1. DevOps and Build Release Engineer

Introduction

- Ls – List files and directories.
- Ls –l – list files and directories with long format
- Ls –lt – Long list format with time
- Ls –lrt – reverse of ls –lt.
- Pwd – present working directory
- Mkdir – creating a new directory
- Touch – used to create new empty file
- Cd – change directory
- Ls –a – list the hidden files
- Cd .. –to come out of present directory
- Cd ../.. – if you want to come out of 2 or more directories.
- Cd - : acts as a recall between last 2 commands
- Cd /temp1/temp2/ - changes to one directory to last directory
- Cat – used to display the content of a file on console
- Vi – used to edit a file on vi editor
    - Esc I – insert mode in vi editor.
    - Esc :wq! – save and quit the file. W-write,q-quit,!-forcefully.
    - Esc q! – quit without saving the content.
    - Esc :set number/ esc :setnu – which sets the number for a file.
    - Esc :setnonu – used to remove the line numbers for a file.
    - Replace a string
        - Esc :%s/string1/string2/g
            - G is not mandatory, g means globally.
            - Esc :%s/linux/docs/g
            - S is substitute
            - Esc :1 s/linux/docs/g – changes in 1st line only
            - Esc :1,5 s/linux/docs/g –changes 1 to 5th line.

- Esc :7,$ s/linux/docs/g – changes 7th to end of line.
- Esc :1 s/linux/docs/1 – changes 1st line 1st occurrence only.

- Search a pattern/string
  Esc:/pattern – it will move to 1st occurrence then keep pressing n to 2nd and so on
- Delete the whole line
  Esc:dd
- Cut and paste in a file
  Cut –d(delimiter) –f(colname) filename
  Example: cut –d- -f2 f1.txt → it removes – and prints 2nd column in a file.
  cut –d- -f2,3 f1.txt
  cut –d- -f2-5 f1.txt
  echo ‑HELLO‖ | cut –c 3
  paste-displays lines of multiple files line by line
  paste f1.txt f2.txt
  paste –d: f1.txt f2.txt
  paste –s f1.txt f2.txt

Commands

- **Cp** command (Copy)
  Is used to copy content of a file from one to another file
  If dest file doent exists, it will create and copy it.if already exists then it will override the content.
  Cp src dest
  Cp file1 file2
  Cp –r temp1 temp2-whole temp1 info ll copied to temp2 recursively.
- **Mv** command (Move)
  The source will get deleted and moved to destination. Or can be called as rename.
  Mv file1 file2
  Mv file dir/
  Mv *.txt dir/
- **Chmod** command
  Used to change permission of a file or directory
  -rwxr--r--
  -=file or directory or link, rwx=user,r--=group,r--=others
  R=read,w=write,x=execute
  It will be in binary format
  Chmod 777 filename – all permission to a file
  Chmod –R 777 dir – all permission to a dir
  Remove or add permission
  Chmod u+w file –user and write permission to a file
  Group and other full permission

Chmod g+rwx,o+rwx file
Chmod g-w,o-w file
Chmod a+rwx file

- **Chown** command
  Change ownership of a file or dir
  Chown newowner filename
  Chown –R newown dirname
- Chgrp command
  ```
  chgrp hope file.txt
  ```
- Shells

  bash rc, kshell, cshell

  to change the shell use,

  chsh

  used to show current shell, echo $SHELL.

- Wc command
  Used to count number of lines, words,characters in a file
  Wc filename
  Wc –l filename,wc –w filename,wc –c filename
- **Du and df command**
  Du-disk usage or size of a file, df-disk free / size of drive
  Du –sh filename
  Du –sh *
  Df –h .
  Df -h
- Echo command
  Print statement
  Echo ‑PRADEEP‖
  Echo –e ‑HI \n HELLO‖
- **Redirect and append command**
  Redirect (>)-used to write output of a command to a file, if not exists it will create new and write it . if already present it will override.
  Echo ‑hi how r u‖ > text
  Cat text
  Ls –lrt > text
  Cat text
  Du –sh * > size
  Cat size
  Append (>>) – used to attach the content or output of a command to end of a file. It will not override the existing content of a file.
  Echo ‑PRADEEP‖ >> log
  Cat log

Echo –DEEPU‖ >> log
Cat log

- **Grep command**
  Is used to search pattern/string in a file.
  Grep –pattern‖ filename
  Grep –w –pattern‖ filename – used to search only word
  Grep –pattern‖ *
  Grep –l ―pattern‖ * - prints only file names which pattern is present
  Grep –w –l ―pattern‖ *
  Grep –R –l ―pattern‖ * or rgrep
  Grep –c –file‖ * - used to count number of lines.
  Grep –e –pattern1‖ –e –pattrn2‖ ---so on- used to search multiple patterns
  Also u can use egrep
  Grep –File‖ * case sensitive
  Grep –I –file‖ * case insensitive
  Grep –^d‖ filename –used to print all lines start with d
  Grep –2$‖ filename – used to search all files which ends with pattern 2
  Grep –v –docs‖ filename- print all lines except docs
- **Pipe filters** (|)
  Pipe is used to give output of a one command as input to next command
  Ls | grep –^t‖ | wc -l
- **Head and tail command**
  Head -Is used to display 1st part of a file, by head filename -> by default it display 10 lines.
  Head -10 filename
  Tail – used to display last portions of a file
  Tail -2 filename
  Recently modified files
  Ls –lrt | tail -1 or ls –lt | head -1
  Head -99 filename | tail -1 – used tp print 99th line of a file
  Head -5 file | tail -3 – used to print 2nd to 5th line of file
  Head -1 filenmae;tail -1 filename – used to club commands but commands runs independent, to display 1st line and last line of a file.
  Head -99 filename | tail -1 | wc –w –count number of words in 99th line of a file
- **More and less command**
  More filename – used to display content of a file page wise, can scroll down but can't up
  Less filename-used to display content page wise, we can scroll up and down.
- **Sed command (Stream Editor)**
  Is used to edit file without opening it, need to replace string in a file.
  Sed _s/file/docs/g' filename
  Need to replace in a 1st line
  Sed _1 s/file/dcs/g' filename

Need to replace string from 3 to 5 line
Sed _3,5/fil/dcs/g‘ filename
Need to replace string 7 to end of line
Sed _7,$ s/file/dcs/g‘ filename
If u need to change in original files,then
Sed –I _s/fil/dcs/g‘ filename
-i=insert to a file/modify to file, if you don‘t use it wont affect in original file.
Print only 5 or 99th line
Sed –n _5 p‘ filename
Sed –n ‘99 p‘ filename
Print 4th to 9th line
Sed –n _4,9 p‘ filename
Print 5th to end of line
Sed –n _5,$ p‘ filename
Delete a line from a file using a sed
Sed _4 d‘ filename
Sed ‘99 d‘ filename
Sed –I _4 d‘ filename

- **Awk command**

  Awk –F ― ― _{print $2}‘ filename
  Is used to cut content of a file column wise and row wise.
  Ls –lrt | awk –F –‖ _{print $2,$3}‘-2nd and 3rd col
  Ls –lrt | awk –F –‖ _{print $NF}‘ –last col
  Ls –lrt | awk –F –‖ _{print $(NF-1)}‘-2nd last col

- **Find command**

  Is used to find the location of a file/dir. Find is a automatic recursive, it searches in sub-directories automatically.
  Find. –name –filename‖
  Find . –name –log‖ - . is pwd
  Find /home/ -name –log‖
  Find –iname –log‖
  Find /home/mobaxterm –iname –log‖
  Find . –iname –type f –log‖
  Find . –type f –iname –log‖
  Find . –iname –type d –log‖
  Find . –type d –iname –log‖
  -mtime –is last modified time
  Find . –type f –mtime +90
  Find . –type d –mtime +90
  Find . –mtime +90
  Find . –type f –mtime -90
  Find . –type d –mtime -90
  Find . –mtime -90
  -mmin –is last modified time

Find . –type f –mmin -9
Find . –type f –mmin +90
Find . –type d –mmin -90
Find . –type d –mmin +90
Empty and non-empty
Find . –type f -empty
Find . –type d -empty
Find . -empty
Find . –type h –empty :h or L is a link
Find the permission of a files
Find . –type f –perm 0777
Find . –type d –perm 0644
Find . –perm 0754
Nonempty
Find . –type f –not –empty
Find . –type f ! -empty

- Maxdepth command
  How do you restrict recursive search for the dir search in find command
  Find . –iname ‑test‖ –maxdepth 2
- Rm command
  Used to remove a file-> rm filename
  Rm –rf –> used to remove dir
- **Xargs**
  Is used to pass as a args to a next command
  Output of one command will be passing as args to next command.
  how to find files which are modified 90 dyas back and del that.
  Find . –type f –mtime +90 | xargs rm
- **Exec command**
  Find . –type f –mtime +3 –exec
  Is used to replace the current shell with the command without spawing a new process and used to assign file descriptor to file.

Soft link and hard link

Is a shortcut to a file, if you make any changes in original file it gets reflected in the link, if I delete original file soft link wont work.
Ln –s filename softlinkname
U can create a file or dir

Hard link
Ln filename hardlinkname
Is a shortcut to a file, if you make any changes in original file it gets reflected in the link, and if I delete the original file, hard link wil work because it points to inode of a file.

- Umask
  Is used to set default permission on a system. It is reverse of chmod.

Umask 000 –who were creates the file on a system will have all permissions.
Umask 777 means- no permission to a file

- Ssh
  It is used to convert to remote server
  Ssh username@server2
  Password:

  Hostname
  Ssh uses port 22
- Scp
  Is used to copy file or dir from server to another server.
  Scp file username@server2:/home/../../
  Password:

  Scp –r dir1 username@server2:/home/../../

  Scp –p file1 username@server2:/home/../../
  P is preserved permission as same

  Scp –rp dir username@server2:/home/../../
  -rp for dir permission as same
- **Ping**
  Is used to check whether the server is up and down
  Ping ip or hostname
- **telnet**
  it uses port 23
  is used to authentic account credentials on remote server
  used to break firewall
  telnet ip or hostname
- **rsync**
  is used to copy files from one server to another server and also within the server, while copying the data if copy is stopped due to network issues in between, if I use _scp‘ it will start copying from begin
  if I use rsync it will start copying where it is stopped.
  Rsync file user@server2:/home/test/../
  If you want to copy in server
  Rsync src dest
- size command
  find . –size +0 –shows empty file
  find . –size +1 – shows nonempty file
- su
  super user
  su –root or su –
  used to login as root user

root has highest permission

- **sudo**
  superuser does
  is used to run command with root permissions
  sudo username
- **who**
  is used to check who are all logged into system
  who
  who | wc -l
- **whoami**
  is used to check, who you logged in name
  whoami
- **uname**
  to check linux version
- uname –a
  complete information about linux
- uname –v
  version and time of linux
- ps
  used to list current running process on a system
  ps –ef | grep ‑system32‖
- **kill**
  used to stop process
  -9 is a special signal to terminate that process
  Kill -9 PID
  Killall -9 pname
  Sudo service servername/Pname stop
  Sudo service servername/Pname start
  Sudo service servername/Pname restart
  Pkill -9 –u ‑username‖ – stop process started by particular user
  Ps –u username or ps –ef | grep ‑system32‖ – list process started by particular user
  How do you run script or command in a background we need to see & at the end of the command.
  How do you run process in foreground
  Fg PID
- **sort**
  is used to sort the data
  sort filename
  sort –r filename
- **uniq**
  used to print only unique values (it removes duplicate)
  it will work only when the contents/data are consecutive values/info/data
  uniq will be always used with the sort

uniq filename
cat file | sort | uniq

- **tee**
  used to write output of a command to a file as well as display on console/prompt
  cat log | tee log1
  ls –lrt | tee –a log →used to append to log
- .profile / .bashrc
  Vi .bashrc
  Echo –hello gm‖
  Pwd
  Esc:wq!
  It is an auto execute file which gets executed automatically as soon as you logged into the system.
- **Netstat**
  Used to check free ports on a system.
  List all listening ports tcp or udp
  Netstat –a
  Netstat –at
  Netstat –au
  Netstat –lt
  Netstat -lu
- **Mount**
  Attaches a filesystem, located on same device or other to a file tree.
  Mount –t type device dir
  Unmounts-detaches the specified file system from file hierarchy

Shell Scripts

- Vi ex.sh

  #!/bin/bash ---------- →>>>>>>>>>>she bang (it invokes bashrc shell)

  Echo –hello‖

  Echo –pradeep‖

  Pwd

  Esc:wq!

  To execute-> ./ex.sh or sh ex.sh or bash ex.sh

- If we wont write she bang it uses default shell.
- 1st line of shell scripts called shebang.
- Writing more linux commands into a single file and working on that and executing a file is called shell script.
- Variables

  #!/bin/bash

  Var=abc

Var1=123

Var2=‖pradi hello‖

Echo –my name: $var‖

Echo –my no: $var1‖

Echo –my names: $var2‖

- Command line args

  #!/bin/bash

  Echo –my name is:$1‖

  Echo –my city:$2‖

  Echo –my filename:$0‖

  Sh Ex3.sh pradeep shimoga

  $1 is 1st args,$2 is 2nd args,$0 is name of script, it will take $0---$9 after that ${10},${11}…

- If statement (Conditional statement)
- If [ ]

  Then

  Statement

  Fi

- If [ ];then

  Statement

  Else

  Statement

  Fi

- If [ ];then

  Statement

  Elif [ ];then

  Statement

  Else

Statement

Fi

- Was(write a script) to check number is 5 or not.

  #!/bin/bash

  If [$1 –eq 5];then

  Echo ‑$1 is five‖

  Else

  Echo ‑$1 is not five‖

  Fi

  Sh ex.sh 5, sh ex.sh 2

- Was to find biggest of 2 no‘s
  #!/bin/bash
  If [$1 –gt $2];then
  Echo ‑$1 is big‖
  Else
  Echo ‑$2 is big‖
  Fi
  ./big.sh 1 2
- Condition

  -eq=equal

  -gt=greater than

  -ge=greater than or equal to

  -lt=lesser than

  -le=lesser than or equal to

  -ne=not equal

- Biggest of 3 nos

  #!/bin/bash

  If [$1 –gt $2] && [$1 –gt $3]

  Then

  Echo ‑$1 is big‖

elIf [$2 –gt $2] && [$2 –gt $3]

Then

Echo –$2 is big‖

Else echo –$3 is big‖

Fi

./ex.sh 1 2 3

- How to make a shell script file into a command

  Cp ex.sh ex

  Cp ex /usr/bin/ or cp ex /bin/

  Ex 1 2 3

Notations

$#-to display total number of arguments

$*-all arguments passed to a scripts

$?-status of last executed command

If it is 0, success. Else failure (1,2,)

$$-display the present PID.

$@-all arguments passed to a script are stored in array format.

$!-PID of current running process, which went into background.

#-is used to commenting a code.

&&-AND,||-OR,!-NOT.

- Restrict to pass two args
  If [$1 –ne 2];then echo –pass only 2 args‖;exit 1;fi
- Was to check whether the args is file/dir/link.
  #!/bin/bash
  Echo –enter the name of file/dir/link‖
  Read name
  If [-f $name];then echo –$name is a file‖
  elIf [-d $name];then echo –$name is a dir‖
  elIf [-L $name];then echo –$name is a link‖
  else echo –$name doesn‘t exits‖
  fi

- Notations
  -f=file,-d=dir,-L or –h=link,-e=exist or not,-r=read permission,-w=write,
  -e=execute, readlink soft1=used to read links and display the path of the fileor link.
- Was to check whether the args is file/dir/link and display the entered file content
  #!/bin/bash
  Echo –enter the name of file/dir/link‖
  Read name
  If [-f $name];then echo –$name is a file‖;cat $name elIf
  [-d $name];then echo –$name is a dir‖ ;cat $name elIf [
  -L $name];then echo –$name is a link‖ ;cat $name else
  echo –$name doesn‘t exits‖
  fi
- While loop
  While [ ]
  Do
  ……
  Done
- Was to print entered number to 1
  #!/bin/bash
  N=$1
  While [$n –gt 0];do
  Echo –$n‖
  N=`expr $n – 1`
  Done
  ``=backquote(below esc key)
- Was to find factorial of a given number
  #!/bin/bash
  Num=$1
  Fact=1
  While [$num –gt 1]
  Do
  Fact=`expr $fact \* $num`
  Num=`expr $num – 1`
  Done
  Echo –factorial of a number is: $fact‖
- Was to add 2 numbers
  #!/bin/bash
  N=`expr $1 + $2`
  Echo –$n‖
- Was to subtract from a bigger number to smaller number and print result.
  #!/bin/bash

If [$1 –gt $2];then
Sub=`expr $1 - $2`
Else
Sub=`expr $2-$1`
Fi
Echo –$sub‖

- Was to add given number
  #!/bin/bash
  Num=$1
  Add=0
  While [$num –gt 0]
  Do
  Add=`expr $add + $num`
  Num=`expr $num – 1`
  Done
  Echo –$add‖
- While loop to read file line by line
  While read line
  Do
  Echo –$line‖
  Do < filename/$1
  If there are 4 lines in a file, it will executed 4 times in a loop.
- Was to count number of words in each lone of a file
  #!/bin/bash
  Num=1
  While read line
  Do
  words=`Echo $line | wc –w`
  echo –$num lines and $words word‖
  num=`expr $num + 1`
  done < $1 or filename
- Was to print names if there age is more than 40 in the data table.
  U can use –AWK command as shorthand‖
  Awk –F — — _$2>40{print $2}‘
  ==========or==========
  #!/bin/bash
  Sed _1 d‘ $1 > temp
  While read line
  Do
  Age=`echo –$line‖ | awk –F – – _{print $3}‘`
  If [$age –gt 40]
  Then
  Echo $line | awk –F — — _{print $2}‘

Fi
Done < $1

- Was to reverse a content of file into reverse order.

```
#!/bin/bash
File=$1
Lines=`car $file | wc –l`
While [ $lines –gt 0 ]
Do
        Head -$lines $file | tail -1 >> reverse
Lines=`expr $lines – 1`
Done
Cat reverse
Rm reverse
==========or=========
Tac filename (reverse of cat)
```

- Was to reverse a word

```
#!/bin/bash
Str=‖hello‖
Len=`expr $str | wc –c`
Len=`expr $len – 1`
Rev=‖‖
While [ $len –gt 0 ]
Do
        Rev1=`echo $str | cut – c$len`
        Rev=$rev$rev1
Len=`expr $len – 1`
Done
Echo –$rev‖
.....or
.....Echo
–pradi‖ | rev
```

- For loop

```
For I in 1 2 3 4 5
Do
        Echo –$i‖
Done
```

- Was to add n numbers

```
#!/bin/bash
Var=‖2 3 4 5‖
Sum=0
For I in $var
Do
        Sum=`expr $sum + $i`
Done
```

Echo ‑$sum‖

- Was to find biggest of n numbers

```
#!/bin/bash
Big=$1
For I in $*
Do
        If [ $big –lt $i ]
Then
        Big=$i
Fi
Done

Echo ‑big is $big‖
```

- Set –x

  Used to debug a shell script

- Was to find smallest of n numbers

```
#!/bin/bash
Set –x
Small=$1
For I in $*
Do
        If [ $small –gt $i ]
Then
        Small=$i
Fi
Done
```

Echo –small is $small‖

- Top
  Whether each process is taking how much CPU usage and memory.
- Uptime
  Is used to check from how long the system is running and also used to check load average.
- Load average
  Load is a measurement work of system performing, this measurement displayed as number. A complete idle system or pc is numbered as 0. Each running process for cpu resource adds 1 to load average.
- Command used to list only files and directories
  Ls –lrt | grep –^d‖ ---for directories
  Ls –lrt | grep –v –^d‖ ---for directories
  Find . –type d –ls --- dir
  Find . –type f –ls ---files
  Find . –maxdepth 1 –type d
  Find . –maxdepth 1 –type f
- Special characters
  Have special meaning to a shell
  To escape special meaning or special charaters we use backslash (escape character-\ ) to special characters.
  Special characters like $,$#,$!,$*,\,*,^,` and etc
- Was to find factorial of a given any number in command line args.

```
#!/bin/bash
For I in $*
Do
        Fact=1
        Num=$i
        While [ $num –gt 1 ];do
                Fact=`expr $fact \* $num`
                Num=`expr $num -1`
        Done
        Echo –$fact‖
Done
```

Crontab (vv imp)

Is a scheduler, which is used to schedule scripts on linux server.
Crontab –e ---is used to edit cron job
Crontab –l ---is used to liust cron jobs

```
*       *       *       *       * /script path
Mins   |hrs    |date   |month |day of week
```

- Schedule to run the script on specified date and time
  - 12th june at 10 am mon
    00 10 12 06 01 /script path
  - 13th june at 4 pm tue

00 16 13 06 02 /script path

- Everyday at 2pm
  00 14 * * * /script path
- 03.30 pm only on tue
  30 14 * * 02 /script path
- Mon-fri at 4pm
  00 16 * * 01-05 /script path
- Mon and Tuesday at 5.30pm
  30 17 * * 01,02 /script path
- Mon every one hour
  00 * * * 01 /script path
- Mon every 30 mins
  */30 * * * 01 /script path
- Mon at 4pm, script should run at every 5min
  */5 16 * * 01 /script path
- 05.15pm one the friday
  15 17 * * 05 /script path

- Sat 10pm
  00 22 * * 06 /script path
- Every month 1st at 11 am
  00 11 01 * * /script path
- If ec2 doesn‘t consists crontab, then
  Sudo yum instal cron
  .....or.....
  Sudo yum install update
  And to check crontab use,
  which crontab
  then,
  /home/…//.//big3.sh > log1
  Crontab –e
  *****/script path
  Crontab –l
- Need to set a crontab every month end at 11 am
  00 23 * * * [`date +%d` -eq `echo \`cal\` | awk _{print $NF}‘`] && job.sh

Realistic Scripts

- Builds (different compilation or developing files) were failing due to memory issue it was giving an error like ―no space on disk‖. To resolve this problem/issue,
- **I have written a script to check disk memory and send a email notification, saying that memory is 90%, and please take appropriate action.**

  Vim memch.sh

  #!/bin/bash

  Mem=`df –h . | tail -1 | awk –F ― ― _{print $4}‘ | sed _s/%//g‘`

  If [$mem –gt 40]

  Mail –s ‒memory reached‖ –c pradeep@gmail.com < filename

  Fi

  --------filename at the end of mail line is,

  You can write a body of mail in that file can read to the mail command.

  Else U can try like,

  Echo ‒hi memory reached‖ | Mail –s ‒memory reached‖ –c pradi@gmail.com

  To run memch.sh for every min,

  Crontab –e

  * * * * * /memch.sh

- Mail command

  Is used to email the file or information to the specified email.

  Mail –s ‒memory reached‖ –c pradi@gmail.com pradeep@gmail.com < filename

  Echo ‒hi memory reached‖ | Mail –s ‒memory reached‖ –c pradi@gmail.com pradeep@gmail.com

  Mail –s ‒memory reached‖ ‒body of mail‖ –c pradi@gmail.com pradeep@gmail.com

REALISTIC SCRIPT 2 – Service installation

- They wanted me to write a **script** to monitor the **_Tomcat‘ services** on the server, if any service or process is stopped accidently by someone, we should get email notification saying that **service is stopped** and also script should restrict the service.----------realistic script example----------

  Services=”ser1 serv2 serv3 ..... ―

  For service in $services

Do

  Ps -ef | grep “$service”

  If [$? -ne 0]

Then

  Mail -s “service is stopped and trying to restart

Sudo service $service restart

Fi

Done

REALISTIC SCRIPT 3 - Logs

Collect all the logs and deletes files older than X days

```
#!/bin/bash

prev_count=0

fpath=/var/log/app/app_log.*

find $fpath -type d -mtime +10  -exec ls -ltrh {} | rm -rf

count=$(cat /tmp/folder.out | wc -l)

if [ "$prev_count" -lt "$count" ] ; then

MESSAGE="/tmp/file1.out"

TO="daygeek@gmail.com"

echo "Application log folders are deleted older than 15 days" >> $MESSAGE

fi
```

GIT

- Used to track versions of files
- Type :- GIT, SVN, clearcase, TFS, perforce, CVS, Mercury
- Workspace
    - It ‘s a place where we edit modify project related files
- **GIT Arch**
    - Workspace → adding area→Commit area
- If I run git add files will be moved to staging area, it‘s a intermediate area,where we can save the changes.
- If I run git commit files will be added to git repository then we can track thefile version.
- **Git init** (used to create non-bare repo)
- Vi test1 (add some content)

Git status (used to check whether files are workspace, staging area or in git repo

- Git add test1 (this will move file from workspace to staging area)
- **Git status** (it will show changes to be committed)
- Git commit –m — — (this will move files from staging area to git repo)
- **Git status** (this will show working directory clean)
- **Git log** (used to check the repo history)
- Git log filename (specified filename)
- Git log -2 (last 2 commits)
- **Git checkout commitId** (used to switch to a previous version, used to switchto branch, also to switch to tags)
- **Git checkout master** (gives to the latest version)

Tags

- Tag is a name given to set of versions of files and directories. It indicates milestones of a project we can easily remember tags in the future. If I want good code in the future, we tag it.
- Command to list tags is,
    - Git tag
- To create a tag,
    - Git tag tag name (it tags the latest version of code by default)

Branching

- Is a parallel development, two teams will work on same peace of code on two different branches, later they can integrate the changes by merging
- **Git branch** – to list the branches
- Git branch branch name –to list branch names
- To switch to a branch name
    - Git checkout branch name

- **Git merge**
    - Is used to integrate two branches
        - Git merge branch name

Merge Conflicts

- when the same peace of code is changed on 2different branches, by 2 different developers, when we try to merge those two branches, then merging conflict will occur,
- To resolve this issue, I don ‘t know whose change should I take to merge, so I contact developers changes the code, personwho modified code of branch1 and branch2. Then they will decide and tell us whose changes should I take into merge.
- Then I take that change and I commit it. I get to know who modified the code on branch1 and branch2 using git log command.

- **to delete branch or tag**
    - git branch –d branch name
    - git tag –d tag name
- used to create branch2 and automatically checkout to that branch
    - git checkout –b branch2
- **Git rebase branch name**
    - Is nothing but a merge, it will add history also to the other branch.
    - It will not allow you to switch any other branch until unless you resolve merging conflicts.
        - Git rebase branch name
    - Latest commit id is called as HEAD,(is a internal tag)

Difference between merge and rebase

Merge: Merging takes the content of a source branch and combines them with target branch here only the target branch is updated in the process

Git Rebase: Git rebase is nothing but merge one branch will get added to tip of another branch

Git revert

- Used to undo the committed changes , history will not be removed we can track the reverted the changes in the git log
- Git revert HEAD – Undo latest commit
- Git revert CommitId – Undo particular commit

Git reset

- Used to undo the committed changes but history will be removed
- There are 3 kinds of reset.
  - Git reset --mixed or git reset HEAD
    - Is used to move files from staging area to workspace
  - Git reset --soft commitId
    - Is used to move files from git repo to staging area it reset to specified commit id and history will be removed.
  - Git reset --hard commit Id
    - Is used to remove the changes from git repo, workspace and also from staging area and commit id also removed.
    - Is as good as you are not committed the changes.

- Difference between **bashrc and bashrc_profile**
  - **.bashrc_Profile**
    - Executed for login shells
    - When you login via console, either sitting at machine or remote ssh, this is executed to configure your shell before the initial command prompt
    - Personal initialization file executed for login shells.
    - Used to customize the user configuration settings
  - **.bashrc**
    - Executed for interactive non-login shells
    - If you already logged into system and open a new terminal then this is executed before the window command prompt
    - Individual per-interactive shell startup file.

Git stash

- If I am working on one branch, if I get some critical work, which needs to be fixed on the other branch. I don‘t want to commit changes in other branch, as I not completed the work, I will do git stash and I switch to other branch I will fix the issue and I come back to the previous branch to continue my work. I need to run git stash pop.

- Git stash will saves files somewhere in temporary area. It willnot stores in working space, staging area or git repo.
    - Touch f1 f2 f3
    - Git add *
    - Git commit –m ―files added‖
    - Switch to other branch and check.
    - Git stash – then come back to other branch and try
    - Git status pop
    - Git reset HEAD

Git cherry-pick

- Git cherry-pick commitId
- Used to merge specific commit to a current branch.
- This command is useful to undo changes when any commit is accidentally made to the wrong branch. Then, you can switch to the correct branch and use this command to cherry-pick the commit.

Git conflict

- Is same as merge conflict
- Sometime you get merge conflicts when merging or pullingfrom a branch.
- Git will then tell you something like conflicts(content) mergeconflict in fakefile.
- It also tells you to fix conflicts and commit the changes.

Git bisect

- Use binary search to find the commit that introduced a bug.
- Git bisect <subcommand> <options>
- This command uses binary search algorithm to find whichcommits in your projects history introduced a bug.
- Can be used to find the commit that changed any property ofyour project.
    - There are 2 kinds of repository in git
- Bare repo git log
- Non-bare repo

Bare repo

- It acts as a central-repo, we can only push and pull the changesto the repository.
- –in base repository, you cant run any git operation on bare repo.

Non bare repo

- It‘s a local repo we can modify the files and push to a central repo we can run all the git commands. Command to create anon bare repo is git init
- Command to create a bare repo is
    - Git init –bare

Git clone

- Bringing the remote repo to local workspace for the first time called as git clone.

- Git clone user@servername: /home/path/central_repo/

Git push

- Moving the changes from workspace to remote repo or central repo
- Git push user@severname: /path.././central_repo

Git pull

- Bring the changes from remote repository and merges to local repo automatically.
- Git pull user@servername: /home/./path/central_repo

Git fetch

- Bring the changes from remote repo and stores it on a separate branch, you can review the changes and merge normally if it is required.
- Git fetch user@servername:/home.//./central_repo/
- Git pull = git fetch + git merge

Git Fetch	Git Pull
The Git fetch command only downloads new data from a remote repository.	Git pull updates the current HEAD branch with the latest changes from the remote server.
It does not integrate any of these new data into your working files.	Downloads new data and integrate it with the current working files.
Command - git fetch h origin git fetch --all	Tries to merge remote changes with your local ones. Command - git pull origin master

Git reflog : - to recover a deleted branch & git checkout command

git branch --merged - Returns the list of branches that have been merged into the current branch.

git branch --no-merged - Returns the list of branches that have not been merged.

A `head` is nothing but a reference to the last commit object of a branch

Makefile

- Abc.exe:big.o facr.o rev.o
- <TAB>gcc –o abc.exe big.o facr.o rev.o
- big.o:big.c
- <TAB>gcc –c big.c

Build

- facr.o:facr.c
- <TAB>gcc –c facr.c
- rev.o:rev.c
- <TAB>gcc –c rev.c

  - Is a executable or binary file, but not yet released to a testing team.
  - Is a untested build.
  - We deliver build to a testing team, not to a customer.
- Release
  - It is a tested build, which is ready to release to a customer.
- ANT
  - When I run ANT, it looks build.xml
- Maven
  - When I run maven, it looks pom.xml
- Hooks
  - .git→hooks→2 sub topics. 1)pre commit 2)post commit
  - You need to trigger some task or action before commit or after commit, we go for hooks.

Difference between git and cvs

- Git
  - Is a distributed version control system
  - It means whole repository will be there in the local workspace
  - If I want to go previous version of file, I can go directly in local workspace itself.
  - Git has many advanced features like rebase,fetch,stash,merge etc.
  - These advance features are not there in CVS.
- CVS
  - Is a centralized repo and also SVN also.
  - If I want to goto previous version of a file, I need to checkout from centralised_repo because initially it will have only one version of a file in local repo.

MAKE FILE

- Works on time stamp basis, if target time is less than the dependencies time it will re-generate the target, that means makefile will compare target time with its dependencies time.
- If target time is latest than its dependencies it will not re-generate the target.
- If there are thousand files, if I change 5 files, only 5 files will get compiled and incorporated at the build.

  Patch build
- It is a critical fix which needs to be deliver to a customer within few hours. Developers will change only required files, make will compile only changed files and changes will get incorporated to the build so patch build will take less time.

  Load build
- We compile source code from the scratch, before we start this build we delete all intermediate files (.o files), so that all files will get compile from scratch. So it take more time.
    - Which make
    - Which gcc
    - Mkdir srccode
    - Vi big3.c
    - Vi fact.c
    - Vi main.c
        - Main () { fact(); big3(); }
    - Vi makefile
        - Syntax
        - Example code
            - Abc:main.o fact.o big3.o
                - Gcc –o abc main.o fact.o big3.o
            - Main.o:main.c
                - Gcc –c main.c
            - Fact.o:fact.c
                - Gcc –c fact.c
            - Big3.o:big3.c
                - Gcc –c big3.c
        - make
        - The target file abc.exe will get created
        - You can executed ./abc to get output of all files.
        - You can add the files and related syntax in makefile.
        - Even you can give your own makefile name as pradeepmakefile,but while making, you should use,
            - Make –f pradeepmakefile and while creating makefile use vi pradeepmakefile.
        - If you want to remove .o and .exe
            - Rm –rf *.o *.exe
    - Vi makefile

- Syntax
- Example code
  - Abc:main.o fact.o big3.o
    - Gcc –o abc main.o fact.o big3.o
  - Main.o:main.c
    - Gcc –c main.c
  - Fact.o:fact.c
    - Gcc –c fact.c
  - Big3.o:big3.c
    - Gcc –c big3.c
  - Clean:
    - Rm –rf *.o
- While making, try
  - Make clean
    - It will remove all .o files.

- Build.sh

```
#!bin/bash

Git pull /../../../

If [ $? –ne 0 ];then

        Git clone /../../.

Fi

Cd /central_repo

Make targetname | tee build.log

If [ $? –eq 0 ];then

Echo ―‖ | mail –s ―build success‖ emailed

Scp targetname user:server:/.././/.

Else

        Mail –s ―build failure‖ emailed

Fi
```

Branching Stratergy

- Branches can be created for multiple reasons, here we create branches for releases.
    - **Development team will be going on development branch, once the code is ready for the first release, we create a release1 branch from devbranch and we make a release (we do build and release it) from this release1 branch.**
    - **Whatever the issues specific to this release will be foxed on release1 branch only. It will act as a maintenance branch for release1.**
    - **Simultaneously development will be going on dev branch for 2nd release. Once the code is ready for 2nd release before we create a branchfor 2nd release, we merge, release 1 branch to dev branch and then we create release2 branch for dev branch for 2nd release. So what ever the issues we have seen in 1sr release should not be visible in the 2nd release and so on**.
        - Build→build success→BVT or sanity test→sanity report→release note→test team
        - Build→build fail→dev team
    - BVT(build verification test) or sanity test.
    - Is a basic functionality of a build which should never break.
    - Sanity report.
        - Is a report related to testing and should checkbox for test.
    - Release note
        - Tag name and description and known issues.
    - Branch strategy example2

      We have 2 branches, one is dev branch another one is master or production branch, developers are allowed to commit changes on dev branch. Once developers commits the code on dev branch. We build it, we do sanity or BVT and we release to a testing team. We merge this code to production branch. If the build is failed after developer commits the changes, we don‘t merge the code from dev branch to production branch. We work with development team to fix the issue. So we always merge good code to productive branches so that producrion branch will have clean working branch always. If developerneeds latest good code they can pull it from production branch, but they cant push it to production branch, because git push is restricted.

Maven and ANT difference

Ant	Maven
Ant **doesn't has formal conventions**, so we need to provide information of the project structure in build.xml file.	Maven **has a convention** to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file.
Ant is **procedural**, you need to provide information about what to do and when to do through code. You need to provide order.	Maven is **declarative**, everything you define in the pom.xml file.
There is **no life cycle** in Ant.	There is **life cycle** in Maven.
It is **a tool** box.	It is **a framework**.
It is **mainly a build tool**.	It is **mainly a project management tool**.
The ant scripts are **not reusable**.	The maven plugins are **reusable**.
It is **less preferred** than Maven.	It is **more preferred** than Ant.

Maven and ANT?

- Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.
- Apache Ant is a Java based build tool from Apache Software Foundation. Apache Ant's build files are written in XML and they take advantage of being open standard, portable and easy to understand.

Git squash

- This method only allows you to squash the last X consecutive commits into a single commit. Also, if you have merged master into your branch along the way, you will have to manually merge your new (squashed) commit into master and resolve the merge conflicts.
- Use this method if you have not merged master into your branch, you plan to combine all commits into one, and you only changed one feature of the project; or, regardless of those conditions, you must use this method if you intend to open a pull request to merge your code.

Combining the commits

To squash the last **3** commits into one:

```
git reset --soft HEAD~3
git commit -m "New message for the combined commit"
```

Deployment

- Web applications
  - Apache tomcat
  - Websphere
  - Jboss
- Executed/binary code→
  - Dev environment
  - QA or SIT (system integration testing)→UAT or pre-production testing(user acceptance testing)→production testing.
- Once the build is generated (war file). We deploy war file to QA and then to UAT and finally we deploy to production, on each environment tomcat will be running, we deploy to Webapps folder under tomcat server.
- Deployment status
  - Stop services→move logs→take bakup of existing build→copy new build (war file)→start services.
  - If new build is not working, I will rollback the last build and use that and fix the issues in new build this process is called Rollback.
- Deployment script //(realistic script)

  #!/bin/bash

  Services=‖ser1 ser2 ser3‖

  For I in $services

  Do

  Sudo service $i STOP

  Done

  Mv *.log /temp

  Mv current.war current_date.war

  Scp filewar user@server:/../../

  For I in $services

  Do

  Sudo service $i START

  Done

- Above code is used to stop and start the services while deploying.

- At the time of deployment if ther is any error will try to debug and fix it within the deployment window.
- If I am not able to resolve the issue I will roll back it.
- Deployment used in shell scripts as → used to automated the process of running process or anything automatically.

- What is configure file for Tomcat?
    - In order to configure a Context within Tomcat a *Context Descriptor* is required. A Context Descriptor is simply an XML file that contains Tomcat related configuration for a Context, e.g naming resources or session manager configuration. In earlier versions of Tomcat the content of a Context Descriptor configuration was often stored within Tomcat's primary configuration file *server.xml*

XML Configuration Files

The two most important configuration files to get Tomcat up and running are called server.xml and web.xml. By default, these files are located at TOMCAT-HOME/conf/server.xml and TOMCAT-HOME/conf/web.xml

The server.xml file is Tomcat's main configuration file, and is responsible for specifying Tomcat's initial configuration on startup as well as defining the way and order in which Tomcat boots and builds.

The web.xml file is derived from the Servlet specification, and contains information used to deploy and configure the components of your web applications.

CATALINA.log?

- catalina.logis where the Tomcat engine writes log messages pertainingto Tomcat itself.

- Difference between web server and application server?
    - Application server are more heavy than web server in terms of resource utilization.
    - **Application Server** supports **distributed transaction and EJB**. While Web Serveronly supports Servlets and JSP.
    - A Web server exclusively handles HTTP requests, whereas an application server serves business logic to application programs through any number of protocols.

- Install tomcat and type on browser as publicIP:80 and install Jenkins and type on browser as publicIP:8080

```
jenkins.io------->>>>>download--->>>>>>>> Download
Jenkins 2.60.1 for: redhar/fedora/centos----->>>>>then u
ll get commands
-------------------------------------------------------------------------------
--------------------------------
 To use this repository, run the following command:

sudo wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-
stable/jenkins.io.key

If you've previously imported the key from Jenkins, the
"rpm --import" will fail because you already have a key.
Please ignore that and move on.

With that set up, the Jenkins package can be installed
with:

yum install Jenkins
```

```
or
-------------------------------------------------------------------------------
----------------------------------------
```

Jenkins

- Jenkins is a CI tool and automated framework is used to integrate development changes automatically without manual intervention.
  - Go to Jenkins main page
  - Click on new item
  - Give job name
  - Select job type (free style) & OK
  - Configure page
  - Description of job
  - SCM (source code management
  - Select git
  - Path of git central repo, add branches, credentials
- Jenkins will create its own workspace and pulls latest code to it.

Build Triggers

- Is used to trigger the job based on the time, based on commit or where otherjobs complete/success.
- **Build periodically**: we need to mention time like a crontab, it will trigger thejob based on the time., and also it is a scheduler crontab.
- **Poll SCM**: will trigger the job based on commits within the specified time job will get triggered based on commit.
- If there is a commit in between a hour, job ll get triggered else simply without commits it won‘t trigger.

Build step

- This section is used to compile and make a target file,and also compile and generate a binary. We can execute shell or invoke aNT.
- If I use execute shell, I can use shell script which internally calls makefile or I can use invoke ANT to compile java source code.

Post build action

- This is used to post build activities after build a project, like deploying build,running test cases or send a email notification or copying the build to shared path.

Master slave

- Master: server on which Jenkins has installed is called master.
- Slave: server on which we run the jobs from the master is called slave.
- Why master-slave or slave machines required?
  - To distribute the load to a different server from the master we go for master-slave.
  - We need to run specific job on specific environment for example, java project/java job we need to compile it on server which has java environment we go for master slave/slave machines.

- How to create slave machines/nodes?
    - Jenkins
    - Manage nodes
    - New node
    - Give node name and description
    - Number of executors
    - Label name
    - Root directory
    - Launch method (SSH), HOST, credentials.
    - Number of executors means, we can run specific number of jobs simultaneously on that server depending on capacity of server we specify number of executors.
    - If number of executors is _2‘ menas , we can run 2jobs from Jenkins simultaneously.
- How to run this slave on the new job or already created job or on the server
    - Got to configure page of job
    - Configure
    - Restrict this project can run
    - Mention the node name/slave name
- We go for labels to run job on available server, we need to add all server names to a same label. We need to mention the label name in the job instead of node name.
- This will work in round robin passion.
- How to add ec2 instance on Jenkins?

    - First, go to EC2 and sign up for the service. Once you've installed the plugin, you navigate to the main "Manage Jenkins" > "Configure System" page, and scroll down near the bottom to the "Cloud" section. There, you click the "Add a new cloud" button, and select the "Amazon EC2" option. This will display the UI for configuring the EC2 plugin. Then enter the Access Key and Secret Access Key which act like a username/password (see IAM section). Because of the way EC2 works, you also need to have an RSA private key that the cloud has the other half for, to permit sshing into the instances that are started. If you have already been using EC2 and have your own key, you can paste it here. Otherwise, you can have Jenkins generate one. If you let Jenkins generate one, you should save this private key in your file system as well, as you'll need this to interactively logon to EC2 instances.
    - Once you have put in your Access Key and Secret Access Key, select a region for the cloud (not shown in screenshot). You may define only one cloud for each region, and the regions offered in the UI will show only the regions that you don't already have clouds defined for them.
    - Use "Test Connection" button to verify that Jenkins can successfully talk to EC2. If you are using UEC you need to click on Advanced and fill out the endpoint details for your cluster.
    - http://www.bogotobogo.com/DevOps/Jenkins/Jenkins_on_EC2_setting_up_master_slaves.php

- Http port 80
- https port 443

How to install plugins

Manage Jenkins→manage plugins→update available installed advanced→in advanced just upload a plugin file .hpi frmat.

File will be get download from Jenkins updater center.

Paramaeterized plugins

- is used to give or pass parameters or input to a job at the run time. We havedeployment job, we run that job based on request from dev team or development teamon test team, at the running the deployment job we need to select build number and server name, these 2 parameters will be taken as input to a job.

Build pipeline (upstream and downstream)

- used to trigger one job after the other and we can pass parameter from one job to another using upstream and downstream plgins.
- In the build now page
- Configure page for build job
  - Upstream-_____
  - Downstream-deploy job
- Configure page for deploy job
  - Upstream-build job
  - Downstream-test job
- Configure page for build job
  - Upstream-deploy job
  - Downstream-_____
- Build job will get triggered by commit, once the build job is success it will automatically trigger deployment job.
- If deployment job is success it will automatically trigger testing jobs, we can trigger job serially using upstream and downstream plugins.

Gearman plugins

- It is a high availability plugin, if jenkin master goes down,jenkin will moves go down if I install this plugins.
- German plugin will allow you to configure other server details, if the master goes down other serverwill act as a master. Both the server will be in a sink frequently.
- Other server will allows have complete Jenkins backup. So jenkin will never go down.

What is the most challenging task you done it?

- When I installed Jenkins for 1st time on the server, after a month server got crashed due

to some hardware issue, server didn‘t come up at all. I lost the Jenkins builds were stopped from the day, we couldn‘t recover Jenkins. I had to resetup Jenkins on the other server.

- Now I searched for the high availability plugin, I got to know about GEARMAN PLUGIN. I installed gearman plugin in Jenkins and I configured other server details in it. If the master goes down, other server will act as a master. So Jenkins will never go down now.

How do you take Jenkins backup?

/var/lib/Jenkins

- We create separate git_repo for Jenkins and can push Jenkins configuration files to git_repo. Then we can clone it, whenever we need it. ((challenege)).

Jenkins safe restart

- It will allow to current running jobs to complete but it will not allow new jobs to be trigger, and it will restart the Jenkins, once the current running jobs are completed.
  - http://localhost:8080/saferestart

Jenkins security

- How to give security permissions to access Jenkins?
- How do you add new used in Jenkins?
  - We use matrix based security, it will allow to give required permission for a user by clicking the checkboxes.
  - Manage Jenkins→configure global security properties→click on enable security(these Jenkins own users db)→allow users to signup→in authorization section→matrix based security will be checked→you can add users and give permission security→then save it.
- How do you add environment vaiables?
  - Goto manage Jenkins→configure system→global env→add environment variables.
    - Name:var
    - Value:1
  - We use configure system to configure smtp server allows.
- Git diff current_cmtid previous_cmtid(what it modified)
- Git show commitid (list all specified modified files in specified commits)

CI CD Continous Integration, Deployment, Delivery

Continous Integration

- Is a continuous integration, integrating the development changes continuously without manual intervention.
- As soon as development team commits the codeto a git, build job get triggered and build will get generated automatically.

CD (Continuous Deployment)

- Is a continuous deployment, deploying the build to testing environment automatically or less manual effort.
- Each change from development team is build and deployed to test environment automatically.

CD (Continuous Delivery)

- Is a continuous delivery, releasing deploying tested build to a production environment as quickly as possible.

Best practices of SCM (source code management)

- Keep workspace clean, mean delete temporary or unwanted files from workspace.
- Go for branching, if it is necessary.
- Tag the source code, whenever if it is necessary or the good code.
- Document the tasks which you do.

Installing tomcat n Jenkins and mainly configuring user n ports

1) Sudo /usr/share/tomcat/conf/tomcat.conf
    a. Tomcat-user="tomcat"→"root"
        i. Conf/server.xml
        ii. 8080—80
        iii. 8443-443
    b. Netstat –ntl
        i. Running ports for tomcat
2) Sudo yum install tomcat
    a. Sudo vi /usr/share/tomcat/conf/tomcat.conf
        i. Change java_opts path
    b. Sudo yum install tomcat-webapps tomcat-admin-webapps
    c. Sudo vi .usr/share/tomcat/conf/tomcat-users.xml
        i. <tomcat-users>
            1. <user username="admin" password="passwd" roles="manager-gui,admin-gui"/>
        ii. </tomcat-users>
    d. Sudo service tomcat start
    e. Sudo service tomcat restart
    f. Chkconfig tomcat on
    g. Service start
    h. IP:8080
    i. Apache home page
3) Install Jenkins and config it
    a. Sudo vi /etc/sysconf/Jenkins
        i. Change jenkinsuser=root
        ii. Then jenkinsport=8081
            1. Bcoz both tomcat n jenkins will take same ports that is 8080.
        iii. Sudo service Jenkins start

    - Bugzilla
    - JIRA
- We deploy the build sat early mrg and sat nights.
- BAT (build automation tool)
    - We were using it before as of now we were migrated to Jenkins.
- Docker
    - Is a container tool, is used to run build in virtual env. All env will be there in this container.
- Virtualization and hypervisor.
- Install docker and check for interview ques.
- Install nagios(monitoring tool) and check the interview ques.
- Build steps
    - SCM
    - Build
    - Code coverage(check the code quality) -→ install it.
        - Tools
            - Sonarqube—install>>>>>>
            - Code collaborator
    - Deploy
- Groovy
    - I know , but I have not used,
    - We have some automation tools to use for builds, so we not used groovy.
- Jenkins version- 2.7 (used about 3yrs)
- tomcat- 8
- issues – memory issues, tomcat issues,build issues due to compilation n some small errors.
- VPC (virtual private cloud)
    - Is a network
    - 10.0.0.0, 1st two are network next 2 are subnet.
    - 10.0.0.0/32
- Private network- used in internal world/env/company.
- Public network-used in external network
- Subnet (is a sub network, is used in segregation)

How do you setup passwordless connection between 2 servers?

Server1.....→ server2

Ssh-keygen
I will be getting private and public key
From server1 copy the public key of server1 to server2 in .ssh/authorized_keys
Then u can access server2 from server1 passwordless connection.

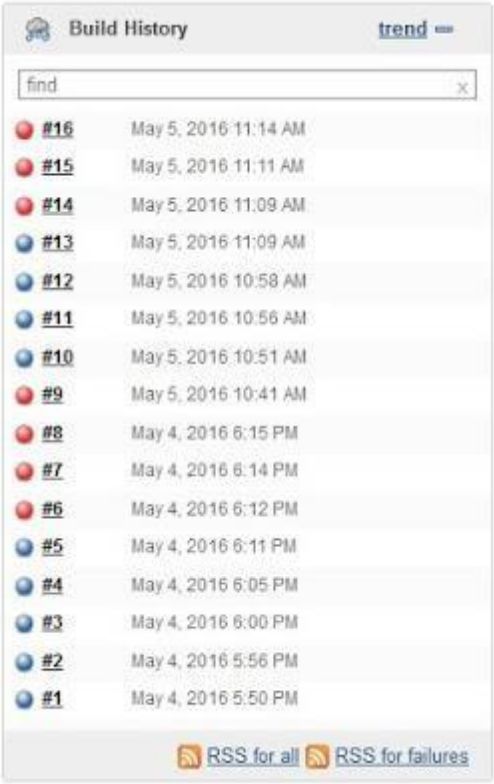
We can use winscp plugin or software to connect to sever sfrom password less connection.

We need to run ssh keygen on server1 it generates the key we need to copy public key to authorized keys files under .ssh folder. So we can connect s2 from s1 without password.

How to clean and reset Jenkins build history

Posted on Wednesday, May 04, 2016 by nayana

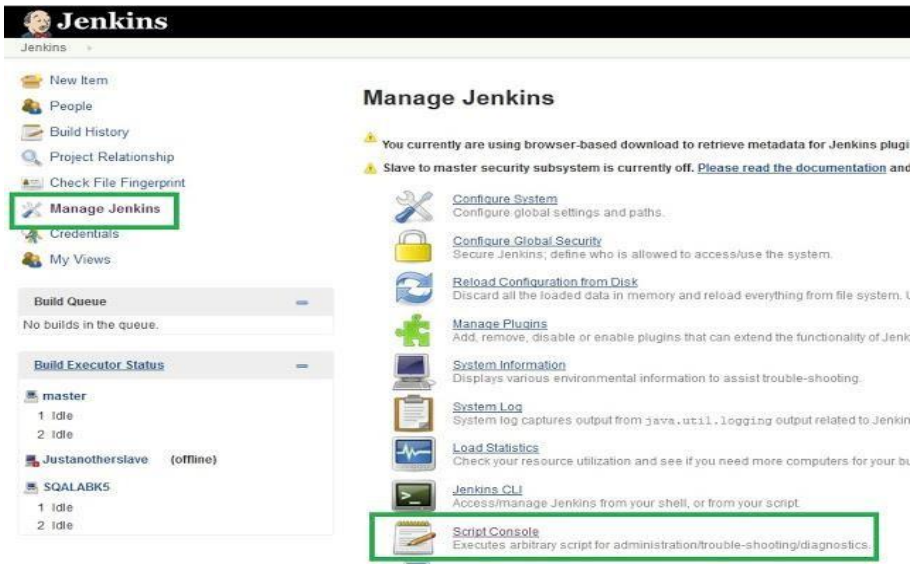
When you working with Jenkins while building your solution, definitely you have to run your project multiple times. It will result to a long Build History list that looks ugly and unnecessary for your production environment.



If you need to clean this Build History, and reset the build number, you can run a simple script in Jenkins Script Console. Here are the steps to do so.

Go to `Jenkins Script Console`

Go to your Jenkins home page > Manage Jenkins > Script Console



Run the script to clean and reset

Copy and paste this script to your Console Script text area and change the "copy_folder" to project name that you need to clean the history. Then click the "Run button".

```
def jobName = "copy_folder"
def job = Jenkins.instance.getItem(jobName)
job.getBuilds().each { it.delete() }
job.nextBuildNumber = 1
job.save()
```

Now check your Build History

AWS (Amazon Web Series)

IAM (Identity and Access Management)

- IAM provides access to accounts services where we can manage User, Roles, Groups & Policy password policy.
- It applies globally to all AWS regions.

Users: we create users and assign necessary permissions to them in the form of policies.

Groups: We can create groups for ex. Dev QA etc. and attach policies at the group level.

Policy: A policy is a set of permission attached to user, group or roles

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": *,
      "Resource": "*"
    },
```

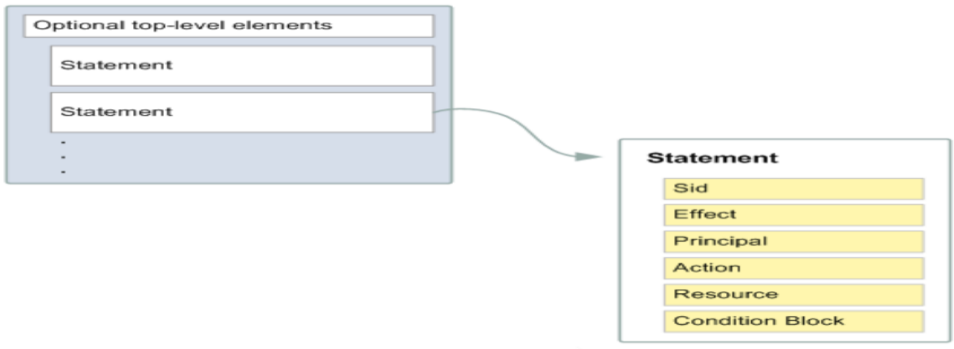
Policy types:

1. **Identity Based Policy:** Applicable on users, groups of users, and roles
   - AWS Managed policy:
   - Custom Managed Policy:
   - Inline Policy
2. **Resource Based policy:** Attach to a resource such as an Amazon S3 bucket
3. **Session based Policy:** create a temporary session for a role or federated user

Imp Notes:

More than one policy can be attached to a user or a group at the same time.
Policies can't be attached directly to resources like EC2 instance, S3 bucket etc.,

Basic Policy structure:



Effect: Can take only two value allow or deny
Principal: who is assuming the policy
Resource: on whom you are assuming the policy

Ex:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FirstStatement",
      "Effect": "Allow",
      "Action": ["iam:ChangePassword"],
      "Resource": "*"
    },
    {
      "Sid": "SecondStatement",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
     ]
}
```

Q: Can we attach multiple policies to user group or a role.
Yes, we attach.

Roles

A role is a set of permissions that grant access to actions and resources in AWS.

- Roles comes between services, like ec2 wants to access S3 or non-AWS user (hybrid account) should access AWS Resources.
- Policies can't be attached to aws resources hence roles come into picture.
- EC2 can be attached one role at a time.
- Can we assign multiple roles to a EC2 instance? No, we can't. we can assign only single role to EC2 instance.

Q: If an ec2 instance is not able to access s3 bucket what could be the reason
A Role needs to be attached with proper policy defined.

Assume Role:

Returns a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to. These temporary credentials consist of an access key ID, a secret access key, and a security token.

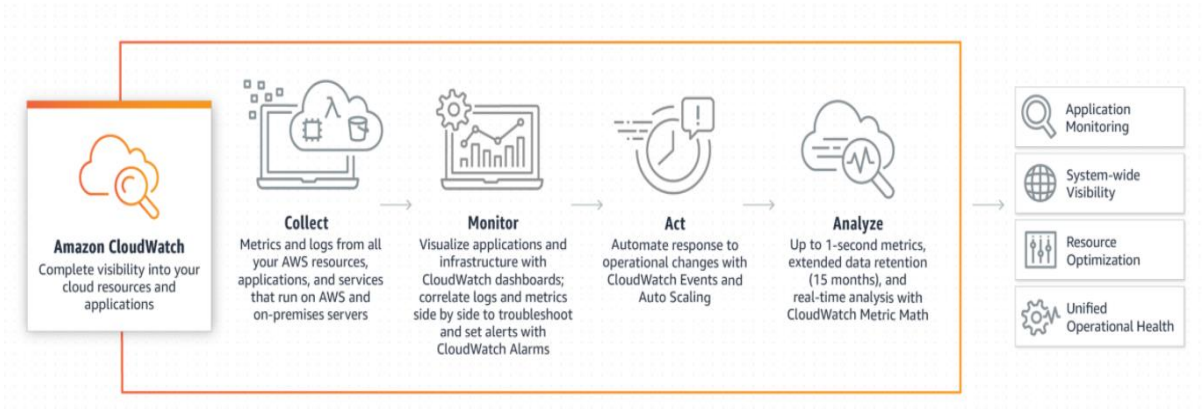
Example:
1. create a IAM user
2. Add him ec2 full access policy
3. Try to list the bucket (aws s3api list-buckets) - you can't list the bucket because the user is not having the permission to list bucket (access denied)
4. Grant the user to assume a role
   Create a role
   Attach a policy s3 full access policy
   get inside the role ---- trust Relationship change it to user instead of ec2
   AWS: "ARN OF IAM USER"
5. aws sts assume-role --role-arn <enter the role arn> --role-session-name s3-access-example --duration-seconds 3600
6. copy the accesskey secretkey and seeison token
7 set AWS_ACCESS_KEY_ID=<enter the copied access key>
  set AWS_SECRET_ACCESS_KEY=<enter the copied session key>
  set AWS_SESSION_TOKEN=<sessiontoken>
8. aws s3api list-buckets
9. Remove the env variable

```
set AWS_ACCESS_KEY_ID=
set AWS_SECRET_ACCESS_KEY=
set AWS_SESSION_TOKEN=
```

The AWS Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).

CloudWatch

Amazon Cloud Watch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time.



Default metrics of EC2 instance: Network usage CPU Usage

Metrics:
Metrics are data about the performance of your systems
Basic monitoring: which polls for every 5 minutes
Detailed monitoring: which polls for every 1 minute.

Alarm:
CloudWatch Alarms feature allows you to watch CloudWatch metrics and to receive notifications when the metrics fall outside of the levels (high or low thresholds) that you configure
Ex:
If CPU utilization goes beyond the static threshold alarm goes to alarm state
Three states in CW Alarm:
Alarm state
Insufficient
OK state

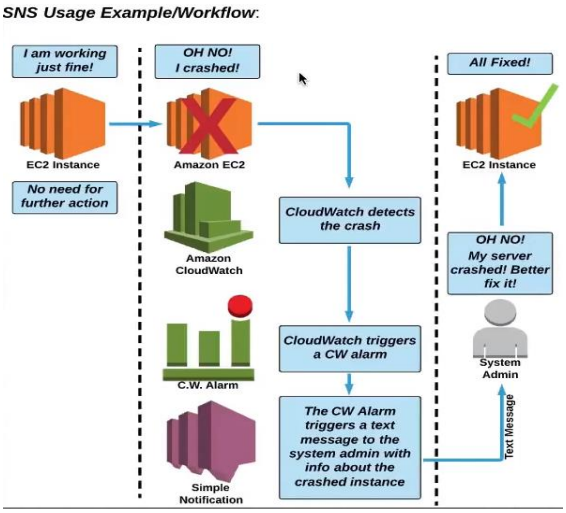
Events: An Event indicates change in AWS environment
Event Resource: Which resource you want to monitor
Event target: to alert the event change through notifications

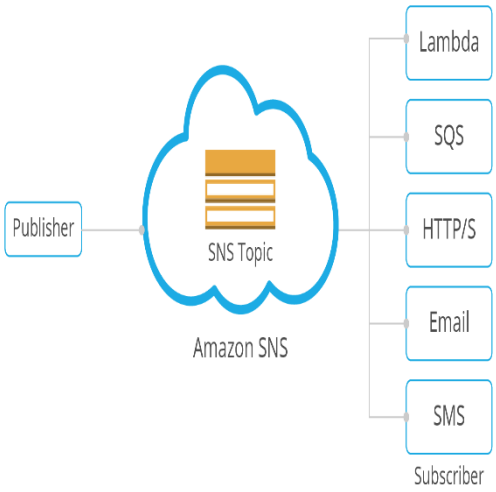
Logs:
CloudWatch Logs enables you to centralize the logs from all your systems, applications, and AWS

Simple Notification Service

- Amazon Simple Notification Service is a notification service provided as part of Amazon Web Service.
- It provides a low-cost infrastructure for the mass delivery of messages, predominantly to mobile users

Topic:
An Amazon SNS topic is a logical access point that acts as a communication channel





CloudTrail

- Auditing tool records all AWS account activity.
- Any action taken by users, roles and AWS services are recorded to cloud trial.
- Cloud trial events are kept for 90 days in event history
- You can create a trail of your own store the event history in s3 bucket.
- There are two types of events

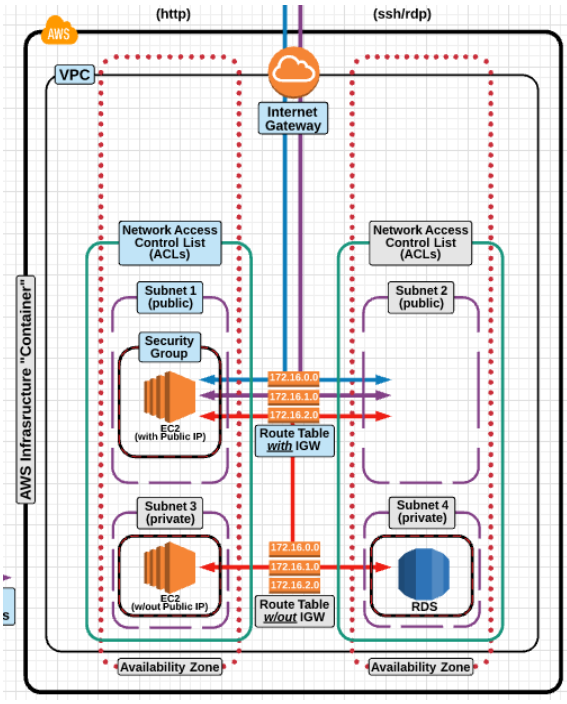
  Management events: Management operations performed on AWS

  Data events: currently supported S3 and Lambda: You can now record all API actions on S3 Objects and receive detailed information such as the AWS account of the caller, IAM user role of the caller, time of the API call, IP address of the API, and other details

  Insights events: AWS CloudTrail Insights helps AWS users identify and respond to unusual activity associated with write API calls by continuously analyzing CloudTrail management events.

Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define like EC2 instance Databases.



CIDR:(Classes interdomain routing)

Classless Inter-Domain Routing is a method for allocating IP addresses and for IP routing.

Ex: The IPv4 block 192.168.100.0/22 represents the 1024 IPv4 addresses from 192.168.100.0 to 192.168.103.255.

I.e., 2^ (32-22) = 2^10 = 1024 IPv4 addresses.

VPC design:

VPC CIDR = 10.180.0.0/16 means we have 65536 IPv4 address			
IPV4 Address range is 10.180.0.0 ---- 10.180.255.255			
Public subnet 1	Public Subnet 2	Private Subnet 1	Private Subnet 2
10.180.0.0/24	10.180.1.0/24	10.180.2.0/24	10.180.3.0/24
256 IPV4 address	256 IPV4 address	256 IPV4 address	256 IPV4 address
10.180.0.0-10.180.0.255	10.180.1.0-10.180.1.255	10.180.2.0– 10.180.2.155	10.180.3.0 – 10.180.3.255

SUBNET

- A **subnet** is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet.
- Some IP addresses are reserved they are
  - 10.180.0.0 Network address
  - 10.180.0.1 VPC Router

- 10.180.0.2 DNS server (**DNS**. (Domain Name System) The Internet's system for converting alphabetic names into numeric IP addresses)
- 10.180.0.3 Future use
- 10.180.0.255 N/W Broadcast address

- VPC spans multiple Availability zones.
- Subnets must be associated with route table
    - A public subnet has a route to internet
    - A private subnet doesn't have route to internet. It creates higher level of security.
- You can use a network address translation (NAT) gateway to enable instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating a connection with those instances.

VPC Quota or VPC limitations

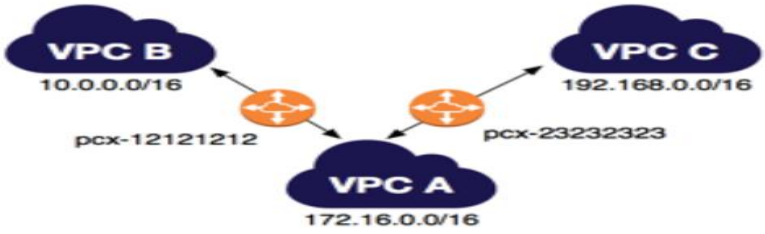
- 5 VPC per region
- 5 IGW per region
- Subnet per VPC 200
- IPv4 CIDR blocks per VPC 4
- Elastic IP addresses per Region 5
- Internet gateways per Region 5
- NAT gateways per Availability Zone 5
- Network ACLs per VPC 200
- Rules per network ACL 200

VPC Peering:

- A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses.
- Instances in either VPC can communicate with each other as if they are within the same network.
- You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account.
- The VPCs can be in different regions (also known as an inter-region VPC peering connection).

Conditions:

- CIDR block shouldn't overlap
- Transitive peering relationships are not supported. i.e here VPC B cannot connect with VPC C.
- If the VPCs are in different regions, inter-region data transfer costs apply.
- You cannot have more than one VPC peering connection between the same two VPCs at the same time.



NACL:

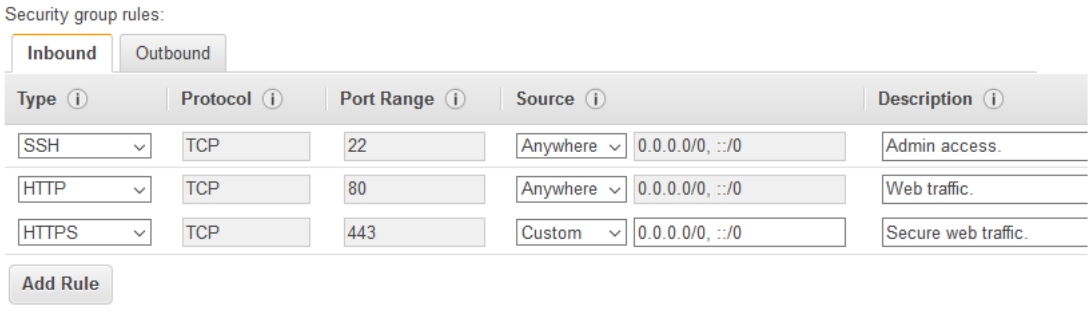
1. A network access control list (ACL) is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. (Firewall at subnet level)

    1. Inbound means – incoming (Ingress)
    2. Outbound means – outgoing (egress)
    3. Always explicit deny take precedence over allow

Rule #	Type	Protocol	Port Range	Source	Allow / Deny
80	SSH (22)	TCP (6)	22	0.0.0.0/0	ALLOW
90	HTTP (80)	TCP (6)	80	0.0.0.0/0	ALLOW
*	ALL Traffic	ALL	ALL	0.0.0.0/0	DENY

Security Group:

A **security group** acts as a virtual firewall for your instance to control inbound and outbound traffic.



Nat gateway:

A NAT gateway is a Network Address Translation (NAT) service. You can use a NAT gateway so that instances in a private subnet can connect to services outside your VPC, but external services cannot initiate a connection with those instances.

Bastion host:

A **bastion host** is a **server** whose purpose is to provide access to a private network from an external network

VPN:

Is mainly used to establish a secure and private tunnel from you network or device to aws network

- Aws site-to-site vpn: enables you to securely connect your on-premises network to your vpc.
- AWS client vpn : enables you to securely connect users to AWS or on premises network.

Do we have another way we can connect to the resources in a private subnet?

We can setup a vpn server in the public subnet and configure it to connect to resources residing the private subnet

Elastic Cloud Compute (EC2)

An **EC2 instance** is a virtual server in Amazon's Elastic Compute Cloud (**EC2**)

EC2 instance types:

EC2 Instance Type:

- Instances Types describe the "hardware" components that an EC2 instance will run on:
  - Compute power (processor/vCPU)
  - Memory (ram)
  - Storage Options/optimization (hard drive)
  - Network Performance (bandwidth)
- Instance Types are grouped into families and types that you can choose from that have different purposes:
  - General Purpose (T2, M5, and M4):
    - T2 - Burstable performance, good for many general purposes
    - M4/M5 - Small or mid-size databases, data processing, enterprise applications
  - Compute Optimized - (C4 and C5):
    - High performance web servers, science/engineering apps, ad serving
  - Memory Optimized - (X1e, X1, and R4):
    - High performance databases, in-memory databases, large data processing engines
  - Accelerated Computing - (P3, P2, G3, F1):
    - P2/P3 - Machine/Deep learning, high performance databases, server-size GPU compute workloads
    - G3 - 3D visualizations and rendering, application streaming, video encoding, server-side graphics workloads
    - F1 - Genomics research, financial analytics, big data, and security
  - Storage Optimized - (H1, I3, D2):
    - D2/H1 - MapReduce, HDFS, network file systems, or data processing applications
    - I3 - NoSQL databases (Cassandra/MongoDB/Redis), data warehouses, Elasticsearch

Note that the instance families and types are the 'current' generation as of April 2018.

Purchasing options:

EC2 Purchasing Options:

On-Demand:

- On-demand purchasing lets you choose any ***instance type*** and provision/terminate it at any time
- Is the ***most expensive*** purchasing option
- Is the ***most flexible*** purchasing option
- You are only charged when the instance is ***running*** (and billed by the second)

Reserved Instances (RI):

- Reserved purchasing allows you to purchase an instance for a ***set time period*** of one or three years
- This allows for a ***significant price discount*** over using on-demand
- You can select to pay upfront, partial upfront, or none upfront
- Once you buy a reserved instance, you own it for the selected time period and are ***responsible for the entire price*** - regardless of how often you use it
- Purchases of AZ-specific RIs provide capacity reservation in that AZ. Regional RI purchases do not - so it is theoretically possible AWS will run out of capacity

Spot Instances:

- Spot pricing is a way for you to ***"bid"*** on an instance type, and only pay for and use that instance when the spot price is ***equal to or below*** your "bid" price
- This option allows Amazon to sell the use of ***unused instances***, for short amounts of time, at a ***substantial discount***
- ***Spot prices fluctuate*** based on supply and demand in the spot marketplace
- You are ***charged per second (with conditions)***
- When you have an active bid, an instance is ***provisioned for you when the spot price is equal to or less than you bid price***
- A provisioned instances ***automatically terminate when the spot price is greater than your bid price***.
- Bid on unused EC2 instances for "non production applications"

Dedicated Hosts:

- A dedicated physical machine that you have full control over. This can help save money on license fees and meet certain regulatory compliances

EBS (Elastic block storage)

Provides block level storage volumes for use with EC2 instances

EC2 Elastic Block Store (EBS) Basics:

- EBS volumes are ***persistent***, meaning that they can live beyond the life of the EC2 instance they are attached to
- EBS backed volumes are ***network attached storage***, meaning they can be attached/detached to or from various EC2 instances
- However, they can only be attached to ONE EC2 instance at a time
- EBS volumes have the benefit of being backed up into a ***snapshot*** - which can later be restored into a new EBS volume
- By default, EBS volumes are replicated within the Availability Zone
- EBS volumes are usually mounted to the file system at /dev/sda1 or /dev/xvda

EBS Types:

EC2 Elastic Block Store Volumes:

General Purpose SSD:

- Use for dev/test environments and smaller DB instances
- Performance of 3 IOPS/GB of storage size (burstable with baseline performance)
- Volume size of 1GB to 16TB
- Considerations when using T2 instances with SSD root volumes (burstable vs. baseline performance)

Provisioned IOPS SSD:

- Used for mission critical applications that require sustained IOPS performance
- Large database workloads
- Volume size of 4GB to 16TB
- Performs at provisioned level and can provision up to 32,000 IOPS per volume

Throughput Optimized HDD and Cold HDD:

- Cheaper than SSD options, also less performant
- Cold HDD - Designed for less-frequent access
- Volume size of 500GB - 16 TB
- Cannot be a boot volume

EBS Magnetic (Previous Generation):

- Low storage cost
- Used for workloads where performance is not important or data is infrequently accessed
- Volume size of Min 1GB Max 1 TB

EFS:

Amazon Elastic file system is a regional service storing data within and across multiple Availability Zones (AZs) for high availability and durability

Difference between EBS v/s EFS v/s S3

AMAZON EBS	AMAZON EFS	AMAZON S3
Hardly scalable	Scalable	Scalable
Block Storage	Object storage	Object Storage
Faster than S3 and EFS	Faster than S3, slower than EBS	Slower than EBS and EFS
Accessible only via the given EC2 Machine	Accessible via several EC2 machines and AWS services	Can be publicly accessible
Is meant to be EC2 drive	Good for shareable applications and workloads	Good for storing backups
File System interface	Web and file system interface	Web interface

Snapshot EBS

- You can back up the data on your Amazon EBS volumes to Amazon S3 by taking point-in-time snapshots. Snapshots are *incremental* backups

Snapshots are stored in S3

- Launch two ec2 instance in different az's(instance1 & instance2)
- Create EBS volume and attach it to instance1
- The volumes are attached to instance1 you can verify it by logging into instance1 and executing "lsblk" command, but it's not mounted you can verify it through by running command "df -TH"
- Mount the volume to instance1

- Format the disk with ext4: “mkfs -t ext4 /dev/xvdf”
- Create a directory in root: 1. “cd /” 2. “mkdir /mnt/mydisk”
- Mount the disk: “mount /dev/xvdf /mnt/mydisk”
- you can verify that disk is mounted by running “df -TH” command.

- Create some files
- Take a snapshot
- Unmount the disk
  - umount /mnt/mydisk
  - Detach the volume from ec2 instance.
  - delete the volume
- Create a new volume from snapshot
- Attach the volume to newly created instance2.
- Mount the volume to instance2
  - Create a directory in root: 1. cd / 2. mkdir /mnt/mydisk
  - mount /dev/xvdf /mnt/mydisk

Data life cycle Manager:

You can use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of snapshots taken to back up your Amazon EBS volumes

Amazon machine image (AMI):

- An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance.
- You can launch multiple instances from a single AMI when you need multiple instances with the same configuration.

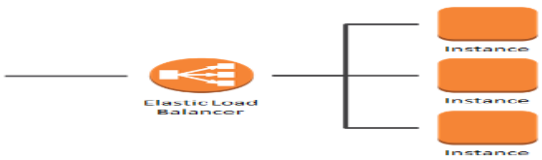
Difference between Snapshot and AMI

An EBS snapshot is a backup of a single EBS volume. The EBS snapshot contains all the data stored on the EBS volume at the time the EBS snapshot was created.

An AMI image is a backup of an entire EC2 instance. Associated with an AMI image is EBS snapshots. Those EBS snapshots are the backups of the individual EBS volumes attached to the EC2 instance at the time the AMI image was created

Elastic load Balancer (ELB):

Manage and control the flow of inbound request to group of targets by distributing the requests evenly across the targets. The targets may be EC2 instances lambda or containers.



Types of Load balancer:

Application load balancer:

- Used mainly for web application running http and https protocols.
- Operates at request level.

Network Load balancer:

- Ultra-high Performance at very low latency.
- Operates at connection level, routing traffic to targets with in VPC.
- Can handle millions of requests per second.

Classic load Balancer:

- Used for applications that were built in existing EC2 classic env.
- Operates both at connection & request level.

Example: Classic load balancer

- Spin up an EC2 instance1 in another availability zone (az1) with http port open in Security group
- Add the below script and launch the instance.

```
#!/bin/bash
yum update -y
yum install httpd -y
echo '<h1> Response from server-1 </h1>' > /var/www/html/index.html
service httpd start
chkconfig httpd on
```

- Spin up one more EC2 instance1 in another availability zone (az2) with http port open in Security group
- Add the below script and launch the instance.

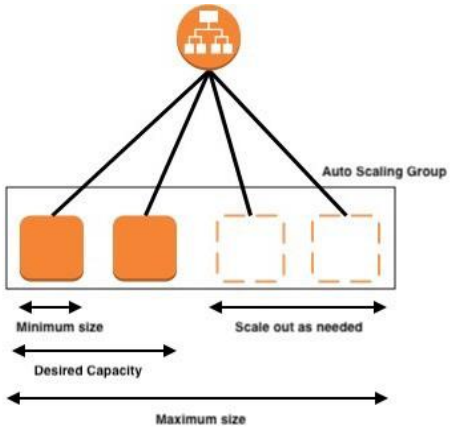
```
#!/bin/bash
yum update -y
yum install httpd -y
echo '<h1> Response from server-2 </h1>' > /var/www/html/index.html
```

service httpd start

chkconfig httpd on

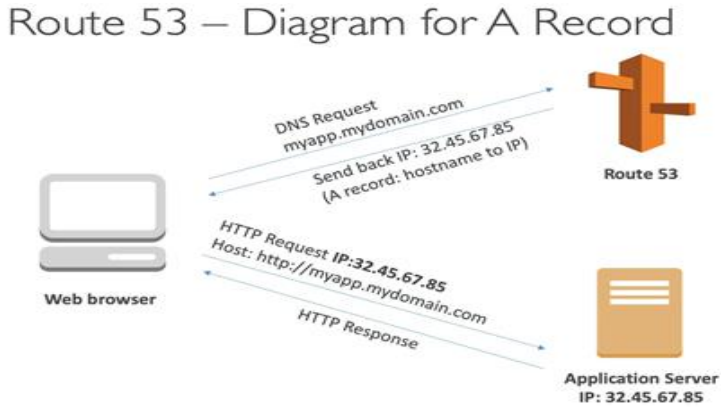
Autoscaling

AWS Auto Scaling monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.



Route53:

Amazon Route 53 is a highly available and scalable Domain Name Server (DNS) web service, where we can point IP address to domain name or point host name to another host name.



A Record:

Maps IP address to domain name ex: 10.180.0.0 to myapp.mydomain.com

CNAME Record:

Maps hostname to another host name: us-east.2.elb.amazonaws.com to myapp.mydomain.com

Alias Record:

points a host name to AWS Resource ex: myapp.mydomain.com to us-east.2.elb.amazonaws.com

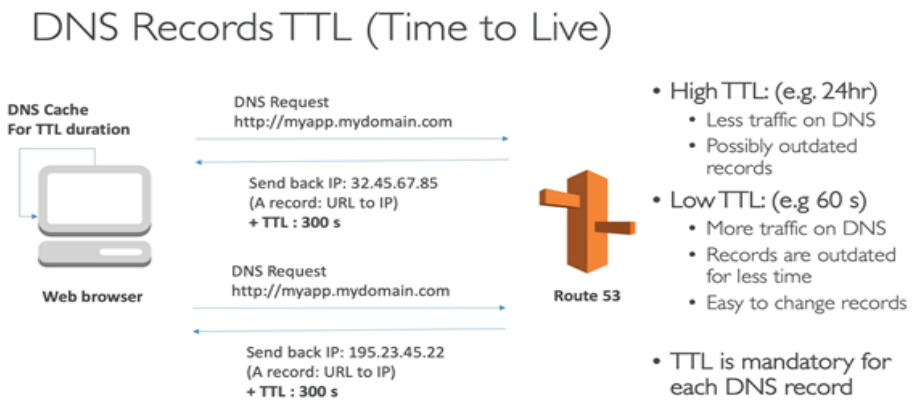
Latency Routing Policy:

Use when you have resources in multiple AWS Regions, and you want to route traffic to the

region that provides the best latency.

Weighted Routing Policy:

Use to route traffic to multiple resources in proportions that you specify.



What happens when you type a URL in the browser and press enter?

https://medium.com/@maneesha.wijesinghe1/what-happens-when-you-type-an-url-in-the-browser-and-press-enter-bb0aa2449c1a

Simple storage service (S3)

Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web.

Single operation upload:

- It's a traditional upload where you will upload the object in one part
- A single operation upload can upload the file up to 5GB in size.

Upload object in parts:

- Using multipart upload, you can upload the large objects up to 5TB.
- You can use multipart upload for the objects from 5MB to 5TB in size.

Rules for bucket naming:

- Bucket names must be between 3 and 63 characters long.
- Bucket names can consist only of lowercase letters, numbers, dots . and hyphens -.
- Bucket names must begin and end with a letter or number.
- Bucket names must not be formatted as an IP address (for example, 192.168.5.4).
- Bucket names can't begin with xn-- (for buckets created after February 2020).

Limitation of S3 bucket:

- Only 100 buckets can be created per account.
- Can hold unlimited objects

S3 Storage classes:

- Standard:
  - Designed for general- and all-purpose storage
  - Default storage option
  - 99.999999999% object durability
  - 99.99% object availability
  - Most expensive storage class.
- Reduced Redundancy storage
  - Designed for non-critical objects
  - 99.99% object durability
  - 99.99% object availability
  - Less expensive than standard
- Infrequent access
  - Designed for less frequently accessed objects.
  - 99.999999999% object durability
  - 99.99% object availability

  Less expensive than reduced redundancy storage

- Glacier
  - Designed for long term archival storage
  - May take several hours to retrieve the objects from this storage
  - Cheapest s3 storage class

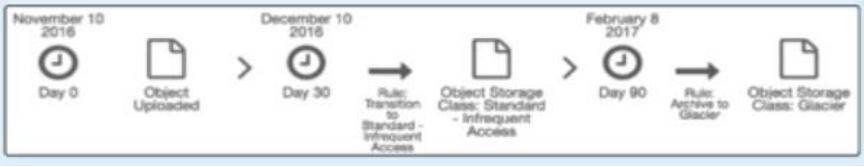
S3 Life cycle policy:

An object lifecycle policy is a set of rules that automate the migration of the object storage class to different storage class

By default, lifecycle policies are disabled for a bucket

Example:

- I have a work file that I am going to access every day for the next 30 days
- After 30 days, I may only need to access that file once a week for the 60 next days
- After which (90 days total) I will probably never access the file again but want to keep it just in case



S3 Encryption:

Two ways of protecting information with S3

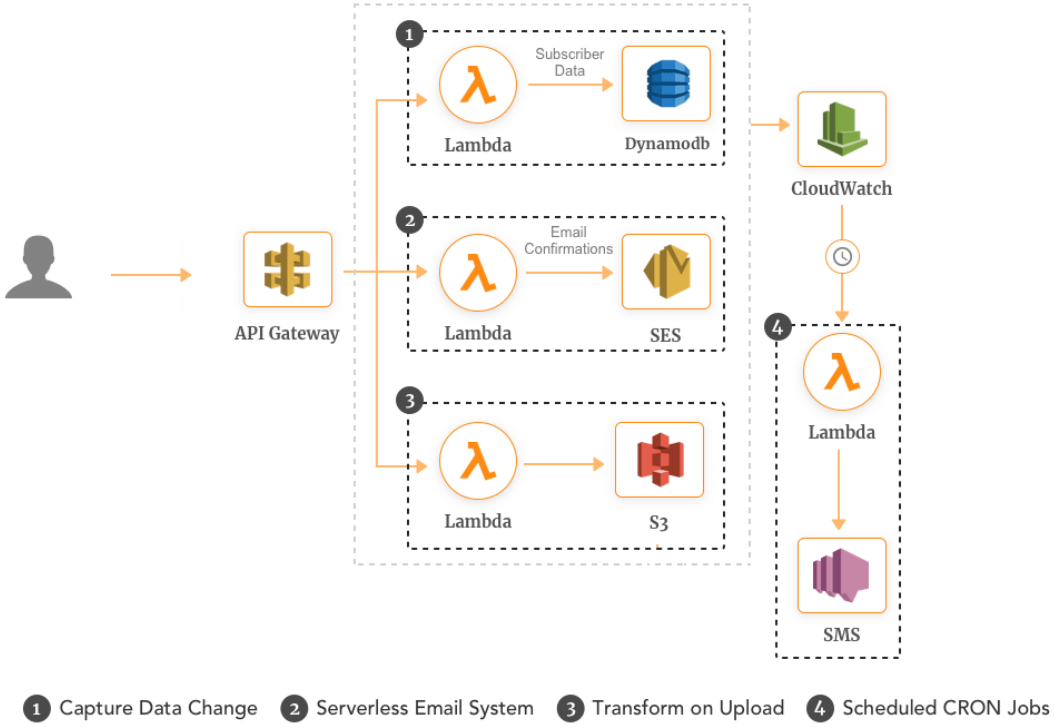
1. **Server side/$_{\text{At rest}}$:**

2. **In-transit/Client-side encryption:**

Lambda

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume.

Just upload your code and Lambda takes care of everything required to run and scale your code with high availability



- Manage your virtual functions not really caring about the servers
- Run on demand
- Scaling is automated

Billing:

Pay per request first one million requests is free $0.20 per one million request.

compute time 0.00001667 for every GB-seconds used.

AWS Lambda Languages:

NodeJS, Python, Python3, Gr00vy, java, csharp, Scala and GO

AWS Lambda Integration

Kinesis, API Gateway, DynamoDB, AWS S3, CloudWatch Events, CloudWatch logs, SNS and Incognito

Key Management Service (KMS)

AWS Key Management Service (**AWS KMS**) is a managed service that makes it easy for you

to create and control customer master keys (CMKs), the encryption keys used to encrypt your data.

AWS secrets

AWS Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources. The service enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle.

Relational Database Service (RDS)

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud.

Which relational database engines does Amazon RDS support?
Amazon RDS database engines:

- Amazon Aurora
- PostgreSQL
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server

Encryption in RDS:
Encryption at rest is supported for

- Amazon Aurora
- PostgreSQL
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server

Q: Can we enable encryption on exciting DB

Encrypting existing DBs is not supported. To do this, you'll need to create a new encrypted instance and migrate data to it. The encryption key can be stored in KMS.

Q: **Which is the non-relational database supported in AWS**

Amazon DynamoDB is the NoSQL database supported by AWS

Classic load Balancer VS Application Load Balancer

The Classic Load Balancer is a connection-based balancer where requests are forwarded by the load balancer without "looking into" any of these requests. They just get forwarded to the backend section

The Application Load Balancer operates at the request level only. If you're dealing with HTTP requests, which you are for your web application, we can use this It also supports advanced features like host and path-based routing

We can create target groups in order to route to traffic to the respective paths

Cognito

- It Mainly provides authentication authorization and user management for your application
- It provides a managed user pool to mange identity for the application

Cognito provides user flows:

- Signup
- Signin
- Forgot or change password
- Multifactor authentication
- Email and phone verification

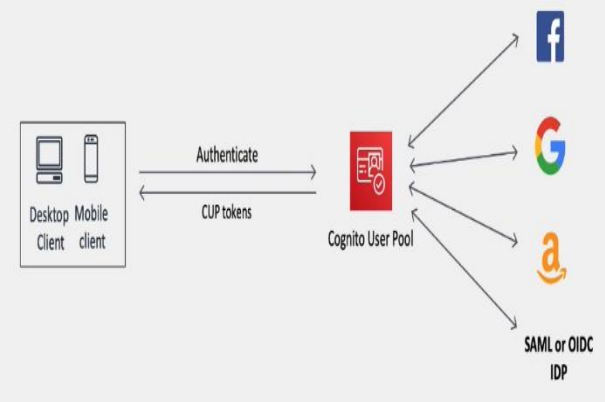
It also provides software development kit to your mobile or web application, and also provide lambda triggers in order to customize any of these user flows with you own business logic

It also provides a built-in hosted UI for these user flows

Social identity can be integrated
Facebook
Google
Amazon
SAML

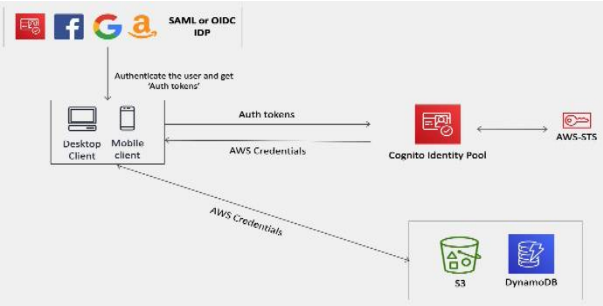
After authentication the user the Cognito provides the best practice way of accessing the AWS resources securely from the app by providing temporary credentials

User Pool:



- User pools acts as mediator between your app and external
  social identity providers
- you can add multiple identity providers as you need.
- The user pool manages the token exchange with each of the
  providers and gives your app standard user pool tokens of
  same format

Identity poll:



- Where you exchange the authentication token to get temporary aws credentials which you can use to access the resources directly from the app
- These can be used independently of each other or used together

Difference between user pool and identity pool

AWS Cognito User Pools is there to authenticate users for your applications

Say you were creating a new web or mobile app and you were thinking about how to handle user registration, authentication, and account recovery, you don't need to implement user authentication inside your application, rather you can integrate AWS Cognito User Pools, which will manage user sign-up, sign-in, password policies.

AWS Cognito Identity pool:

- This is a service which was designed to authorize your users to use the various AWS services. The source of these users could be a Cognito User Pool or even Facebook or Google.

In other words, Identity Pools are used to assign IAM roles to users (who had been authenticated through a separate Identity Provider which could be Cognito User Pools or Social logins (e.g; Gmail, Facebook & etc.)). Because these users are assigned an IAM role, they each have their own set of IAM permissions, allowing them to access AWS resources directly.

So, the difference is

- AWS Cognito User Pools: Granting access to a application
- AWS Cognito Identity Pools: Granting access to amazon service

Difference between IAM and Cognito

AWS IAM gives securely and control access to AWS services and resources for your users
AWS Cognito It Mainly provides authentication authorization and user management for your application

IAAS PAAS SAAS DIFFERENCE

Basis Of	IAAS	PAAS	SAAS
Stands for	Infrastructure as a services.	Platform as a services.	Software as a services.
Uses	IAAS is used by network architects.	PAAS is used by developer.	SAAS is used by end user.
Access	IAAS give access to the resources like virtual machines and virtual storage.	PAAS give access to run time environment to deployment and development tools for application.	SAAS give access to the end user.