Devops Demo

*Devops cloud:*

*Devops:*

*development& operations:*

*developers: programming language-c, c++, java,.net.*

*data base (db) -pql ,pl sql ,msbi*

*net working*

*admin*

*deveops cloud*

*operating system:*

*Operating System : (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.*

*An operating system is software that enables applications to interact with a computer's hardware. The software that contains the core components of the operating system is called the kernel.*

*linux: we need to learn commands*

*we need server*

*cloud:*

*amazion web services-ec2 server*

*devops: tools*

*source code management-src code,git,github,git tab.*

*build tool- maven-artifacts- images*

*continous- integrations tool -automation-agile metholodologies*

*ci cd-jenkins- groovy scripting*

*cd- contionuous deployment*

*containerization tool -docker*

*container orchestration tool -kubernet -gcp*

*(google cloud platform)*

*configuration management tool-ansible*

*cloud:*

*public cloud*

*aws*

*azure*

*gcp*

*alibaba cloud*

*private cloud :*

*oracle cloud*

*ibm*

*aws: ec2*

*networking vpc*

*back up -ebs*

*load balancer*

*auto scaling group-100 servers*

*30 servers- 70 servers*

*cloud watch -monitroning tool*

*sns- e- mail notification*

*iam -identity access management*

*ami- amazon machine images*

*eip*

*code commit*

*code build*

*s3- simple storage services*

*terraform:*

*laac*

*aws free account:*

*pre-requisites*

*e- mail id*

*card- debit/credit 2/- rs*

*4- different jobs*

*1-deveops engineer*

*devops & cloud engineer*

*linux, devops & aws engineer*

*sre- site reliablity engineer*

*testing- 4yers-8-9 laks*

*devops-11-12 laks*

*45 days*

*monday- friday*

*day to day recoring*

*operating system*

*windows*

*linux*

*unix,centos, Ubuntu, susulinux*

*linux pre requisities*

*server*

*cloud*

*aws cloud*

*we are launching the server*

*aws account*

*we need to create the server*

*aws account*

*we need to create free account*

*10 hrs -740 hrs*

*Aws account-pre – requisites*

*Email id*

*Credit/debit card-2-/ rps*

*Aws account creation*

*5 setps*

*Amazon aws login*

*Create on aws account*

*Root user email –id*

*Aws account name*

*Verify email address*

*Sign into an existing aws account*

*devops demo*

*Devops cloud:*

*devops:*

*development& operations:*

*developers: programming language-c,c++,java,.net.*

*data base (db)-pql,plsql,msbi'*

*net working*

*admin*

*deveops cloud*

*operating system*

*Operating System : (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.*

*An operating system is software that enables applications to interact with a computer's hardware. The software that contains the core components of the operating system is called the kernel.*

*linux: we need to learn commands*

*we need server*

*cloud:*

*amazion web services-ec2 server*

*devops: tools*

*source code management-src code,git,github,git tab.*

*build tool- maven-artifacts- images*

*continous- integrations tool -automation-agile metholodologies*

*ci cd-jenkins- groovy scripting*

*cd- contionuous deployment*

*containerization tool -docker*

*container orchestration tool -kubernet -gcp*

*(google cloud platform)*

*configuration management tool-ansible*

*cloud:*

*public cloud*

*aws*

*azure*

*gcp*

*alibaba cloud*

*private cloud :*

*oracle cloud*

*ibm*

*aws: ec2*

*networking vpc*

*back up -ebs*

*load balancer*

*auto scaling group-100 servers*

*30 servers- 70 servers*

*cloud watch -monitroning tool*

*sns- e- mail notification*

*iam -identity access management*

*ami- amazon machine images*

*eip*

*code commit*

*code build*

*s3- simple storage services*

*terraform:*

*laac*

*aws free account:*

*pre-requisites*

*e- mail id*

*card- debit/credit 2/- rs*

*4- different jobs*

*1-deveops engineer*

*devops & cloud engineer*

*linux, devops & aws engineer*

*sre- site reliablity engineer*

*testing- 4yers-8-9 laks*

*devops-11-12 laks*

*45 days*

*monday- friday*

*day to day recoring*

*AWS Account 1-class*

*===========*

*1.free account - 750hrs - 1 yrs*

*Debit/credit - 2/-rs*

*AWS*

*2. Putty software(puttygen also)*

*Optional - Puttygen*

*Linux - OS*

*=========*

*Server*

*Linux - command line*

*Linux Server*

*==========*

*Launch server*

*EC2 instnace/service/server*

*Elastic Compute cloud - EC2 - VM(Virtual Machine) server*

*EC2 - Instance*

*==============*

*7 steps*

*linux20april.pem*

*How to convert the privatekey*

*========================*

*puttygen - public we need private key(ppkfile)*

*how to connect the server*

*========================*

*linux-server(ec2-instnace) - public IP Address*

*21/04*

*=========*

*Linux*

*========*

*linux OS, Free open source os*

*Unix - paid one, os*

*Linux*

*========*

*CentOS -*

*Ubuntu*

*SUSE Linux*

*Windows*

*=========*

*what is sudo -i?*

*=============*

*Sudo(Superuser do) allows a system administrator to give certailn users(or group of user) the ability to rum domr(or all) commands as root while logging all commnads and arguments*

*what is Root?*

*=============*

*Primary Hierarchy root and root directory of the entire file system hirerachy*

*1. Each single file and directory starts from the root directory*

*2. Only root user has the right to write under this directory*

*3. /root is toot user's home directory, which is not same as /*

*Linux*

*========*

*FSH - File System Hierarchy*

*=========*

*The Linux FSH defines the directory structure*

*Folder/ Directory*

*pwd - present working directory*

*mkdir linux*

*ls*

*cd linux-demo*

*file -creation*

*================*

*touch file1*

*Note :*

*=====*

*In side directory you may have 100 files - directories are difined with "blue color"*

*In side file there no directories - files are defined with "white color"*

*Delete/Remove directory/folder*

*=============================*

*rm -rf dirname*

*or*

*rm -r firectory (type - y)*

*Delete/Remove files*

*============*

*rm -f filename*

*how to create multiple directories*

*=============================*

*mkdir d1 d2 d3*

*How to create parent directories*

*====================*

*To create parent directories using the mkdir command pass -p option*

*suppose the directory path /foo/bar/baz is to created even if /foo/bar directory is not existing*

*Permissions and rights(ownership)*

*===========================*

*A linux is a Multi-user Operating system which can be accessed by many user simultaneously.*

*Hence this raises security conecrns as am going to restrict on the perticular user*

*2 levels*

*1. Ownership*

*2. Pemission*

*How to change the ownership for file level*

*========================================*

*The chown command change ownership of files and directories in a linux filesystem*

*to change the owner of a files/directory*

*syntax: chown <ownername>(shobha) <filename>(file.txt)*

*How to change the ownership for group of file*

*=================================*

*"chown" command is also used to the chnage group name of a file, like "chgrp" command*

*chown :shobhs file.txt*

*file name - file.txt*

*1.chown ec2-user*

*-rw-r--r-- 1*

*-rw- r-- r-- 1 (file - level)*

*user group others*

*Group:*

*=====*

*chown commands is also used to change group name of a file like, "chgrp" commands*

*synatx: chown <groupname> <filename>*

*chown :ec2-user file.txt*

*To recursively change ownership of directories and their contents*

*========================*

*synatx: chown -R root >filename>*

*mkdir test*

*d -rw- r-- r-- 1*

*directory users group others*

*2.chown -R root filename*

*-R = To recursively change ownership of directories and their contents*

*mkdir test*

*chown -R ec2-user:ec2-user test*

*How to chnage the Permission*

*========================*

*All file system object on Unix-like system have three main types of permissions*

*read, write and execute access.*

*permissions are based upon 3 possible class(users, group, others)*

*user, the usergroup and all system users*

*syntax: ls -lha*

*1. r: Read =4*

*2. w: Write =2*

*3. x: Execution =1*

*Permissions*

*==============*

*Read - r - - 4*

*Write - w - - 2*

*Execution - x - - 1*

*====*

*7*

*Synatx: ls -lha*

*Example*

*===========*

*d rwx r-x r-x*

*dir (user) (Group) (Others)*

*task:*

*======*

*setting 442(rrw) permissions for the file of file.txt(filename)*

*rwx r-x r-x*

*r r w*

*file.txt rrw*

*chmod 442 file.txt*

*How to change the permissions & ownership recursively*

*==============================*

*to chnage the file access permissions you need to sue the chmod command. it has -R or -recursive option thar chnage files and directories recursively*

*mkdir dir5*

*cd dir5*

*touch file6 file7*

*dir5 - rwx*

*d rwx r-x r-x*

*dir (user) (Group) (Others)*

*7 0 0*

*700 --(user)--(group)--(others)*

*file2 r r w*

*r-- r-- w--*

*4 4 2*

*chmod 442 file2*

*permission with dire - anusha*

*d(rx) user(rx) group(rx) others*

*5 5 5*

*chmod -R 555 anusha*

*if you want give the permissions directory level(you should not entire inside the directory)*

*Linux/Unix Command*

*=========*

*1. File commands*

*ls - directory listing/ files listing*

*ls -al - hidden files to be displyed*

*cd dir - change dir to dir*

*cd - change to home*

*rm file - delete file*

*rm -f filename - force remove file*

*rm -r dir -*

*rm -rf dir*

*cp file1 file2 - copy the file1 file2*

*cp -r dir1 dir2 - copy the dir1 dir2*

*mv file file2 - rename or move file1 dile2*

*touch file -*

*cat file - to see content inside file*

*more file - output the content of file*

*head file - output the first 10 lines of file*

*tail file - output the last 10 lines of file*

*tail -f file - output the content of files as it grows, starting with the last 10 lines*

*2. Process managemnet*

*ps - display your current active process*

*top - display all running process*

*kill pid - kill process id pid*

*bg - list stopped or backgroundjobs; resume a stopped job in the background*

*fg - bring the most recent job into foreground*

*fg n - bring job n into foreground*

*3. File Permissions*

*chmod octal file - chnage the permisiions of file of octal, which can be found separately for user, groups and world by adding*

*4 - read(r)*

*2 - write(w)*

*1 - Execution(x)*

*Exmaple:*

*-*

*chmod 777*

*chmod 755*

*4. SSH*

*ssh user@host - connect to host as user*

*ssh -p port uset@host - connect to host on port port as user*

*5. System info*

*date*

*cal*

*uptime*

*w - display who is the online*

*whoami*

*finger user - diplay information about user*

*uname -a - shows kernal information*

*cat /proc/cpuinfo - cpu information*

*cat /proc/meminfo - memory information*

*man - manual for command*

*df - disk usage*

*du - shows ditecrory space usage*

*free - shows memory and swap usage*

*whereis app -*

*which app*

*VIM Editor:*

*==========*

*Vim is powerfull text editor used in CLI(Command Line Interface).*

*Linux uses a lot of configuration files.*

*if you edit them and vim is a great tool do so, aletrnatively VIM is a command line editor*

*vim flie.txt*

*50 lines*

*50 lines*

*vi editor*

*vim editor*

*It has 3 modes:*

*----*

*1. Command Mode*

*2. Insert Mode -i*

*3. Extended Mode -*

*Note: when you open the vim editor , it will be in the command mode by default.*

*Insert Mode - i*

*Extended mode*

*----*

*save - esc:wq!*

*withoutsave- esc:q!*

*Command Mode*

*+++++++++++++++++*

*gg - to go to the beginning of the page*

*Shift+g - go to the end of the page*

*w - to move the cursor forward, word by word*

*b - to move the cursor backword, word by word*

*nw - to move the cursor forward to n words(5W)*

*nb - to move the cursor backend to n words(5W)*

*u - to undo last change(word)*

*Ctrl+u - to undo previous chnages (entire line)*

*Ctrl+r - to redo changes*

*yy - to copy a line*

*nyy - to copy n lines (5yy or 4yy)*

*p - to past line below the cursor position*

*P - to past line above the cursor position*

*dw - delete the word letter by letter(like Backspace)*

*x - to delete the word letter by letter( delete key)*

*dd - to delete entire line*

*ndd - to delete n no.of positions from cursor positions(5dd)*

*/ - To search a word in the file*

*Insert Mode(edit mode):*

*+++++++++++++++*

*i - To begin Insert Mode*

*I - To insert beginning of line*

*a - To Append the next word's letter*

*A - To append at the end of the line*

*o - To insert a new line below the cursor positions*

*O - To insert a new line Above the cursor positions*

*Extended Mode(colon Mode)*

*++++++++++++*

*Esc+:w - To save the changes*

*Esc+:q! - to quit(without save)*

*Esc+:wq - to save and quit*

*Esc+:w! - To save forcefully*

*Esc+:wq! - To save and quite forcefully*

*Esc+:x - to save and quit*

*Esc+:X - to give password to the file and remove password*

*Esc+:20(n) - to go to line no 20 or n*

*Esc+:se nu - to set the line numbers to the file*

*Esc+:se nonu - To remove the set line number*

*Monitoring:*

*+++++++++++++++++++*

*- when you needs to see the running process on your linux in real time, you can have top as your tool that.*

*- top also displayes other info besides the running process*

*top*

*tasks - shows the numbe of process and their current state*

*%cpu- show CPU uilizaion details, user process, system process running, system Idle cpu.*

*KIB Mem- VIRT(Virtual Memroy Size(KiB), the total amount of virtual memory used by the process*

*RES- Residenet Memory Size*

*SHR - Shared Memory Size*

*Swap Memory in Linux:*

*===================*

*Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space.*

*While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM.*

*zombie process in linux:*

*========================*

*Zombie processes in Linux are sometimes also referred to as defunct or dead processes.*

*They're processes that have completed their execution, but their entries are not removed from the process table.*

*What is buff cache Linux?*

*========================*

*buff/cache is memory used by the Linux kernel buffers and page cache.*

*You force clear buff/cache using the 'echo' command but observe it gradually increase again to 85% memory consumption.*

*Shutting down the SAP NW application and/or database server. does not show buff/cache release any memory.*

*PID - ProcessID*

*USER - Effective UserID*

*PR - Dynamic Prority*

*NI - nice Value, also known as base priority*

*VIRT - Vitrual size of the task. This includes size of process executable binary, the data area and all the loaded shared libraries*

*RES - The size of RAM Currently consumed by the task, Swapped out portion of the task is not included*

*SHR - Shared memory ares clould be shared b/w two or more task, this fileds reflects the shared area.*

*S - Task status*

*%CPU - the %CPU time dedicated to run the task since the last top's screen update*

*%MEM - the %Mem of RAM currently consumed by the task*

*TIME + - The total CPU time and task has been used since it started "+"sign means is displayed wih hundredth of second granularity.*

*BY default, TIME/TIME+ desont account CPU Time used by the task's dead children*

*COMMAND- Showing program names.*

*Kill userdetails*

*-------------*

*kill -9 PID*

*Intracting with Top*

*================*

*M- Sort by memroy usage*

*P- sORT BY CPU Usage*

*T- Sort by cumulative Time*

*z - display the colour*

*k - kill a process - kill -9 userid*

*q - quit*

*r- to renice a process*

*h- help*

*How do I find zombies on Linux?*

*===================*

*Image result for zombie in linux*

*How to find zombie processes?*

*D = uninterruptible sleep.*

*I = idle.*

*R = running.*

*S = sleeping.*

*T = stopped by job control signal.*

*t = stopped by debugger during trace.*

*Z = zombie.*

*Ping commnad*

*===========*

*packet internet Groper*

*ICMP - internet Control Messages*

*TTL - Time To Live*

*ping -c2 google.com*

*c - here "c" represents no.of hops you want as output.*

*incase we gave 2. hence we arr getting hopeand then its getting exited*

*vmstat command:*

*================*

*vmstat (virtual memory stastics)as a valuable monitoring utility, which also provide information about block IO and CP activity in adition to memory*

*info about block IO and CPU*

*Shortcut File commands*

*===============*

*wc -l : Prints the number of lines in a file.(filename)*

*wc -w : prints the number of words in a file.*

*wc -c : Displays the count of bytes in a file.*

*wc -m : prints the count of characters from a file.*

*wc -L : prints only the length of the longest line in a file.*

*what is /etc*

*===========*

*This is the nerve center of linux system, its contains all system related configuration files in here or in this sub-directories.*

*A configuration file isdfines as a local files used to control the operations of a program.*

*Software Management*

*====================*

*its important role in installaling and configuring the softwares.*

*2 ways we can approach*

*1.rpm - Redhat package manager*

*2.yum - yellow dog update manager - (centos)*

*Ubuntu - apt-get*

*RPM:*

*========*

*RPM is a package managing system(collection of tools to manage software packages) - ex- git, maven, jenkins, python, java*

*RPM is powerful software mangement tools for installing, uninstall, verifying, querying and updating software packages.*

*RPM straight forward program to perform the above s/w management tasks.*

*synatx:*

*rpm -qa (where q stands for query, and a stands fo all)*

*rpm -qa python*

*installion syntax:*

*====*

*rpm -ivh package name*

*list out of package in linux server*

*==============*

*rpm -qa |grep -i python*

*grep - global regular expression print*

*yum*

*=====*

*yellow dog update manager*

*yum install git -y*

*Repo:*

*clean*

*list*

*Networking:*

*==============*

*It is a connection between two or more machines to communicate with ach other*

*Basic Requirement for Networking are*

*==================*

*1. NIC(Network Interface Controller or Card)*

*2. Media*

*3. Topology*

*4. Protocol*

*5. IP Address*

*1. NIC(Network Interface Controller or Card)*

*==================*

*NIC is sued for network adaptor, LAN Adaptor*

*2. Media*

*=========*

*Media is the medium via which two diff computors NIC card will be connected*

*CIDR - Class Inter Domain Routing*

*3. Topology:*

*==============*

*topology is the scheme or design in which computer in the n/w will be connected*

*4. Protocol:*

*==============*

*a n/w protovol defines rules & conventions for communication b/w n/w devices*

*TCP - Transimission control protocol -80*

*UDP - user datafram protocol - 80*

*HTTP - Hypertext transfer protocol - 80/8080*

*HTTPS -Hypertext transfer protocol Service - 443*

*Networking Commands:*

*======================*

*Linux Networking Commands*

*ifconfig.*

*ip.*

*traceroute.*

*tracepath.*

*ping.*

*netstat.*

*ss.*

*dig.*

*Netwprk Configuration & Trouble Shooting*

*-----------------------*

*Networking - it is a connection b/w two or more machines to communicate with each other*

*#/etc/sysconfig/network*

*======================*

*is a file which keep the information abot the hostname assigned to the system*

*if you want to change the hostname permannetly.*

*#/etc/sysconfig/network-scripts/*

*ls /etc/sysconfig/network-scripts/*

*cat /etc/sysconfig/network:*

*---------------------*

*is a file whiich keep the information about the hostname assigned to the system.*

*if you want to change the hostname permenently , you need to change the hostname in this file.*

*checking the IP Address*

*===========*

*ifconfig -a*

*DNS*

*----*

*Domain Name spacific*

*nslookup ipaddress/hostname*

*nslookup*

*What is the difference between sudo and su command?*

*===================================*

*The main difference between Su and Sudo is that the Su command can interchange between superuser and root user if executed without prior additional options while the Sudo command provides single root privileges.*

*Su demands the password of the root account while Sudo demands the password of the current user account*

*what is inode in linux?*

*=======================*

*What is an inode? By definition, an inode is an index node. It serves as a unique identifier for a specific piece of metadata on a given filesystem.*

*Each piece of metadata describes what we think of as a file. That's right, inodes operate on each filesystem, independent of the others*

*Why is inode used?*

*===================*

*Inodes keep track of all the files on a Linux system. Except for the file name and the actual content of the file, inodes save everything else.*

*It's like a file-based data structure that holds metadata about all of the files in the system.*

*what is softlink in linux?*

*======================*

*Soft links are similar to shortcuts, and can point to another file or directory in any file system.*

*Hard links are also shortcuts for files and folders, but a hard link cannot be created for a folder or file in a different file system.*

*These can be created in the following way on Linux and Mac operating systems.*

*what is hard link in linux?*

*=======================*

*Every file on the Linux filesystem starts with a single hard link. The link is between the filename and the actual data stored on the filesystem*

*what is hardening in linux?*

*========================*

*What is OS Hardening? Operating system (OS) hardening, a type of system hardening, is the process of implementing security measures and patching for operating systems, such as Windows, Linux, or Apple OS X, with the objective of protecting sensitive computing systems.*

*what is the difference between grep and find?*

*=====================================*

*The main difference between the two is that grep is used to search for a particular string in a file whereas find is used to locate files in a directory, etc. also you might want to check out the two commands by typing 'man find' and 'man grep'*

*what is the Linux Volumes*

*============================*

*A physical volume will be seen as /dev/sda, /dev/sdb; a physical disk that is detected by Linux.*

*If we have two physical disks of 1TB each, we can create a volume group of almost 2TB amongst the two. From the volume group, we can create three logical volumes each of any-size not exceeding the total volume group size*

*What is the use of LVM in Linux?*

*==============================*

*LVM is used for the following purposes: Creating single logical volumes of multiple physical volumes or entire hard disks (somewhat similar to RAID 0, but more similar to JBOD), allowing for dynamic volume resizing.*

*what is the diffrence between centos & Ubuntu*

*========================================*

*https://www.google.com/search?q=what+is+the+difference+between+centos+and+ubuntu&sxsrf=ALiCzsYDtdhgVqbmPVL1ZxcAJhNWc0\_oZA:1668047189762&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjdkduByKL7AhUDSmwGHUF9BLcQ\_AUoAnoECAEQBA&biw=1422&bih=641&dpr=1.35#imgrc=yYhEibw6OM0I9M*

*FileSystem Management:*

*=======================*

*Linux plays a crucial role in full filling the system managemnet*

*2 types*

*1. folder*

*2. filesystem - part c. d - LVM - Logical volume management*

*working with Filesystem:*

*===========================*

*2 types of approaches*

*1. standard partition*

*2. Logical Volume Management - LVM*

*standard partition:*

*=========================*

*i have 1000GB Disk*

*250 GB*

*3 Types*

*1. Primary Partition(P)*

*2. Extended Partition(E)*

*3. Logial Partition(L)*

*2. Logical Volume Management - LVM*

*GIT Commnds:*

*yum install git*

*git init*

*git --version*

*git status*

*git add*

*git commit -m*

*git push*

*git commit –a –m “adding file”*

*git config --global user.name “lingareddy”*

*git config --global user.email "lokeshlingareddy@gmail.com"*

*git log*

*git log --oneline*

*git show ‘commitID’*

*git diff 'filename'*

*git rm filename*

*git commit -m "deletion of file1"*

*git rm --cached file2*

*git commit -m "deteled file from local"*

*git revert commitid*

*git branch*

*git branch branchname*

*Devops Tools:*

*=============*

*1. Source Code Management:*

*---------------------------*

*GIT, GIT HUB, Bit Bucket, Subversion(SVN), Tortise SVN), CVS, VCS*

*What is GIT?*

*- its open source and is developed by Linus Torvalds.*

*- Distributed version Control System*

*- Tool that tracks changes in your code/files over time.*

*- Working Directory, Stagging area, local repository and remote repository*

*- GIT stores the data in sanpshots that is the reson that its performance is very fast and it can store huge amount of data.*

*What is GIT hub?*

*GitHub is the website where we host all of our git repositories.*

*Connect to Ssh client*

*# ssh -i "pemkey" ussername@public ip adress*

*# ssh -i "dockertoday.pem" ec2-user@publicipaddress*

*What is repository?*

*Why we are using GIT?*

*- A developer has to manage or track changes to the code.*

*- When multiple developers have to work simutenously and contribute to final code*

*- When we have to track who or when changes were made.*

*- When we have to manage all the older versions of the code.*

*The Solution is: SOURCE CODE MANAGEMENT SYSTEM or VERSION CONTROL SYSTEM*

*What is a Version Control System?*

*- A system that documents changes made to a file or set of files.*

*- manages multiple revisions to files*

*- manages files from multiple users.*

*"It is snapshot of your project overtime"*

*Types of Version Control System:*

*-----------------------------------*

*1. Local Version Control*

*- One to One interation between developer and the local Version System.*

*Challenge:*

*This system doesnot support managing of data when multiple developers are working parallely on a project*

*Eg: RCS*

*2. Centralized Version Control System*

*- In this system multiple developers are working together and can move and track there chnages on to a Central repository.*

*eg: CVS,subversion, perforce, TFS(team foundation Version Control) etc*

*3. Distributed Version Control System*

*This system is Distributed in nature where in all the developers will have a working Directory, local repository and a Central repository*

*eg: GIT, Mercurial, BITKeeper, Darcas, Bazzar etc.*

*=================================================================*

*GIT Installation:*

*------------------*

*# sudo su -*

*# yum install git*

*# git --version*

*===========================================================*

*Once file is created in GIT:*

*----------------------------*

*git status*

*git add*

*git commit -m*

*==============================================================*

*Scenario1: Create a Working Directory*

*# mkdir gitDemo*

*# cd gitDemo*

*# mkdir project*

*# cd project*

*- Working directory is ready*

*Initiating a Local repository in working directory*

*# git init*

*# ll -al*

*Local repository intiated in project working directory. Now all files in working directory will be tracked by git.*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Scenario 2: Creating a new file and adding it to local repository*

*# vim file1*

*# git status // file is not tracked by git*

*# git add file1*

*# git status // file is in stagging area and not commited yet*

*# git commit -m "commit file1"*

*# ls*

*# git ls-files // file is now being tracked by GIT*

*Again one more example:*

*# vim file2*

*# git status // file is not tracked by git*

*# git add file2*

*# git status // file is in stagging area and not commited yet*

*# git commit -m "commit file2"*

*# ls*

*# git ls-files // file is now being tracked by GIT*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Scanrio 3: Check commit history or log*

*# git log // will show the commit history in detail*

*# git log --oneline*

*# git show commitid // will show chnages happened in this commit*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Scenario4: COMMIT and ADD an Existing file*

*# vim file2 // open exisiting file and make chnages to it*

*# git status // file is being tracked only chnages have to be commited*

*# git commit -a -m "modified file2"*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Scenario 5: Checking Unstagged chnages and stagged changes*

*# vim file2 // open exisiting file and make chnages to it*

*# git diff file2 // will show chnages that are made to file2 when it is not stagged yet*

*# git add file2 // stage the file*

*# git diff --staged file2 // will show chnages that are made to file2 when file is in stagging area*

*# git commit -m "new chnages"*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Scenario6: Deletion of file*

*- Deletion of file from WD & local repo*

*# git rm file1*

*# git status*

*# git commit -m "deletion of file1"*

*# ls*

*# git ls-files*

*- Deletion of file local repo only*

*# git rm --cached file2*

*# git commit -m "deteled file from local"*

*# git status // file will be untracked*

*# ls*

*# git ls-files*

*# add the untracked file to local repo*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Scenario7: Ignoring Files*

*# vim log*

*# git status // file untracked*

*# vim .gitignore // add name of file "log" , which wil be not ignored*

*# git status // log will be ignored and .gitignore untracked*

*# git add .gitignore*

*# git commit -m "added ignore file"*

*# git status // working tress has to be clean*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*what is Git Revert?*

*---------------------*

*what ever the file you have deleted to take back we can use git revert commitid.*

*Scenario 8: Reverting the changes:*

*# vim file3 // create a new files*

*# git add file3*

*# git commit -m "added file3"*

*//Delete the file*

*# git rm file3*

*# git commit -m "deleted file3"*

*# ls*

*# git ls-files*

*# git log --oneline // last commit id is of delete action, copy it*

*// lets revert the commit*

*# git revert commit id*

*// file back to WD and local*

*# ls*

*# git ls-files*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Scenario 9: Resetting the commits OR Removing Certain changes*

*# git log --oneline*

*# git reset --hard commitid // its removed in staging area and back to working directory.*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Branching Strategy*

*----------------*

*Dev branch*

*qa branch*

*stg branch*

*master-->prod branch*

*How to create branch:*

*--------------------*

*git branch*

*git branch branchname*

*git branch dev1*

*How to delete branch:*

*---------------------*

*git branch -d branchname*

*git branch -d qa*

*how to switch/checkout the Branch*

*---------------------------------*

*git branch*

*\*Master*

*dev1*

*git checkout dev*

*Master*

*\*dev1*

*GIT merge:*

*--------------------*

*Merging the data between two different branches*

*git merge sourcebranchname(dev1) and destinationbranchname(qa)*

*git branch qa*

*git merge dev1 qa*

*vi file5*

*git status*

*git add file5*

*git commit -m "add file5"*

*ls*

*git checkout qa*

*file5 is not there in qa*

*then merge it from dev1 to qa*

*git merge dev1 qa*

*now file5 is in qa branch*

*===============================================*

*git conflict*

*-----------*

*git branch*

*\*dev1*

*qa*

*master*

*git checkout master*

*dev1*

*qa*

*\*master*

*vi file5*

*git status*

*git add 'file5"*

*git commit -m "added file5"*

*git status*

*then merge master file5 with dev branch(it is already having file5)*

*git merge master dev1*

*you can see now git merge conflicts error beacuse of dev1 is already having file5 directory*

*To remove the merging conflict error, open the file5 and clear the extra characters as below*

*<<<<<<<*

*======*

*>>>>>>>*

*then*

*git status*

*git add file5*

*git commit -m "added file5"*

*git merge master dev1*

*===========================================================*

*GIT stash(stashing files in GIT):*

*-------------------------------*

*git stash-->stores the temporary space*

*git stash list*

*vi login.txt(don't do add or commit)*

*git stash*

*git stash list*

*# git stash ==> all the untracked changes will be moved to temperory space*

*Where is this temporary space?*

*# git stash list ==> will give list of chnages stashed*

*# git show stashnumber ==> give stash number*

*# git stash pop stash@{0} ==> will revert back all untracked file chnages*

*After unstashing, the files will be dropped from temperory shelves.*

*OR*

*# git stash apply statsh@{0} ==> will revert back chnages from temperory shelves to working directory*

*// and files still remains on temperory shelves if needed latter in other branches.*

*# git statsh list ==> nothing will be there on executing stash pop.*

*PARTIAL STASH*

*# git stash -p ==> give y for which ever file we want to stash*

*==> give N if we don’t want to stash*

*# git stash drop stash@{0}*

*# git stash list*

*# git stash clear ==> no stash will be there*

*===============================================================*

*GIT rebase:*

*------------*

*git branch*

*\*dev*

*qa*

*master*

*vi file6*

*git commit -a -m "file6"*

*git status*

*git log --oneline*

*copy the commitId here*

*then checkout to qa branch*

*git checkout qa*

*qa don't have the file6, then merge the file6 to qa branch using rebase instead of merge commands*

*git branch*

*dev*

*\*qa*

*master*

*git rebase commitid*

*=======================================================================================*

*difference between git merge & rebase?*

*=====================================================================================*

*git reset --hard commitid -->which will remove any changes form local repository*

*git reset --soft commitid -->it will keep your files, stage all the cahnges back automatically*

*git reset --mixed commitid --> keeps all the files same, but unstages the changes*

*=============================================================================================*

*GIT HUB:*

*--------*

*It is website hosting, we can store all the repositories, branches, files, and code*

*GIT HUB Account Creation:*

*--------------------------*

*Note: From OCT 2020 they have introduced "Main" branch in GIT HUB*

*==========================================================================================*

*GIT Clone:*

*----------*

*Donwloading the data from Remote repository to local repository*

*git clone URL*

*git clone https://github.com/lokeshlingareddy/gitlearning.git*

*=======================================================================================*

*git push:*

*----------*

*Pushing the data from local repository to remote repository*

*git remote add origin https://github.com/lokeshlingareddy/gitlearning.git*

*git remote -v*

*git push origin master*

*username : lokeshlingareddy*

*password : ghp\_Yc6r8aa6uFferVuup5BHUpbZgw5SSA3bxNBz -->personal access token*

*=======================================================================================*

*git pull:*

*------------*

*git pull*

*downloading the data from remote to local repository*

*git fetch:*

*---------*

*git fetch*

*whatever the data you updated on the existing file, that will be download*

*Personal Access token creation:*

*--------------------------------*

*GITHUB merge the data(GIT pull request creation):*

*-----------------------------------------------*

*In GITHUB, we can merge the data using compare&pull request from GITHUB console instead of merge command*

*=========================================================================================================*

*How to pork the data from one account to another account(New person joined):*

*---------------------------------------------------------------------------*

*Pork is nothing porking the specific branch from one account to another github account*

*How to Import the repository:*

*---------------------------------*

*Import means we can import the entire repository from one account to another github account*

*clikc on + and select Import repository*

*How to invite collaborators to work on same repository:*

*------------------------------------------------------*

*go to setting-->Collaborators->Add people*

*How can we change the default branch from Main to another:*

*----------------------------------------------------------*

*go to settings-->Default branch, there you can change it*

*How can we delete the branch from GITHUB console:*

*-----------------------------------------------------*

*================================================================================================*

*How to release the verion or GIT tag:*

*----------------------------------------*

*git log*

*git tag v1.1 commitid*

*git tag*

*from Console, go to settings-->tags-->create rule*

*can you please give the git tag naming conventions:*

*---------------------------------------------------*

*alpha--v1.0*

*beta--v1.1*

*charle--v1.2*

*gama--v1.3*

*================================================================================================*

*What is the solution when :*

*- A developer has to manage or track changes to the code.*

*- When multiple developers have to work simutenously and contribute to final code*

*- When we have to track who or when changes were made.*

*- When we have to manage all the older versions of the code.*

*The Solution is: SOURCE CODE MANAGEMENT SYSTEM or VERSION CONTROL SYSTEM*

*What is a Version Control System:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*- A system that documents changes made to a file or set of fil- manages multiple revisions to files*

*- manages files from multiple users.*

*"It is snapshot of your project overtime"*

*Types of Version Control System:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*1. Local Version Control*

*- One to One interation between developer and the local Version System.*

*Challenge:*

*This system doesnot support managing of data when multiple developers are working parallely on a project*

*Eg: RCS*

*2. Centralized Version Control System*

*- In this system multiple developers are working together and can move and track there chnages on to a Central repository.*

*eg: CVS,subversion, perforce, TFS(team foundation Version Control) etc*

*3. Distributed Version Control System*

*This system is Distributed in nature where in all the developers will have a working Directory, local repository and a Central repository*

*eg: GIT, Mercurial, BITKeeper, Darcas, Bazzar etc.*

*Continuous Integration: (CI)*

*======================*

*Jenkins, cloudbess Jenkins*

*Teamcity,*

*GitLab,*

*Bamboo*

*codeship*

*Travis CI*

*AWS:*

*=====*

*Code pipeline*

*what is jenkins:*

*==================*

*Pre-Req:*

*============*

*1. Git - done*

*2. Java -installation - done*

*environment variable (env variable) - env path*

*/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.372.b07-1.amzn2.0.1.x86\_64*

*3. Maven - installation - done*

*maven env*

*Maven home: /usr/share/maven -*

*4. jenkins - jenkins.io*

*jenkins dashboard:*

*====================*

*jenkins-master- public ip:8080*

*subversion plugin - install*

*Jenkins Logs Path:*

*====================*

*/var/lib/jenkins*

*Slide 65 for Maven*

*Git installation:*

*yum install git -y*

*If using Git class instance-- go to root Directory*

*SEE THAT YOU ARE IN ROOT DIRECTORY*

*# cd*

*# pwd*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*JAVA Installation*

*SEE THAT YOU ARE IN ROOT DIRECTORY*

*sudo amazon-linux-extras install java-openjdk11*

*yum install fontconfig java-11-openjdk*

*export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-11.0.19.0.7-1.amzn2.0.1.x86\_64*

*export PATH=$JAVA\_HOME/bin:$PATH*

*source /etc/profile*

*echo $JAVA\_HOME*

*MAVEN INSTALLATION*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*# yum install maven*

*# mvn --version*

*# vim /etc/profile*

*i*

*export MAVEN\_HOME=/usr/share/maven*

*ESC :wq!*

*# source /etc/profile*

*# echo $MAVEN\_HOME*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Jenkins Installation:*

*=============================*

*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-2023.key*

*yum install jenkins -y*

*systemctl status jenkins*

*systemctl start jenkins*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Clone GIt DEVOPS repository*

*mkdir myproject*

*cd myproject*

*mkdir projects*

*cd projects*

*git clone https://github.com/Sonal0409/DevOpsClassCodes*

*ls*

*cd DevOpsClassCodes*

*ls*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*RUN MAVEN COMMANDS*

*# mvn compile*

*# cd /root/myproject/projects/DevOpsClassCodes/target/classes*

*ls*

*cd com*

*ls*

*cd edurekademo*

*ls*

*cd utilities*

*ls*

*cd*

*pwd*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*INSTALL JENKINS*

*Go to https://pkg.jenkins.io/redhat-stable/*

*and copy these commands*

*# 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*

*# yum install jenkins*

*# systemctl start jenkins*

*# systemctl status jenkins*

*# clear*

*# cat /var/lib/jenkins/secrets/initialAdminPassword*

*Click on install plugins*

*Give username and password details*

*Jenkisn is ready to use*

*Creation of a Job:*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*#1 Jenkins --> New Item --> Job1 --> Build --> shell comand---> echo 'this is my first job'*

*Save --> click on Build Now --> click on build number --> click on Console to see output*

*#2 Jenkins --> New Item --> Job2 --> Build --> shell comand--->*

*date >> /tmp/date.txt*

*echo 'this is my second job'*

*Save --> click on Build Now --> click on build number --> click on Console to see output*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Creation of new User*

*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Jenkins--> Manage Jenkins--> Create/Add Users ----> left side ---> add User*

*Add Name as DEV-->Last name -->Password--> Confirm Password--> Email --> submit*

*New user created.*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*MANAGE AUTHORIZATION*

*Matrix based Security*

*Goto MangeJenkins--->Global security --> Authorization Section---> select Matrix based Security*

*---> Click on Add Users --> write admin---> give adminsitation access*

*----> click on ADD --> write dev--> give Read access and job read access*

*--> logout as admin -- login as Dev --> no jobs, only read access*

*--> logout as Dev ---> login as Admin*

*Project based Matrix Security*

*Goto MangeJenkins--->Global security --> Authorization Section---> Select Project based Matrix Security*

*--> Click on Add Users --> write admin---> give adminsitation access*

*----> click on ADD --> write dev--> give Read access*

*--> go to Job1 --> configure--> general --> Check Enable project based security*

*--> Add user as Dev --> give him access to read, build, write, workspace, build*

*--> go to Job1 --> configure--> general --> Check Enable project based security*

*--> Add user as Dev --> give him access to read*

*Save file*

*Logout as admin--> login as Dev --> you will see both the jobs , job2 will be only read access.*

*Logout as dev and login as Admin*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Plugins*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*AUDIT TRAIL :*

*-->Manage Jenkins-->Manage plugins--> go to Available tab -->give audit trail*

*---> click on checkbox-- Install without restart -- Installation complete*

*--> Manage Jenkins--> Configure System--->Scroll down to AUDIT TRAIL*

*--> From drop-down select --> log file*

*---> log location : /tmp/jenkinsAudit.log*

*---> Log file size --> 25*

*---> Log file count --> 5*

*---> Log Seperator ==> leave it blank or give any special character.*

*Go to job 1 and rerun it*

*Now go to server*

*# cd /tmp*

*# ls*

*Audit log file will be available with name as ==> jenkinsAudit.log.0*

*# cat jenkinsAudit.log.0 ==> to see contents of log.*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*EMAIL NOTIFICATION*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*--> Manage Jenkins--> Extended Email Notifications*

*Give:*

*Smtp server ==> smtp.gmail.com*

*port : 465*

*Click on advance*

*SMTPUsername:*

*Password: --> --> click on chnage password to enter yur password.*

*Check SSL checkbox*

*Save the chnages*

*Now go back to Jenkins--> Job1 --> configure--> post build actions-->edit email notification-->*

*---> recipient list as ==> batch123@....*

*--> scroll down select Attach build log*

*--> Click on ADVANCE*

*--> scroll down to TRIGGER*

*--> Remove Failures*

*--> Click on Add trigger button below and select Always*

*---> select recipents list and remove developers.*

*Save and build now on Job.*

*Email will be sent with log*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*TRIGGERS*

*If we want to automatically trigger a Job*

*1. Trigger prieodically*

*Create anew Job--> build trigger--> build preodically --> give \*/2 \* \* \* \* --> Save and build now*

*All build will be genrated automatically every 2 mins*

*2. Trigger by POLL SCM*

*Create a new Job-->Source code management--> Add git path*

*https://github.com/Sonal0409/GITHUBJENKINSDEMO*

*Build Triggers--> Poll SCM --> \* \* \* \* \**

*Build will be generated each time there is a chnage in the repository only.*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Delivery Pipeline in Jenkins*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Jenkins--> Manage Jenkins-->Global Tool configuration*

*Go to JDK section*

*In this give:*

*1. Uncheck -- install automatically*

*2. JDK name as myjava*

*3. Go to server type*

*# echo $JAVA\_HOME ==> will give java location*

*JAVA\_HOME =*

*/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.265.b01-1.amzn2.0.1.x86\_64*

*Save it.*

*Under GIT==> leave it same*

*Under maven ==> lets install it automatically*

*Type mymaven and check the install automatically box.*

*Save the changes*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*CREATION OF COMPILE JOB*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*1. new Job --complie-->freestyle project*

*2. Source code management --> select git*

*--->give git path ===> https://github.com/Sonal0409/DevOpsClassCodes.git*

*3. build ==> select invoke top level maven targets*

*select maven version===> mymaven*

*goal ==> compile*

*4. save ==> build now*

*So all the compiled files will be present at this location*

*==> look for this line in console output at the end on jenkins*

*Compiling 13 source files to /var/lib/jenkins/workspace/compile1/target/classes*

*5. Go to workspace on the left side of jenkins Job compile*

*In the folder go to ==> target folder==> go to classes/com/edurekademo*

*==> click on edurekademo==> got to utilities==> all class files will be there which have recently been compiled.*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*2nd JOB ==>Code Review*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Jenkins--> new item--> Name= code Review==> freestyle project*

*==> source codemanagement==>select git==>*

*give git repo name https://github.com/Sonal0409/DevOpsClassCodes*

*Step 2: build ==> invoke top level maven target==>*

*select maven version==>mymaven ==> goal = pmd:pmd*

*save==> build now*

*==> click on build number and see the console*

*Goto Workspace on the left side of jenkins Job code review*

*In the folder go to ==> target folder ==> you will find pmd.xml file.*

*CONVERT FILE TO TREND REPORT*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*manage jenkins--> manage plugins--> available --> Search for pmd plugin--> install it.*

*Now go to the job Code Review==> go to post build actions ==> select Publish PMD anaylsys report*

*under PMD results==> give path of pmd.xml file ie: target/pmd.xml*

*Save the file and build now.*

*After the build is successfull, you will see PMD Warnings*

*So you will see 12 new warning have been generated on the code.*

*Scroll down*

*Under package click on first one to check warnings.*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*JOB3: Testing --Unit Test report*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*1. jenkins--> new item--> Name= unitTest==> freestyle project*

*==> source codemanagement==>select git==> give git repo name https://github.com/Sonal0409/DevOpsClassCodes*

*2. build ==> invoke top level maven target==> select maven version==>mymaven ==> goal = test*

*3. Save and Build now.*

*4. Check the workspace*

*5. surefire-reports will be present*

*But we cannot understand them easily. So lets generate understandable reports by using Junit reports option under post build actions*

*6. Go back to job==> post build actions==> select junit test result job*

*==> give target/surefire-reports/\*.xml ===>under test report xml*

*This is path where xml files are there.*

*Here \*.xml is as we want to use all the xml files*

*7. Save and build again*

*Go to build number ===> on left side you will see Test Result*

*Click on it and you can see all pass and fail details of the tests*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*JOB4 : Metric check*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*1. Go to jenkins--> new item--> Name= Metriccheck==> freestyle project*

*==> source codemanagement==>select git==> give git repo name https://github.com/Sonal0409/DevOpsClassCodes*

*2. build ==> invoke top level maven target==> select mymaven*

*goal = cobertura:cobertura -Dcobertura.report.format=xml*

*3.==> save==> build now*

*==> click on build number and see the console*

*4. go to Workspace ==> target/site/cobertura/coverage.xml*

*The xml file will be there, But we cannot understand them easily. So lets generate understandable reports by downloading cobertura plugin*

*Go to manage jenkins--> manage plugins--> cobertura --> install it*

*Go back to the job--> post build actions--> select publish cobertura coverage report*

*Under cobertura xml pattern give ==> target/site/cobertura/coverage.xml ==> you can take the path from below displayed text.*

*Save and build again*

*Click build number ==> on left side you will find Coverage Report==> click on it ==> we cna see the coverage report.*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Job5 Package Job*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Go to jenkins--> new item--> Name= Package==> freestyle project*

*==> source codemanagement==>select git==> give git repo name https://github.com/Sonal0409/DevOpsClassCodes*

*build ==> invoke top level maven target==>mymaven*

*goal = package*

*==> save==> build now*

*==> click on build number and see the console*

*==> go to workspace ==> target/addressbook.war*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Connecting all the 5 jobs so that each job can be triggered:*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Pipeline as a code*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Lets create a pipeline using code:*

*Create 1 job and add multiple task (using code) in the job and run multiple tasks in 1 job*

*This code is written using pipeline DSL(domain specific language]*

*It is derieved using groovy*

*Create a new job ==> select project as pipeline*

*pipeline{*

*tools{*

*jdk 'myjava'*

*maven 'mymaven'*

*}*

*agent any*

*stages{*

*stage('checkout'){*

*steps{*

*git 'https://github.com/Sonal0409/DevOpsClassCodes.git'*

*}*

*}*

*stage('compile'){*

*steps{*

*sh 'mvn compile'*

*}*

*}*

*stage('codeReview'){*

*steps{*

*sh 'mvn pmd:pmd'*

*}*

*}*

*stage('unitTest'){*

*steps{*

*sh 'mvn test'*

*}*

*}*

*stage('metricCheck'){*

*steps{*

*sh 'mvn cobertura:cobertura -Dcobertura.report.format=xml'*

*}*

*}*

*stage('package'){*

*steps{*

*sh 'mvn package'*

*}*

*}*

*}*

*}*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Master and Slave Set up*

*restart the instance.*

*Restart the jenkins server*

*now take public ip and loginto jenkins*

*Step 1: Go to Manage Jenkins ---> CONFIGURE SYSTEM ---> Check the Jenkins URL is same as URL of browser --> Save*

*Step 2: Go to Manage Jenkins---> CONFIGURE GLOBAL SECURITY --> Scroll down to AGENTS-->select RANDOM radiobutton--> Save*

*Step 3: Go to Manage Jenkins---> MAnage Nodes and Clouds --> New Node*

*Select New Node--> give name as Winslave--> select radio button permanamet Agent*

*Name : winslave*

*Description: give windows 10 machine*

*==> # of executors = 1 ==> means number of jobs that we want to run on this slave node. Let it be 1 for now*

*===> Remote root directory ==> C:\jenkinsdir*

*Location of a directory folder on your windows machine , so that jenkins can create a workspace folder and store all the files. As we have seen earlier while executiong jobs.*

*For this --> got to C drive and create an emapty folder.*

*C:\jenkinsdir*

*====> Label : win\_slave ==> this is like a tag to slave machine. This is how we will identify the slave machine--> remeber this*

*===> Usage:*

*Select 2nd option: Only build jobs with label expressionmatching the job :*

*This means for which ever job we will give label as win\_slave that job only will run on this job.*

*===> Launch Method : select option : Launch agent by connecting to master.*

*Scroll down*

*Add tools*

*Add git path as C:\Program Files\Git\cmd\git.exe*

*Add java path as C:\Program Files (x86)\Java\jdk1.8.0\_191*

*SAVE the job.*

*Now go to win\_slave ==> download the agent.jar file and put it in your jenkinsdir folder.*

*Copy the path and change the public ip in below path*

*java -jar agent.jar -jnlpUrl http://3.139.60.71:8080/computer/winslave/slave-agent.jnlp -secret 61e8e20badd531b48caeffac8d686f182eb2d77a1c303ee39c5db9d00117c0be -workDir "C:\jenkinsdir"*

*Go to bash and start jenkins, refresh instance*

*Go to cmd prompt*

*change to cd c:\jenkinsdir*

*copy the above jenkins agent path*

*System will get connected.*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*RUN a job on Agent*

*Go to Dashboard*

*Select the package job---> configure --> GENERAL --> click on Restrict where this project can be run and enter win\_slave*

*SAVE THE JOB*

*CLICK on BUILD NOW -- to start the build*

*CLick on build number--> got to console --to check it is executing on AGent*

*Go to C folder-->jekinsdir---> C:\jenkinsdir\workspace\package\target==>*

*you will see addressbook.war file.*

*On linux:*

*$ which git*

*CI - Continuous Integration*

*1.Jenkins - 80% - CloudBees Jenkins*

*2.Circle CI*

*3.Teamcity*

*4.Bambooy*

*5. GitLab*

*6.Travis CI*

*7. CodeSHip*

*Jenkins*

*+++++++++++++*

*Jenkins is a CI server that manages and control process such as*

*Plan, Code, Build, test, deploy operate and monitor in DevOps Env.*

*Jenkins:*

*+++++*

*3 Pre-Requists*

*1. Java - 1.8 version*

*2. Git*

*3. Maven - Build Tool ( Ant, Gradle)*

*#(root directory)*

*8080 - default*

*Note: Jenkins is purely working with plugins*

*what is jenkins job?*

*====================*

*A Jenkins job is a sequential set of tasks that a user defines.*

*For example, a job can fetch source code from version control, compile the code, run unit tests, and more. Note that in Jenkins, the term “job” is synonymous with “project”*

*2 Types:*

*1. Freestyle job - Manual - UI*

*2. Pipeline - Groovy Scripting - CI/CD*

*what is Freestyle Job?*

*================*

*This is the central feature of Jenkins.*

*Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.*

*what is a pipeline?*

*================*

*Orchestrates long-running activities that can span multiple build agents.*

*Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.*

*Jobs Creation:*

*Build Tool of Maven*

*=================*

*1. compile :*

*===========*

*convert into machine understandable language*

*2. Code Reviiew*

*================*

*reporting task, code analysis report - PMD Tool*

*3. Unit Test*

*============*

*execution of test cases, test report( pass or fail)*

*4. Code Coverage*

*===========*

*% code covered by the test executes, coverage report*

*5. Package*

*==============*

*its convert into .jar, .war, .ear*

*Build Life Cycle*

*=================*

*validate - validate the project is correct and all necessary information is available*

*compile - compile the source code of the project*

*test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed*

*package - take the compiled code and package it in its distributable format, such as a JAR.*

*verify - run any checks on results of integration tests to ensure quality criteria are met*

*install - install the package into the local repository, for use as a dependency in other projects locally*

*deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.*

*why Jenkins is so popular*

*==========================*

*open source*

*Good Plugin support*

*Good community support*

*Fast and reliable*

*Good OS Support*

*Scripted Build*

*Features of Jenkins*

*===============*

*Easy installation process*

*provides advance security*

*upgrades are easily available*

*light weight container support*

*Distributed Team Management*

*Master & Slave configuration:*

*++++++++++++++++++*

*Jenkins Master:*

*1.Jenkins Master perform basic installtion and handles task related to build and configuration.*

*2.schedule builds*

*3.they moniter slaves*

*4.records and presents the build resulits*

*Jenkins Slave:*

*\_\_\_\_\_\_\_\_\_\_\_-*

*1. Slaves are basically setup to offload builds from the master and disturbute the work loads*

*2. They listen to the master request*

*3. Slaves can run on variety of OS.(Linux & windows)*

*4. They minely execute bild jobs which are dispached by jenkins master instance jobs*

*Jenkins:*

*CI*

*scm:*

*git, svn, cvs,TFS*

*SVN - Plugin needs to be in the jenkins*

*Master and Slave Configuration:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Jenkins Pipeline :*

*2 type of pipeline jobs*

*1. Scripted pipeline - node (old )*

*2. Declarative Pipeline -*

*Groovy script we are using declarative pipilene*

*2 ways*

*- pipeline script - Groovy script*

*- Snippest generator - jenkins jobs dash board*

*5 Stages:*

*++++++++++*

*1. Continous Downloading(github)*

*2. Continuous Build*

*3. Continuous Deployment- automatic/manual deployment*

*4. Continuous Test - test*

*5. Continuous Delivery - ( final production delivery)*

*Groovy Scripting:*

*+++++++++++++*

*1. Pipeline{*

*agent none*

*stages{*

*stage('Checkout from git'){*

*agent any*

*stpes{*

*sh 'https://github.com/SobhaReddy/myE2e05Aug.git'*

*}*

*}*

*Docker*

*========*

*Containeraizaion tool that we are going to learn docker*

*Rancher*

*Developing the code - dockerfile - implement docker*

*Building the code - maven - docker build the images*

*shipping and running container- jenkins - docker*

*docker is purely working with port number*

*docker pre-req:*

*====================*

*docker server is available*

*ec2 -instance*

*install docker*

*On promises application:*

*========================*

*100GB*

*500 GB in VM wares*

*===================*

*200 GB - We are ging to production deployment*

*300 GB - Pay the amount*

*cloud:*

*=============*

*200 GB -*

*what is the diffrence between Virtual Machine vs Containeraization/Docker:*

*======================================================================*

*VM - Allocat the memory*

*docker - no memory allocation*

*VM - Guest - linux*

*docker -linux - server*

*crate a Dockerfile:*

*======================*

*Docker instructions*

*FROM*

*MAINTINAER*

*RUN*

*CMD*

*EXPOSE*

*install git in Docker:*

*========================*

*take single line command - hub.docker.com*

*docker pull git*

*dockerfile for git installation*

*==============*

*FROM centos:7*

*MAINTAINER shobha*

*RUN yum install -y git*

*CMD ['/bin/bash']*

*EXPOSE 80*

*Execution:*

*==============*

*build the images*

*1. docker build -t git .*

*create the container*

*2. docker run --name myc1 -p 8080:80 git /bin/bash*

*Docker:*

*\_\_\_\_\_\_\_\_\_\_\_*

*Containeraization tool- Docker , Rancher*

*Docker is a open source plaform for develeping, building, shipping and running application*

*Docker installation:*

*+++++++++++++*

*https://docs.aws.amazon.com/AmazonECS/latest/developerguide/create-container-image.html*

*VM - On Primises -200gb,*

*Docker - Cloud - mb size*

*Diff b/w Docker conatiner & VM(Virtual Machine)*

*=================================*

*https://dockerlabs.collabnix.com/beginners/difference-docker-vm.html*

*Docker image - is a set of binaries and libraries, its just a stastic file*

*Container- Container is nothing but running instances of an images*

*vim Dockerfile*

*+++++++++++++++++*

*FROM --> Which base image that you want to use your custom image either it could be OS*

*MAINTAINER - author of the file -my name/emai id*

*RUN --> When you want to install/update/upgrade - using with Linux command - RUN yum install git -y & Maven, Jenkins*

*EXPOSE --> Container port 80, 8080, 9090*

*COPY - Copy file from host to container*

*ADD - Copy file from host to container can download files from remote server and place on container*

*LABEL --> specify with the tag name/version*

*USER --> Default user who should log on to the container / root*

*CMD --> Run an application on the container -- /bin/bash*

*ENTRYPOINT--> to run default process to the container*

*VOLUME--> is a keyword that use to define default volumes of the container - container is crashed*

*ENV - ENV is for future running containers. ARG for building your Docker image. ...*

*ENV is mainly meant to provide default values for your future environment variables.*

*Running dockerized applications can access environment variables.*

*It's a great way to pass configuration values to your project*

*Dockerfile:*

*++++++++++++++*

*FROM centos:7*

*MAINTAINER sobha*

*RUN yum update -y*

*RUN yum install -y git*

*CMD ['/bin/bash']*

*\_\_\_\_\_\_\_\_\_\_\_\_*

*Execution:*

*1. Build the Docker Image:*

*docker build -t centos .*

*2. Creating/running the Docker:*

*docker run -itd --name c1 -p 8080:80 centos*

*3.Execution/inside the docker*

*docker exec -it c1(containername/conatinerid) bash*

*git --version*

*Checking with Images:*

*++++*

*docker images*

*or*

*docker image ls*

*Remove Image:*

*++++++*

*docker rmi imagename/imageid*

*Checking with Containers:*

*+++++++*

*docker ps( will show only working containers)*

*docker ps -a( will show working & exicited containers)*

*Delete All Container :*

*++++++++++++++++++++*

*docker rm -f $(docker ps -aq)*

*Delete All Imges:*

*+++++++++==*

*docker rmi prune -a*

*Login inside the container*

*=====================*

*docker exec -it containername /bin/bash*

*Remove container:*

*+++++*

*docker rm containerid/container name*

*what is nginx:*

*++++++++++++++*

*nginx is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache.*

*What is reverse proxy:*

*====================*

*A reverse proxy is a server that sits in front of web servers and forwards client (e.g. web browser) requests to those web servers.*

*Reverse proxies are typically implemented to help increase security, performance, and reliability.*

*what is daemon off in nginx*

*+++++++++++++++++++++*

*For Docker containers (or for debugging), the daemon off; directive tells Nginx to stay in the foreground. For containers this is useful as best practice is for one container = one process.*

*One server (container) has only one service*

*what is Foreground and background in docker*

*======================================*

*Background mode = Run in the background continuously without stopping (like daemon).*

*Foreground or console = If you exit console the process may stop*

*End to End Deployment for nginx webserver*

*======================================*

*Nginx Webserver using with Ubuntu Image:*

*+++++++++++++++++*

*vim Dockerfile*

*FROM ubuntu*

*MAINTAINER sobha*

*RUN apt-get update -y*

*RUN apt-get install -y nginx*

*COPY index.html /var/www/html*

*EXPOSE 80*

*CMD ["nginx", "-g", "daemon off;"]*

*Exc:*

*docker build -t myn1 .*

*docker run --name mynginx2 -p 80:80 -d myn1*

*Connect the webserver*

*===================*

*Dockerpublic:80*

*login with hub.docker.com*

*docker login in local server*

*username:*

*password*

*docker tag mynginx z12bsobh/mynginx(your private dockerid)*

*docker push z12bsobh/mynginx:latest*

*what is nginx reverse proxy:*

*=======================*

*Configure NGINX as a reverse proxy for HTTP and other protocols, with support for modifying request headers and fine-tuned buffering of responses.*

*This article describes the basic configuration of a proxy server. You will learn how to pass a request from NGINX to proxied servers over different protocols, modify client request headers that are sent to the proxied server, and configure buffering of responses coming from the proxied servers.*

*Diff B/w P and p:*

*++++++++++++++*

*P(Upper Letter) -It will take automatic port number*

*p(Lower Letter)- it will ask Externalip(ContainerIP):internalip(HostIP)*

*List all containers that exited?*

*==============================*

*docker ps --filter "status=exited"*

*what is docker networking:*

*++++++++++++++++++*

*Docker networking is primarily used to establish communication between Docker containers and the outside world via the host machine where the Docker daemon is running.*

*Docker supports different types of networks, each fit for certain use cases.*

*types of docker networking:*

*+++++++++++++++++*

*1.bridge networks- used within a single host*

*2.overlay networks- for multi-host communication*

*3.macvlan networks - which are used to connect Docker containers directly to host network interfaces.*

*4.Host networks - are best when the network stack should not be isolated from the Docker host, but you want other aspects of the container to be isolated.*

*5.Third-party network plugins allow you to integrate Docker with specialized network stacks.*

*Docker network creation commands:*

*==================*

*docker network create myn1*

*attach network container*

*==================*

*docker run -itd --network=myn1 nginx(image name)*

*docker run -itd --name myc4 -P nginx*

*Docker inspect containerid/name*

*what is Tomcat Application:*

*++++++++++++++++++++*

*Apache Tomcat (called "Tomcat" for short) is a free and open-source implementation of the Jakarta Servlet,*

*Jakarta Expression Language, and WebSocket technologies.[3] Tomcat provides a "pure Java" HTTP web server environment in which Java code can run.*

*Tomcat Application - Dockerfile*

*+++++++++++++++*

*rm -rf docker*

*# Create a Directory*

*mkdir docker*

*cd docker*

*# copy the .war file into the current directory*

*cp /var/lib/jenkins/workspace/package/target/workspace/addressbook.war .*

*# Create Dockerfile*

*touch dockerfile*

*cat <<EOT>> dockerfile*

*FROM tomcat*

*ADD addressbook.war /user/local/tomcat/webapps*

*EXPOSE 8080*

*CMD ["catalina.sh", "run"]*

*EOT*

*sudo docker build -t myproject:$BUILD\_NUMBER .*

*sudo docker run -dt -P myproject:$BUILD\_NUMBER*

*Permission Issue*

*================*

*vim /etc/sudoers*

*jenkins ALL=(ALL) NOPASSWD: ALL*

*what is Docker CGroup?*

*=====================*

*Control Groups (cgroups) are a feature of the Linux kernel that allow you to limit the access processes and containers have to system resources such as CPU, RAM, IOPS and network.*

*In this lab you will use cgroups to limit the resources available to Docker containers.*

*Docker micro-services*

*===================*

*Docker allows you containerize your microservices and simplify the delivery and management of those microservices.*

*Containerization provides individual microservices with their own isolated workload environments, making them independently deployable and scalable.*

*Docker Compose:*

*++++++++++++*

*yaml file - Yaml Ain't Markup Language*

*- Creating with multiple containers we are using docker-compose*

*- It uses YAML format compose file to configure the application services*

*- Using a command , the user can then created and start the service from the configurations.*

*Docker-Compose Working:*

*++++++++++++*

*1. Creaing a Dockerfile for the application*

*2. Define the service tha make application in docker-compose.yml*

*3. Run Docker-compose up to start the application*

*Multi-ContainerDeployment with Compose*

*+++++++++++++++++++++++*

*1. Docker compose uses project names to isolate environment*

*2. The data for all the containers in a service is preserved in volumes*

*3. if a service restarted but nothing has changed*

*4. Variables can be used in Compose file to customise the service for diff env.*

*Compose Common Commands:*

*+++++++++*

*docker-compose build -> build or rebuild a service from the given Dockerfile*

*docker-compose run --> Allow user to run a one-off command in the service*

*docker-compose up --> Create and runs service containers*

*docker-compose down --> Removes the containers, networks, images, and volumes related to the service.*

*Environment Varaible in Compose:*

*++++++++++++++++++++*

*1. Compose can utilize environment variables inside the compose file from the shell.*

*2. Environment Varaible can also be set in the containers by using docker run command with -e flag*

*3. this can be accomplished using docker-compose run command*

*4. Deafult values fo environment variable can also set by creating a new .envfile*

*key: value*

*name:*

*++++++++++++++++*

*3-Tier Architecture:*

*+++++++++++*

*Frontend :*

*Node.js (Developer code)*

*Backend:*

*Python Automation*

*DataBase:*

*\_\_\_\_\_\_\_\_*

*MongoDB*

*Docker-compose:*

*+++++++++*

*yaml file*

*Docker-compose imstallation: on Ubuntu Machine*

*+++++++++++++++++++*

*https://docs.docker.com/compose/install/*

*dockr installation*

*snap install docker*

*docker-compose installation*

*=======*

*1. sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose*

*2. https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose*

*3. sudo chmod +x /usr/local/bin/docker-compose*

*4.docker-compose version*

*Wordpress Application:*

*++++++++++++++*

*WordPress (WP, WordPress.org) is a free and open-source content management system (CMS) written in PHP[4] and paired with a MySQL or MariaDB database. Features include a plugin architecture and a template system, referred to within WordPress as Themes.*

*Pre-requisits:*

*+++++++++++++*

*php*

*mysql*

*wordpress*

*Docker-Compose Installation:*

*==================*

*https://linuxhostsupport.com/blog/how-to-install-and-configure-docker-compose-on-ubuntu-20-04/*

*vim docker-compose.yml*

*======================*

*version: '3'*

*services:*

*db:*

*image: mysql:5.7*

*volumes:*

*- db\_data:/var/lib/mysql*

*restart: always*

*environment:*

*MYSQL\_ROOT\_PASSWORD: password*

*MYSQL\_DATABASE: wordpress*

*MYSQL\_USER: wordpress*

*MYSQL\_PASSWORD: wordpress*

*networks:*

*- wpsite*

*phpmyadmin:*

*depends\_on:*

*- db*

*image: phpmyadmin/phpmyadmin*

*restart: always*

*ports:*

*- '8181:80'*

*environment:*

*PMA\_HOST: db*

*MYSQL\_ROOT\_PASSWORD: password*

*networks:*

*- wpsite*

*Wordpress:*

*depends\_on:*

*- db*

*image: wordpress*

*ports:*

*- '8000:80'*

*restart: always*

*volumes: ['./:/var/www/html']*

*environment:*

*WORDPRESS\_DB\_HOST: db:3306*

*WORDPRESS\_DB\_USER: wordpress*

*WORDPRESS\_DB\_PASSWORD: wordpress*

*WORDPRESS\_DB\_NAME: wordpress*

*networks:*

*- wpsite*

*networks:*

*wpsite:*

*volumes:*

*db\_data:*

*Execution*

*============*

*docker-compose build*

*docker-compose run*

*docker-compose up -d*

*Docker Volumes*

*============*

*Volumes are the preferred mechanism for persisting data generated by and used by Docker containers.*

*While bind mounts are dependent on the directory structure and OS of the host machine, volumes are completely managed by Docker.*

*Volumes have several advantages over bind mounts:*

*Volumes are easier to back up or migrate than bind mounts.*

*You can manage volumes using Docker CLI commands or the Docker API.*

*Volumes work on both Linux and Windows containers.*

*Volumes can be more safely shared among multiple containers.*

*Volume drivers let you store volumes on remote hosts or cloud providers, to encrypt the contents of volumes, or to add other functionality.*

*New volumes can have their content pre-populated by a container.*

*Volumes on Docker Desktop have much higher performance than bind mounts from Mac and Windows hosts.*

*Docker Volumes*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Docker Volume Containers*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*# docker volume create myvol // create a volume on host*

*# docker volume ls // list the volume on the host, myvol will be there*

*# docker volume inspect myvol // gives the path where volume is created on the host*

*/var/lib/docker/volumes/myvol/\_data // copy this path on notepad*

*# cd /var/lib/docker/volumes/myvol/\_data*

*# ls // no files*

*# cd*

*MOUNT THE VOLUME ON A CONTAINER*

*# docker run --name u1 -it -v myvol:/tmp ubuntu*

*# cd /tmp*

*# ls*

*# touch file1 file2*

*#ls*

*# Ctlp+Ctlq*

*Use the same path copied above*

*# cd /var/lib/docker/volumes/myvol/\_data*

*# ls // file1 and file2 files will be present*

*Create 1 more new file here --> file should be available in container also*

*# touch file3*

*# ls*

*#cd*

*# docker ps -a*

*# docker attach u2*

*# ls*

*# cd /tmp*

*# ls*

*# file1, file2 , file3 will be there*

*REMOVE THE CONTAINER AND CHECK IF FILES ARE AVIALBLE*

*# docker rm -f u2*

*# cd /var/lib/docker/volumes/myvol/\_data*

*# ls ==> still files will be available.*

*Delete the volume :*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Delete containers before.*

*# docker volume rm myvol // volume will be removed.*