location of app
* -$ which app

**Compression**

* -$ tar cf file.tar files create a tar named file containing files (c stands for create)
* -$tar xf file.tar extract the file from file.tar (x stands for extract)
* -$ tar czf file.tar.gz file create a tar with Gzip compression (c create, z zip f file)
* -$ tar xzzf file.tar.gz extract a tar using Gzip
* -$tar xjf file.tar.bz2 extract a tar using Bzp2
* -S tar cjf file.tar.bz2 create a tar using Bzip compression
* -$ gzip file Compresses file and rename it to file .gz
* -$ gzip –d fil.gz decompresses file.gz back to file
* $ rpm -i file.rpm extracting a rpm file

**Network**

* -$ ping host ping host and output result
* -$ whois domain get whois information for domain
* -$ dig domain get DNA information for domain
* -$ dig –x host reverse lookup host
* -$ wget file download file
* -$ wget –c file continue a stopped download
* **-$ netstat –tulpn | grep LISTEN** (to check the ports that are used by applications/processes)
* **$ ifconfig ( network Ethernet settings and other network activities)**

**Installation**

* -$

**Shortcuts**

* -$ Cltr + C Halts the current command
* -$ Cltr + Z Stops the current command, resume fg in foregroung and bg in background
* -$ Cltr + D log out the current session, similar to exit
* -$ Cltr + W Erases one word in the current line
* -$ Cltr + U Erases the whole line
* -$ Cltr = R type to bring up the recent command

**Using Putty**

* Get a new IP address by starting the aws ec2 instance
* Load the previous existing session with the previous IP address
* Copy the new IP address
* Check to see you are pointing to the Private Key
* Open Putty
* Type ec2-user
* Press enter

**SSH COMMANDS (Secure Socket Shell) -**is a protocol which allows you to connect securely to a remote computer or a server by using a text-based interface.

* **-$ ssh-keygen –t rsa Generating public/private rsa key pair**
* **-$ cd .ssh Change directory to .ssh**
* **-$ ls –al Display all and long private and public key**

**The above command will display…..**

**Private Key – id\_rsa**

**Public Key – id\_rsa.pub**

* **-$ sudo su Change to root user instead of home**
* **root@Joel:/home/jmasham# cp /mnt/c/Users/JOEL\ MASHAM/Desktop/key/joellll.pem /home/jmasham/ This command is used to copy the joellll.pem key download file from the desktop directory to /home/jmasham/ directory**
* **root@ip-172-31-18-104 ec2-user # ssh Displays the ssh information**
* **-$ sudo –i/-s/exit return you to user from root**
* **If the above doesn’t return you to -$ press Cltr + D**

**To access your instance:**

1. Open an SSH client. (find out how to [connect using PuTTY](https://docs.aws.amazon.com/console/ec2/instances/connect/putty))
2. Locate your private key file (joellll.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

**chmod 400 joellll.pem**

1. Connect to your instance using its Public DNS:

**ec2-3-15-161-27.us-east-2.compute.amazonaws.com (NOTE: Use the example command below instead of this)**

**Example:**

**ssh -i "joellll.pem" ec2-user@ec2-3-15-161-27.us-east-2.compute.amazonaws.com**

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our

**LINUX ADVANCED**

**Creating two directories and a file inside a directory using one command**

**$ mkdir -p DIR1/DIR2 $$ touch ./filename**

**SHELL SCRIPTING**

* **-$ cat /etc/shell** Displays the shells in your Distribution

Example Ubuntu Supports:

/sh

/dash

/bash

/rbash

We will be using /bash shell which stands for Bourne Again Shell.

* **-$ which bash To show where bash is located (Result /bin/bash)**

After getting the two piece of information the next step is to open the Ubuntu text editor which is called “nano”

* **-$ nano Open nano text editor**

Syntax

#! /bin/bash/

* -$ vi file.sh Open/create the file .sh in vi editor
* -$ Press Escape if you want to save and exit
* :w used to save
* :q used to quit/exit
* :wq usedt to save and exit
* ***To execute the file in terminal use ./file name*** (Note: make sure you have right permissions)

**Variable- Are containers used to store data**

* **System Variables – Created or generated by the OS (CAPITAL CASES)**
* **User Defined Variables – Created and maintained by the user**

**e.g SYSTEM VAR**

* -$ BASH
* -$ BASH\_VERSION
* -$ HOME
* -$ PWD

**USER DEFINED VAR**

* Name=Joel
* Echo my name is $name

**Allowing User to enter variables**

**Example:**

Echo “Enter Name: “

**Read name (To get input from the terminal we use read command)**

Echo “Entered Name: $ name”

More than one name.

Echo “Enter Names: “

Read name1 name2 name3

Echo “Names: $name1, name2, name3”

**Reading and input in the same line**

**Example: Enter Username**

Read –p “username : “ user\_var ( - p is used for entering variable in same line)

Echo “username : $ user\_var”

Read –sp “password : “ pass\_var (-s means silent)

Echo “password: $ pass\_var”

**Read without declaring variable**

Echo “Enter Username: “

Read

Echo “Username : $ REPLY” (We system variable to capture variable which hasn’t been declared by the user)

**Entering TEXT (SED COMMAND)**

* **-$ cat > file.txt (used to allow text input to a file)**
* **-$ sed ‘s/y/x/ determinate change value (1,2,g,3g etc)’ file.txt (used to swap the position of x and y in a text file) NOTE: 1 – Replaces the first occurrence of a word in a paragraph. 2- Replaces the second occurrence of a word in a paragraph….. g- replaces the global occurrence of a word in a paragraph. 3g – replaces global all the occurrence of a word from the third occurrence. …. ng – replaces globally the nth occurrence of a word in paragraph.**

**AWK COMMAND**

Default Behavior: Awk prints all lines of data from the specified file.

-$ awk ‘{print}’ file.txt

-$ awk ‘ /word/ {print}’ file.txt prints all lines which matches the word

-$ awk ‘ {print $1, $4}’ employee.txt prints all the 1st and 4th variables in the line

Example

Joel US 31 85,000

Harsh US 33 85,000

Vikas CA 30 80,000

John CA 32 80,000

The above awk command will print

Joel 85,000

Harsh 85,000

Vikas 80,000

-$ awk ‘{print NR, $0}’ employee.txt Used to display line number

e.g

1 Joel US 31 85,000

2 Harsh US 33 85,000

3 Vikas CA 30 80,000

4 John CA 32 80,000

**Pass Arguments To a Bash Script**

Args=(“$@”)

* Echo $ {args [0]} $ {args[1]} $ {args [2]} Prints the array values entered
* Echo $@ Prints the same values entered because it is declared variable
* Echo $# prints the number of variable (3)

**If Statement**

**SYNTAX If [ condition ]**

**then**

**Statement**

**Fi**

**Note:**

**Integer Compariso**

* **-eq is equal []**
* **- ne not equal []**
* **- gt greater than []**
* **- ge greater than or equal to []**
* **- lt less than []**
* **< less than (())**
* **<= less or equal(())**
* **> greater than (())**
* **>= greater or equal(())**

**String Comparison**

* **= is equal to []**
* **== is equal to []**
* **!= not equal to []**
* **< less than [[]]**
* **> greater than [[]]**
* **-z string is null, it has zero length**

**Example**

**count=10**

**If [ $ count –gt 7 ] (Using and to join if [ “$count” –gt 7 –a “$count” –lt 11)**

**then**

**echo “condition is true”**

**Fi**

**Using else**

**word=g**

**If [[ $ word > k ]]**

**then**

**echo “conditions is not true”**

**else**

**echo “condition is true”**

**fi**

**Task**

1. **Write a script that allows user to check for a file from a list of all file. The script should display file\_name found if the file is found otherwise it should display file\_name not found.**

**Answer**

**echo –e “Enter file\_name : \c” (Note: -e and \c are used to keep cursor in the same lien)**

**read file\_name**

**if [ -e $file\_name ] (-e is used to check whether the file exists or not)(-f can be used instead of –e to check if the file is a regular file or not) (-d can be used instead of –e for checking directory) (-b can be used instead of –e to check for block special file) (-c can be used instead of –e to check for character special file) (-s can be used instead of –e to check if the file is empty or not)**

**then**

**echo “$file\_name found”**

**else**

**echo “$file\_name not found”**

**fi**

**HOW TO CREATE MULITPLE USERS**

1. **Create a file for user ($ sudo vi users.txt)**
2. **Add users and passwords in the users.txt file**

joelm:213254lost:1002:1002:Tecmint Admin:/home/ravi:/bin/bash

jm:@!#@%$Most:1003:1003:Tecmint:/home/tecmint:/bin/bash

1. **Change permission of the users.txt file ($ chmod 600 users.txt)**
2. **Run the newusers command to create all the users in the file ($ sudo newusers users.txt)**
3. **Check to see if users are installed ($ cat /etc/passwd)**

**GIT**

**Installing Git**

**Ubuntu ----- apt-get install git**

**Putty (EC2) ------------ sudo yum install git –y**

**Configuring Git**

Create git directory -$ mkdir mygit

* -$ cd mygit change directory to mygit
* -$ git init initialize git
* -$ git status check status to see if git is initialized

**Github Credentials**

Username: imanijoe21

Password: Jesuslovesme123\*

Repository Name: DevopsTraining

* Install Git for windows
* Create a directory under C: for your poject
* Open the directory and right click.
* Select git bash here from drop down
* The git command line interface will open
* Create a git repository using the **git init** command
* Add the remote repository (Origin) using the **git add origin (link)** command

Example

Type the following command to add origin to my project.

**git remote add origin “url from the github clone/download”**

**git pull origin master**

**git add**

it will ask to setup global config

* **Git config –global user.email “email”**
* **Git config –global user.name “name”**
* **Git commit –m “enter text of what you committed”**

Creating Branch

* **git branch branch\_name**
* Check current branch **(git branch)**
* To migrate to different branch **( git checkout branch\_name)**
* Pushing changes into Origin **( git push –u origin branch\_name)**
* Creating and directly migrating to branch we use $ git branch –b branch\_name

Merging Branches

* Create branch ( git branch branch\_name)
* Create file ( touch filename)
* Add txt ( cat > filename)
* Perform git add .
* Perform git commit –m “notes”
* Enter text
* Create second branch (git …)
* Create file (touch filename)
* Add txt (cat > filename)
* Perform git add .
* Perform git commit –m “notes”
* Perform git merging using the git merge command ( git merge branch2)
* Correct the merge conflict error by removing the unnecessary symbols in the vi editor
* vi editor filename
* perform git add .
* perform git commit –m
* push the changes in both branches (**git push –u origin branch1**)
* push the changes in both branches **(git push –u origin branch2**)
* You can commit more than 1 file using (**git commit –a –m “Text”**)

<https://www.youtube.com/watch?v=_d-c9uGcUrU> Tomc

Troubleshooting Remote Git Repository:

## 1. List your existing remotes

To list the existing remotes we open the terminal and type in the following command:

$ git remote -v

If you copied the link to the repository from **Clone with SSH** in your GitLab, the output should look something like this:

origin [git@gitlab.xenovation.com](mailto:git@gitlab.xenovation.com)

:user/repository.git (fetch)  
origin [git@gitlab.xenovation.com](mailto:git@gitlab.xenovation.com):user/repository.git (push)

If you copied the link to the repository from **Clone with HTTPS** in your GitLab, the output should look something like this:

origin https://gitlab.xenovation.com/user/repository.git (fetch)  
origin https://gitlab.xenovation.com/user/repository.git (push)

Note: To find the SSH and HTTPS URLs, go to your GitLab, select your project, and click on **Clone**.

## 2. Change a remote Git repository

We can change the remote repository by using git remote set-url command:

$ git remote set-url origin [git@gitlab.xenovation.com](mailto:git@gitlab.xenovation.com):user/repository2.git

The command takes two arguments: existing name of the remote (in our case **origin**) and a new remote URL (in our case [**git@gitlab.xenovation.com**](mailto:git@gitlab.xenovation.com)**:user/repository2.git**)

In case you change your remote repository to https URL, you will be prompted for your username and password next time you use git fetch, git pull or git push.