DevOps – end to end product delivery

4 env :

1. Dev env
2. Test
3. UAT
4. Prod

CI – Continuous Integration, on integration side

CD – Continuous Deployment, on deploying side

Linux – kernel (not os)

Types of root users – 1. Superuser 2. Administrator 3. Root home directory

In linux temp folder is world readable

cat – to display things

ip a s – to check ip address

ll – to list items in detail

ls – to list items

/etc/ : etc directory contains all conguration data

Ex – cat /etc/group - will show all the groups

User made groups will have unique number from 1000.

pwd - print working directory

cd – change directory

/bin/ - contains commands that any user can use (including root user)

/sbin/ - contains commands that only root user can use

When we setup linux from cloud, by default root is locked, we need to set password of root

passwd username / **passwd root**

if the file name starts from **.** (dot) - means it's a hidden file

touch filename.txt - to create a file

cat > filename.txt - to edit in that file

rm filename.txt - to delete, if did not work then - **rm -rf filename.txt**

tree – to list all files in tree structure

su - username / **su - root** - switching or logging in from user

exit - to logout from user

alias – just like define of c++

clear - to clear the terminal

Types of user in linux :

Super user – root

System user – apache, samba, httpd, etc

Regular user

useradd username - create new user

passwd username – set password of that username (necessary as if won't set password can't login to the user)

usermod -s /sbin/nologin username – won't allow the username to login.

groupadd groupname – creates a group

usermod -G groupname username – add the username to groupname

usermod -aG groupname2 username – add same user to a diff group too (if we'll only use -G again then it will just jump out of that group1 and will go to group2)

cat /etc/group | grep -i groupname - in etc group all groups and users are stored and grep is for find, -i for ignore case . this command will find and list the groupname inside etc group.

In linux it doesn't understand name it creates an unique id (uid) for everything.

In devops we have to do operations cost efficient way, removing unnecessary files

Commands are case and space sensitive in linux.

Linux Permissions:

UGO permisiions – User Group Other Permission

Id in start there I '-' it means a regular file.

To check the ugo info : **ll -d /directory/**

Output - drwxr-xr-x 5 root root 4096 Aug 5 09:30 /directory/

U : r – read, w – write, x – execute, owner can do these things

G : r-x : group can read and execute

O : others who belong to neither user or group, they can also read and execute

Group is a collection of users

chown <username> <file name> - to change owner

chgrp <grp name> <dir name> - to change group owner to a directory (grp members will act as root of that directory)

chmod 777 /dir :

read value – 4, write value- 2, execute – 1, so here 777 means ugo will have all the permissions, 777 respectively ugo.

chmod u+rwx g+rwx o+rwx /data : same as chmod 777 /data :

If want to revoke can write : o-rwx or g-wx etc...

execute means directory ke andar ghusna, read uske files ko dekhna and write koi files add karna, yadi execute permission hata diye to read write kar hi nhi sakta.

Max permission on folder/directory – 777
on files – 666

In a group anyone can read write remove/delete anyone's data, that's a problem.

To prevent this – stickybit
chmod o+t /directory/ : only root user can do this, by this no one can delete anyone else's data.

But root user can do anything wither he is owner of the group or not.

If someone from group wants to edit other's data – sgid (set group id): new files or directories inside group will automatically inherit the properties of group

chmod g+s /directory/

When we start the terminal it opens as regular user, we will have to convert to root, by default it is locked we have to change root's password.

sudo su - : logins ec2 to root user.

passwd root : set password for root

bash : sometime hostname wont reflect even after setting it, so bash command.

hostnamectl set-hostname server.asmit.com : set hostname

if you get access denied add **sudo** at start of hostnamectl command –

sudo hostnamectl set-hostname server.asmit.com

Other commands :

hostname – shows hostname

hostnamectl - show all host details

Ec2 user – means sudo user, like a class – teacher is root, monitor is sudo user, sudo has some powers assigned by root

It is used by writing sudo at start of every command.

vim /etc/sudoers : all configuration files are here (if u want to edit)

in vim sudoers it doesn't show the errors (if done by mistake in any line), so use **visudo** – it shows the error line

esc + : (colon) then write- **se nu** : it will show line numbers

if you want to make any user to sudo user:

in line 100 info of root is written, after that line write –

username ALL=(ALL) /usr/sbin/useradd - by this we gave username the permission to add new users

after usr sbin useradd command, by adding coma can give other permissions.

when the username wants to do any op of root he will need to write **sudo** at start of every command

if instead of usr sbin we wrote all, system will crash.

In company don't even give any sudo permission to anyone, it can lead to termination.

Permissions are given to sudo user but I want to restrict him from doing on root:

/usr/sbin/useradd, **!/usr/sbin/useradd root**

To get out of vim editor - **shift** + **esc** + **:** then **wq!** Press enter.

fdisk -l : to show disk details

Daemon :

yum install httpd -y : download apache package, **-y** means yes for all permissions.

systemctl status httpd : shows status of apache

systemctl start httpd : starts apache service

systemctl enable httpd : when restart, service will stop, to keep it active always

default document add of httpd - **/etc/httpd/conf/httpd.conf**

If want to edit the configuration of httpd : **vim /etc/httpd/conf/httpd.conf**

The default port for apache is **80** (can change in httpd.conf (if needed))

systemctl restart apache : reastarts service

Systemctl reload-or-restart httpd.service : when we restart directly server can go down for some time, so by this command computer will decide what is needed reload or restart.

Always use reload-or-restart

systemctl stop httpd : stops the service in current session.

It will get enable in next session if you have run the enable command.

systemctl disable httpd : disables the service, wont get enable in next session.

systemctl mask httpd : masks the service

now you cant start the service (like someone started the service by mistake or intentionally who has the root access), you have to unmask it first and while unmaksing it will get alerted.

systemctl unmask httpd : unmasks the service

Set password for ec2 at start because sometimes it asks for password, so for being in safe side.

VERSION CONTROL

Someone is working on frontend at other location someone on backend at other location on same project, other person joins for same project, will we share files manually everytime? No

To solve it we need a method to keep updating the versions.

SVC – Source code Version Control system - to maintain multiple versions on your system

Like linux is an open source, anyone can contribute but what if a hacker did something bad, that's why linux officials checks it they make it available for everyone.

Linux made SVC for themselves – **GIT**

2 types of version control system :

Centrailised vcs –

Distributed vcs - git

GIT is a distributed version control system.

Github – gui for git

Being a devops you must know to create a git environment.

To create environment:

- Need 3 vms
- assign hostname, must be unique
- linux should be install in every machine
- **vim /etc/ssh/sshd_config**
    - **shift esc : se nu** - shows the line number
    - in line 40, remove **#** (comment), clear **prohibited**, write **yes**
    - line 65, replace no to **yes**
    - **shift esc : wq!** - wq means write(save) and quit, just **q!** means quit without save
- in one vm - **ip a** , copy ip add
- in same vm - **vim /etc/hosts**
- paste the ip and after ip write that ip's hostname (can write any name)
- go to other vms, copy ip and paste in 1st vm
- Do this vim etc hosts and copy all ips of all vm in each other
- in server - **systemctl start sshd** or **systemctl restart sshd** – starts the ssh
- in server - **systemctl enable sshd** - enables ssh (it will auto start on boot)
- in client vms navigate to **cd .ssh/** (optional)
- in client vms – **ssh-keygen** – will generate private and public key
- in server - **ip a** or **ip a s** – opens ip address (both do the same thing)
- on client machine – **ssh-copy-id root@<ip address>**

GIT

- install git package - **yum install git -y**
- make directories on all 3 machines with diff names
- **cd directory**
- **git init --bare** : to make machine a server, if you will have many machines by running this command its configuration will get advanced, it will act as server. (have to do inside the directory)
- **git init** - in other 2 machines (have to do inside the directory)

*when you will do **git init –bare** or **git init**, at the last in the output of the **git init** one there will be a **/.git/** and in bare one it won't be there. We can check the successful process of bare by this*

git remote add origin root@<ip address>:~/<dir name of server> - (origin is the alias)

git remote -v - shows status that if u can fetch and push or not

git status - shows the status

create/modify a file

git add . - add all files in repo

git commit -m <any message> - commit

git push origin master

to share the files to a colleague – in his system –

git clone root@<ip add of server>:~/< dir name of server >

git log – see all commits

in server all commits will show, and client can see his commits only

I want to see changes done by other system - **git pull origin master**

Git server drawback – u need to manage server, infrastructure yourself, also u have to host yourself.

So we will use ***github***: create **ssh-keygen** in your system

To see the key - **cat ~/.ssh/id_rsa.pub**

Now in github – open profile – settings – ssh and gpg keys – new ssh key – write any title – paste the key.

JENKINS

Have to maintain pipeline by which well deploy project to client.

Jenkins open source, github is close source.

Can run Jenkins anywhere either containers or cloud, github is in Microsoft.

Sometimes we need to share credentials I don't want to do it with Microsoft

Diff in continuous deployment and cont delivery –
delivery is manual approach and deployment is automatic approach

ci – build test merge

c delivery –

c deployment –

Jenkins - developed in java, default port – 8080

Jenkins – security add token copy paste in github webhook

connect Jenkins to linux:

- create an instance on aws
- do all sethostname , root stuffs
- open ***Jenkins on aws*** on browser, there under ***Downloading and installing Jenkins*** run all commands
    - **sudo yum update –y**
    - **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**
    - **sudo yum upgrade**
    - **sudo yum install java-17-amazon-corretto -y**
    - **sudo yum install jenkins -y**
    - **sudo systemctl enable Jenkins**
    - **sudo systemctl start Jenkins**
    - **sudo systemctl status jenkins**
- open aws, select instance – security – edit inbound rules – add – custom tcp, 8080, 0.0.0.0/0
- under details of instance, copy public ip, ex - **34.204.50.142:8080**, till 142 is ip and 8080 is default port number.
- Paste this **34.204.50.142:8080** on brower, Jenkins will open
- On terminal - **sudo systemctl status Jenkins** – it will show the path where password is stored.
- **cat /var/lib/jenkins/secrets/initialAdminPassword** - open that path by cat
- insert password in Jenkins platform that we just opened

- Download plugins
- Generate ssh key and paste in github – settings – ssh and gpg keys
- In terminal, **mkdir dirname, cd dirname, git init**
- **git remote add origin <ssh url from your github repo>**
- make a file, add, commit, push
- Add webhooks
    - Github repo – settings – webhooks – under payload url – paste
        - **http://<instance public ip>:8080/github-webhook/**
        - (don't add right now)
    - Jenkins – profile – security – add token – set any name – generate token – copy token – open github – repo settings – webhooks – under secret – paste
    - Content type – ***application json***
    - Select ***just the push event*** then add webhook
- Open Jenkins dashboard – manage Jenkins – manage plugins – check if git plugin is instaled (if not then install it)
- Dashboard – new item – enter job name – freestyle – ok
- Under source code management – git – paste repo https url from github
- Set branch which is your repo's branch name (check by **git branch**)
- Build trigger – tick ***github hook trigger for gitscm polling***
- If Jenkins is running slow when you stopped instance and started it later then :
    - **cd /var/lib/jenkins/**
    - **vim jenkins.model.JenkinsLocationConfiguration.xml**
    - here replace the ip with Jenkins ip address (can copy from address bar of Jenkins tab)
    - **systemctl restart jenkins** then refresh jenkins or Reboot the instance and refresh Jenkins
- Go to manage Jenkins – nodes – open the node (if it shows no space)
    - **df -h /tmp** - to check the sizes
    - **sudo mount -o remount,size=2G /tmp** – increase size immediately
        - writing sudo not mandatory if doing by root
    - but the storage will get back it to its prev size when instance reboots, to make it permanent :
    - **vim /etc/fstab**
    - **tmpfs /tmp tmpfs defaults,noatime,mode=1777,size=2G 0 0** - paste it there
    - **esc colon wq!** - save and quit
    - **sudo mount -a**
    - **sudo systemctl daemon-reload**
    - **sudo systemctl restart jenkins** or reboot the instance
    - now refresh the Jenkins page in browser
- under my project – workspace – if nothing is there then tap *build now* (only for the 1st time) then I can see files of my repo, if you will add/modify files in the repo either by terminal or directly by github, it should automatically build the update.
- *If there is any problem check webhook is connected rightly or not, if not check it and update the right ip from add bar of Jenkins tab in github webhook section.*

Deploy java based application by maven, will deploy tomcat apache server

Host on windows – iis

In linux – apache, httpd, tomcat (developed in java)

We wll work in tomcat java

Dev server – github – jenkin –

BUILD MAVEN PROJECT IN JENKINS

- Do all the setup of linux, github and jenkins
- In Jenkins install plugins
    - ***Maven*** – to run java apps
    - ***deploy to container***
    - ***github integration***
- Isntall git, maven, Jenkins,java in terminal
    - yum install java*
    - yum install maven
    - open jenkins on aws site – run all commands of installation of jenkins
- **rpmquery maven** - to check maven installed and to check its version
- **mvn -v** - shows java and maven home directories
- open jenkins – manage jenkins – tools –
    - under jdk - add jdk – under JAVA_HOME - paste java home dir from terminal
    - under maven installations – add maven – under MAVEN_HOME - paste maven home dir from terminal
- what we did, created a repo, cloned sir's repo indide our repo through terminal
    - **git add remote origin <ssh url of our repo>**
    - **git clone <sir's repo ssh url>**
    - check bt **ls,** if files are cloned then ok if the folder get cloned then
        - **cd <cloned dir name>**
    - **git add .**
    - **git commit -m "xyz"**
    - **git push origin master**
- Add webhooks
    - Github repo – settings – webhooks – under payload url – paste
        - **http://<instance public ip>:8080/github-webhook/**
        - (don't add right now, add after next step)

    - Jenkins – profile – security – add token – set any name – generate token – copy token – open github – repo settings – webhooks – under secret – paste
    - Content type – ***application json***
    - Select ***just the push event*** then add webhook
- dashboard – new item – title – maven – ok – under source code manag. – check git – paste github repo https url – set branch – build trigger - Build trigger – github hook trigger – save – workspace – build now (for the 1st time)

wget, apt, yum are command for installing packages.
wget is for downloading from internet, and remaining from local.

TOMCAT

- Make another server / instance with ***10 gb gp3***
- In security – inbound rules – add – custom tcp - port ***8080***
- Set hostname, login to root
- In terminal – **yum install wget**
- **yum install java* -y**
- Open site from browser – download tomcat 9 – core – ***tar.gz*** – copy link address
- **wget <paste url that we just copied>** - will download the tar (zip) file
- **ll / ls** - to check that file – copy the apache file that just get downloaded
- **tar -xvzf <paste filename>** - will exract the tar file
- **ll** - list all files, one will be red, one blue, red is the tar(zip) file and blue one is the file that we just extracted - we have move that file in /tmp/ because that file is unnecessary now
- **mv <apache red file> /tmp/** - move the file in tmp

Tomcat setup

- **cd <apache filename>** (can see the filename by **ls**)
- **cd bin**
- **ll** – list all files with permission details
- **chmod +x startup.sh**
- **chmod +x shutdown.sh**
- **cd <apache filename>**
- **find -name context.xml** – will see many files
  #comment valve tag sections in below all files
  vim ./webapps/examples/META-INF/context.xml
  vim ./webapps/host-manager/META-INF/context.xml
  vim ./webapps/manager/META-INF/context.xml
- **cd apache-tomcat-9.0.55** – (my tomcat file name should be here)
- **cd conf**

- **vim tomcat-users.xml**
  #Add below lines between <tomcat-users> tag
  <role rolename="manager-gui"/>
  <role rolename="manager-script"/>
  <role rolename="manager-jmx"/>
  <role rolename="manager-status"/>
  <user username="admin" password="admin" roles="manager-gui,manager-script,manager-jmx,manager-status"/>
  <user username="deployer" password="deployer" roles="manager-script"/>
  <user username="tomcat" password="s3cret" roles="manager-gui"/>
- **cd <apache filename> - cd bin**
- **./startup.sh**
- Go to aws – tomcat instance – copy public ip – paste in brower - ***<ip>:8080***
  - Sometimes it has browser issue so can open in chrome incognito
- In browser open ***manager app*** – id password – *admin admin* respectively
- If any problem occurs we can restart tomcat
  - **cd <tomcat filename>** - **cd bin** - **./shutdown.sh** - **./startup.sh**
- Jenkins – open my project - configure – build settings – add post build – deploy war/ear to a container – write ****/*.war** – context path – write **/** (slash) – save
- Manage jenkins – credentials – global – add credentials – username – ***deployer*** – password – ***deployer*** – create
- My project – configure – build settings – post build actions – add containers – ***tomat 9*** – credentials – select ***deployer/******* - tomcat url – paste ***<tomcat server ip>:8080/*** – save.
- Workspace – build now – check if working – github – edit files – commit – check in jenkins