వైకుంఠం

ch. Venkata Siva Rama Krishna

DevOps Notes

9/12/19 to 22/2/2020

DevOps

9/12/19

2

DevOps:- Development and Operations

Devops is addressing all traditional issues and implementing complete Automation in entire company (or) IT industry (or) Organisation

Development

plan
↓
Code
↓
Build
↓
Test
↓
Quality Assurance

Operations

Deployment
↓
Maintainence
↓
Monitor

→ Devops is a software development strategy which bridges the gap between the Dev and the Ops side of the Company

→ Devops is a technical Methodology as a Agile process. Agile process is a continuous process

Needs of the organisation?

Faster Development
Higher Quality
Lesser Spending
Reduced Outages

DevOps is a set of practices that combines Software development and information-technology operations which aims to shorten the systems develop-ment life cycle and provide continuous delivery with high software quality

Problems of Devops Engineers:-

In devops to learn all tools are difficult. We have to maintain the knowledge in all tools and must be perfect in two or more tools

Linux:-

Each and Every tool of devops are installed only on linux

→ To learn and use devops must understood almost all linux commands

AWS:- Amazon Web Service

The Linux machine taken from the AWS

→ We have to know the

Dev Vs Ops
Build Vs Test
Manual Vs Automation
Delivery of whole Product in one go

DevOps Services	Tools
Source Code management	Git, Mercurial,
Continuous Integration	Jenkins, Teamcity, GitLab, Travis CI
Configuration Management	Puppet, CHEF, SALTSTACK
Continuous Deployment	Puppet, docker
Continuous Monitoring	Nagios, Zabbix, PRTG Network Monitor

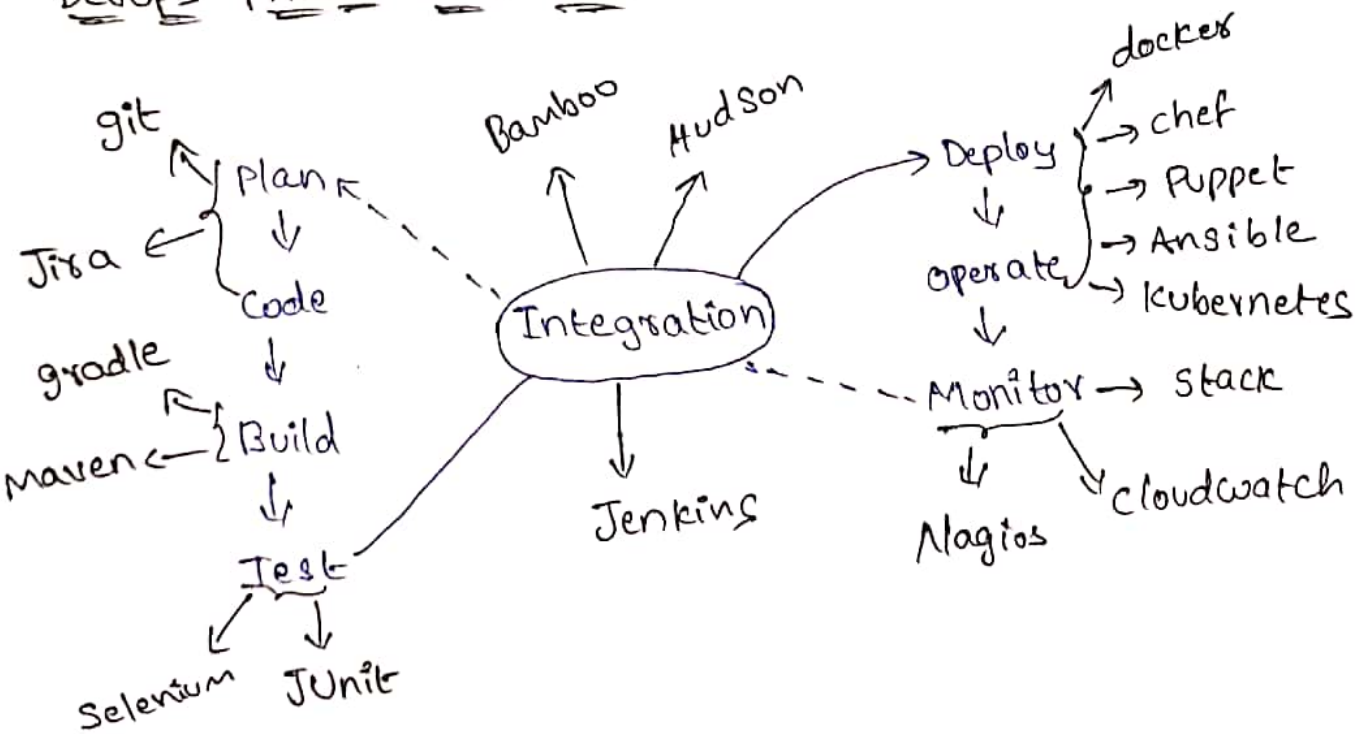
Automation Scripting → Pearl / Python, Ruby, Ansible

DevOps Stages:-

01	02	03	04
Version Control	Continuous Integration	Continuous Delivery	Continuous Deployment
Maintains different versions of the code	Compile, validate, code Review, Unit Testing, Integration Testing	Deploying the build application to test servers Performing UAT	Deploying the tested application on the Prod Server for release
Source Code management	Continuous Build	Continuous Testing	Configuration management & containerzation

Continuous monitoring

DevOps phases and Tools:-



DevOps:-

AWS
↓
Linux
↓
Devops

AWS:- Amazon Web Service

Cloud:-

Cloud computing is the on-demand delivery of IT resources over the internet with pas-as-you-go pricing.

Instead of buying, owning and maintaining physical data centers and servers, you can access technology services, such as computing power, storage and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

Who is using cloud Computing?

Organizations of every type, size and industry are using the cloud for a wide variety of use cases such as data backup, disaster recovery, email, virtual desktops, software development and testing big data analytics and customer-facing web applications.

Ex:-

Health care Companies are using the cloud to develop more personalized treatments for patients. Financial Services Companies are using the cloud to power real-time fraud detection and Prevention. And video game makers are using the cloud to deliver online games to millions of players around the world.

Benefits of cloud Computing:-

→ Agility
→ Elasticity
→ Cost Savings
→

Agility:- The cloud gives you easy access to a broad range of technologies so that you can innovate faster and build nearly anything that you can imagine. You can quickly spin up resources as you need them - from infrastructure services, such as compute, storage and databases, to internet of things, Machine learning, data lakes and analytics and much more.

You can deploy technology services in a matter of minutes and get from idea to implementation several orders of magnitude faster than before. This gives you the freedom to experiment, test new ideas to differentiate customer experiences and transform your business

Elasticity:-

With cloud computing, you don't have to over-provision resources up front to handle peak levels of business activity in the future. Instead you provision the amount of resources that you actually need. You can scale these resources up or down to instantly to grow and shrink capacity as your business needs change

Cost Savings:-

This cloud allows you to trade capital expenses (such as data centers and physical servers) for variable expenses, and only pay for IT as you consume it. Plus, the variable expenses are much lower than what you would pay to do it yourself because of the economies of scale

Deploy globally in Minutes:-

Using the cloud, we can expand to new geographic regions and deploy globally in minutes. Ex:- AWS has infrastructure all over the world, So you can deploy your application in multiple physical locations with just a few clicks. putting applications in closer proximity to end users reduces latency and improves their experience.

Types of cloud Computing:-

The three main types of cloud computing include infrastructures as a Service, platform as a Service and software as a service. Each type of cloud computing provides different levels of control, flexibility and management So that you can select the right set of services for your needs.

Infrastructure as a Service (Iaas):-

Iaas Contains the basic building blocks for cloud IT. It typically Provides access to a networking features, computers and data storage space. Iaas gives you the highest level of flexibility and management control over your IT resources. It is most similar to the existing IT resources with which many IT departments and developers are familiar

Platform as a Service (Paas):- Paas removes the need for you to manage underlying infrastructure, and allows you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry

about resource procurement, capacity planning, 9
Software maintainence, patching, or any of the other undifferentiated heavy lifting involved in running your application

Software as a Service (Saas):-

Saas Provides you with a complete product that is run and managed by the service Provider. In most cases, people referring to Saas are referring to end-users applications. With a Saas offering, you don't have to think about how the service is managed. You only need to think about how you will use that particular software

AWS-Amazon Web Service 10 11/12/19

Search AWS Management Console in Google

click on AWS Management Console

Create a Free Account
(or)
AWS

→ Payment 2 Rupees for AWS account for one year

→ Goto AWS login

AWS Console

→ click on services

EC2

EC2 - Elastic Compute cloud

→ Server (or) Machine (or) PC (or) Box (or) Instance

→ In AWS Machine is called as instance

Servers

1, OS - AMI (Amazon Machine Image)

2, CPU - 1 2 4 128

RAM - 1 4 8 1000

Instance types

3, HD - (Hard Disk) - EBS (Elastic Block Store)

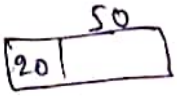
object
block
> Two types of Storages

object storages - MP3, MP4, Excel, Notepad, pics

Block storages - Install OS, Install Databases
(Pen drive, Google drive)

(C drive)

Block - OS, DB (C drive)



Block storage also acts as object

Installing Windows os on AWS || 12/12/19

EC2 (Windows)

EC2 - Launch instance
Windows Server 2012 - Select
T2 - Micro - Next
Number of instances - 1 - Next
Next (storage section)
click to add a Name tag: Windows - Next
Security group name: Windows SG
Description: Windows SG
Source: Anywhere
Review and Launch - Launch
Create a new keypair - demo - Download key pair -
Launch Instances - click on Instance ID
click on InstanceID (Can see Windows,
Machine running)

Connect - Get Password - choosefile - demo.pem (-
.upload) - Decrypt password
Open RemoteDesktopConnection application in
your laptop
Enter all 3 details (DNS name, Username &
Password) - Connect - Yes

→ Select an existing key pair or create a new key pair

Create a new key pair

key pair name

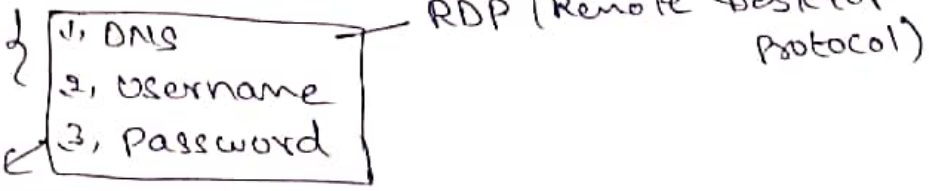
RamDemo → the file in .pem format

download Key Pair

↓

Launch instances

View instances



Not Providing

Click on Connect

↓

Public

username

Password: Get password click on it

key pair path: choose file

Decrypt password

Copy the DNS and user name in RD Connection

Click on Connect

↓

After password

Admin

Click on OK

Click on Yes

Actions

instance status: stop → We can start the machine again

terminate

↙

We Cannot start this machine again

Installing Linux on AWS

Goto AWS Management Console
↓
Log in
↓
Click on EC2 Services
↓
Launch instance

Step-1:- choose an Amazon Machine Image (AMI)
Amazon linux free tier eligible select

Step-2:- choose an Instance type
t2 micro 1 CPU core 1 GB RAM

Step-3:- Configure Instance Details
Number of instances 1
2 when we needed

Step-4:- Add storage
Root drive: 8 GB click on Next

Step-5:- Add tags
click on add a new tag
Mylinux

Step-6:- Configure Security groups
Security group name:
Description:

click on Review & launch
click on launch
launch instances

Selecting an existing key pair or create a new key pair

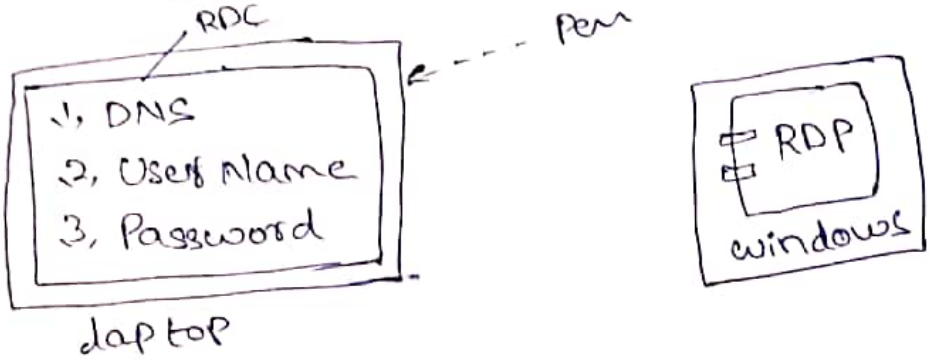
Create a new key pair

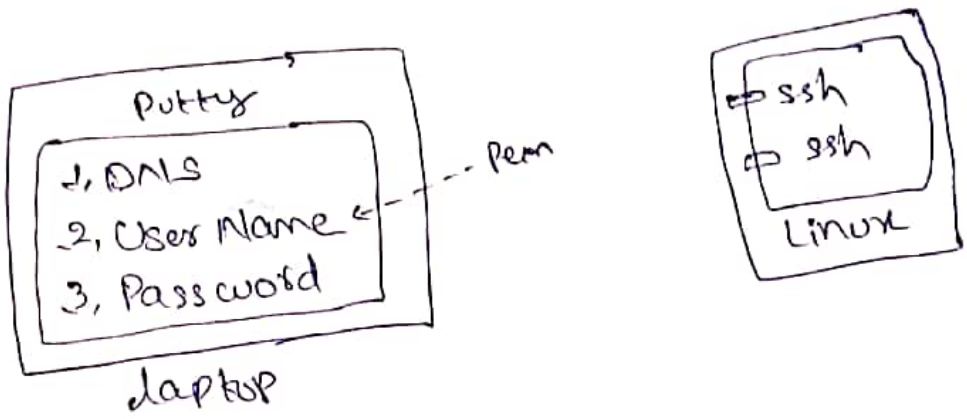
key pair name

9am text

Download key pair
launch instances
view instances

Pem - Private extended
PPK - Putty Private key





Putty:-

Putty gen:-

Google:-

↳ Putty and Puttygen
↳ Putty.exe. 64-bit
↳ Puttygen.exe. 64-bit

Putty generator:-

First open putty gen

click on load → Goto downloads → All files

→ qam test.pem

click on open the qamtest.pem ok + Save Private key

click on yes yes

qam test.PPK Save

Putty:-

Mylinux → click on Connect

ec2-user@ copy

open putty hostname

SSH
↳ Auth [Browse]

qamtest.PPK click on open

EC2 (linux)

EC2 - Launch Instance
Amazon linux - Select
T2 - Micro - Next
Number of instances - 1 - Next
Next (Storage section)
click to add a Name tag: linux - Next
Security group name: linuxSG
Description: linuxSG
Source: Anywhere
Review and launch - Launch
Create a new key pair - demo - Download
Click on Instances (can see linux machine running)

Download putty & puttygen tools
open puttygen tool - load - demo.pem - olc - Save
Private key - Yes - Desktop - demo - Save

Select linux machine - Connect - Copy (user name@dns name)

Open putty tool - paste (user name@dns name) -
SSh - Auth - Browse - demo.PPK - open

Creating Web Server

```
#!/bin/bash
sudo su
yum update -y
yum install httpd -y
cd /var/www/html
echo "My Google" > index.html
ls
service httpd start
chkconfig httpd on
```

1. Goto Security groups
2. Select Inbound
3. Edit
4. Add rule
5. Select HTTP

Creating Load Balancer

Go to Load Balancers - Create Load Balancer - classic Load Balancer - Create

Load Balancer name: MyLB - Next

Select an existing security group: WebSG - Next - Next

Response Timeout: 2

Interval: 5

Unhealthy threshold: 2

Healthy threshold: 2 - Next

Select EC2 (both) - Next - Review and Create - Create - close

Take DNS name of LB and paste in browser (can see content in web page)

Load Balance - Controls the traffic and sends the requests to servers equally.

→ And load balancer checks the server health

17/12/19

1. Linux Machine
2. Web Package
3. Linux Machine
4. Web Package
5. Load balancer

Load balancer worries about files

Creating load balance

1. Application load balancer
2. Network load balancer
3. Classic Load balancer

load balancing

↓

Click load balancer

↓

classic Load balancer

[illegible] [next]

↓

1. Define load balancer
   name: MyLB [next]
2. Security group
   - ○ New
   - ◉ existing [next]

3. Setting Next
4. Health check

Protocol [http ▽]

Port [80]

Path [/index.html]

Advanced Details

Response Time out :- the time between the input request to output request

Response Time out [2] Seconds

Interval [5] Seconds

unhealthy threshold [2] times

Health threshold [2] times

5. Add EC2 instances

6. Add tags

Review & Launch

create

close

Load Balancer

Edit

instances

inService (Running)

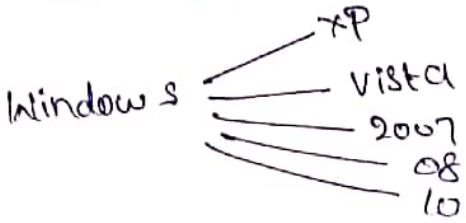
Out-of Service (not Running)

Linux

→ Two Operating Systems are in the world
ie., Windows and
Unix

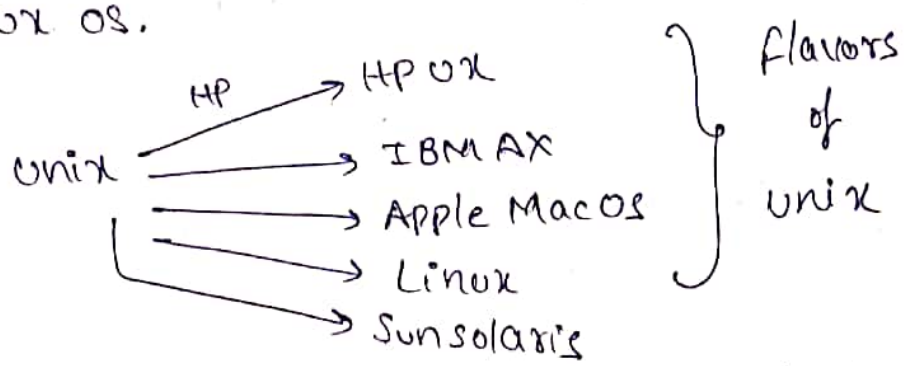
Windows:-

→ Windows Source code is not open to the world
→ It is not free
→ It is Desktop OS
→ 95% versions are same only 5% is change acc-ording to versions



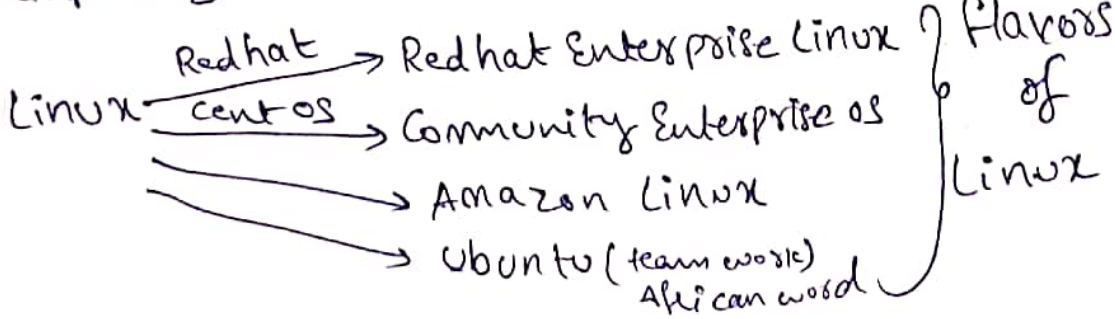
Unix:-

Unlike windows, Unix is free
→ Unix Source Code is open to the world
→ Linus Torvalds modifies the unix and named as Linux OS.



Linux:-

Linux source code is open source and it is completely free



Advantages of Linux:-

1. Open Source
2. Security < hackers → too difficult
   Virus
3. Light weight

   linux - Command line

   windows - Command line + GUI (Graphical user Interface)

In windows Command line is used by the only developers

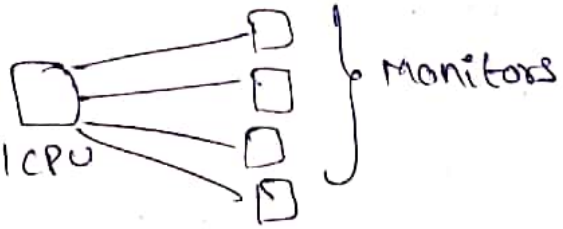
50 GB of Ram windows = 1 GB of Ram Linux

4. Multiuser Multitask

Single user - Single task Ex:- MS-DOS

Single user - Multitask Ex:- Windows 7, 10

Multiuser - Multitask Ex:- flavors of unix & Linux



OS:- Operating System

OS is an interface between the user and the computer hardware

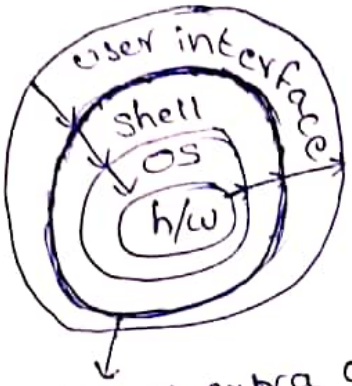
Why linux?

- → Multiuser & Multitasking
- → Open Source
- → Security
- → Needless resources

Architecture of Linux:-

Structure of OS:-

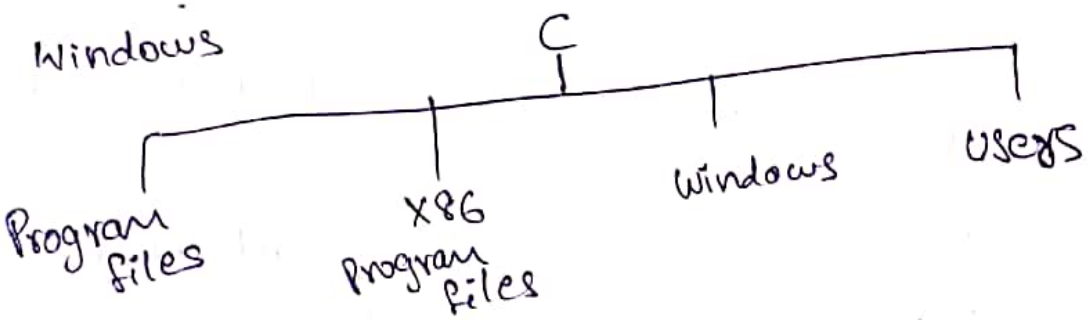
Windows



GUI is extra cover of shell Graphical layer

Linux

File System hierarchy:-



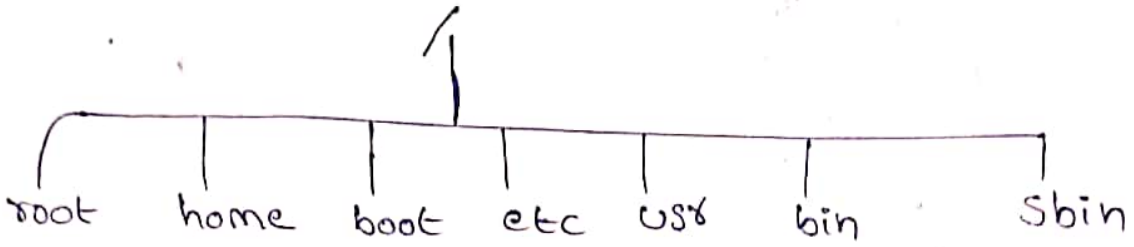
Windows	Linux
Folder	Directory
OS	Kernel
Software	Package
Administrator	Root user

Linux

/ → Top level root directory

/root/krishna → Separations

Top level root directory



/root - It is home directory for root user

/home - It is home directory for other users

/boot - It contains bootable files for Linux

/etc - It contains all configuration files

/usr - By default softwares are installed in this directory

/bin - It contains commands used by all users

/sbin - It contains commands used by only root user (root)

** Linux is a Case Sensitive

Case Sensitive:-

Differentiating between capital and lower-case letters

Ex: Computer and computer are two different words

→ sudo su

sudo - Super user do

su - switch user

It converts ourselves as a root user

→ yum update -y

It updates the system os

→ clear

The clear command is used to clear the screen

shortcut key:- ctrl+l

Creating files in linux:-

→ Linux not have any extensions like .txt, .pdf

→ Linux treats the extensions as a filename

Commands to creates the files:-

1, cat

2, touch

3, nano

4, Vi/Vim

Cat:- The cat command is

→ used to create files

→ is used to append the information (or) data to the files

→ is used to view the content in the files

1, cat > filename

This is used to create the file

Ex: cat > file1

Haii

→ To come out of the file using ctrl+d

→ To adding the data to the file is

cat > file1↵

To overwrite

cat >> file1↵

To append (or) adding the data

cat file↵ To view the content in the file

2, touch:- Touch command is used to create blank files.

touch file2↵

→The touch files are used for testing purpose

To create multiple touch files

touch filea fileb filec↵

3, nano:- nano command is used for creating and editing the file

nano file3↵

To come out of the file ctrl+x

To save ctrl+y↵

To not save ctrl+N

To add data to nano files

nano file3

4, Vi/vim:- Vi file4↵

for insert press i to ~~add~~ Add the content

To Command ~~Command~~ mode esc

To come out esc: w to save, q to quit,

! - force quit

cat - To create, to append, to view

touch - To create empty files

nano
vi/vim } - To edit the content of the files and to create the file

Directories:-

To create the directory using the mkdir

mkdir:- mkdir is used to create the directory

mkdir dir1

ls Command:- the ls command is long lists the files and directories present in the system

drwxr-xrw-r

Here, d indicates it is directory

→ To create hidden files for safe the data the hidden files and hidden directories are creation is simple.

→ the dot is place before the creation of file name (or) directory name

create hidden files

touch .filename

(or)

cat .filename

create hidden directories

mkdir .dir2

→ To see the hidden files or hidden directories

ls -a here a indicates the all files

→ To list all the normal files and hidden files (or) normal directories and hidden directories

ls -al (or) ls -la

To create multiple files (or) multiple directories:-

Creation of multiple directories

mkdir dirc dird

the above example creates the directory side by side

mkdir -p dira/dirc/dire

the above example creates the directories one inside another

Copy Command:- The copy command copies the information (or) data from file to file (or) file to directory (or) directory to directory

CP Source destination

Source - Present file place (or) name

destination - where to be file will placed

move:-

move command is used to moves the files from one place to another.

mv Source destination

mv file2 dir1

Rename:- Rename is works to changes the name

mv file4 myfile

It changes the name of the file

rm remove:-

Remove command is removes the files and directories from the system.

rm -rf myfile

-rf = recursively forceful

Remove a single directory

rm -rf dir1

Remove all files and directories

rm -rf *

Linux Basic Commands for DevOps:-

- → cat (create & append files)
- → touch (create blank file)
- → nano (create & edit file)
- → vi/vim (create & edit file)
- → ls (list) (-a, la)
- → cd (change directory)
- → pwd (Print working directory)
- → mkdir (create directory multiple)
- → cp (copy)
- → mv (move)
- → mv (rename)
- → tree (see the files & directories in tree structure)
- → rm -rf (remove files & directories forcefully)
- → grep (Pick & print)
- → less (see output)
- → head (see top 10 lines)
- → tail (see last 10 lines)
- → sort (display in alphabetic/Numeric order)
- → user
- → Group
- → Softlink (Shortcut)
- → Hardlink (backup)
- → tar (to pack)
- → gz (to Compress)
- → yum (to install)
- → wget (to download)
- → File/directory Permissions:
  - chmod (permissions)
  - chown (owner)
  - chgrp (group)

hostname (to see hostname)
ifconfig (to get ip address)
cat /etc/* rele* (to get os version)
yum install httpd (to install package)
yum update httpd (to upgrade package)
yum remove httpd (to uninstall package)
yum list installed (to see installed packages)
Service httpd status (to see status)
Service httpd start (to start Service)
Service httpd reload (to restart service)
Service httpd restart
chkconfig.httpd on (to start service permanently)
chkconfig.httpd off (to stop service permanently)
Redirection (redirecting output)
which (to see package installed or not)
sudo (to get root Priviliges)
whoami (to see user)

Find Command:-

→ find -type f (to see all files in current directory)
→ find -type d (to see all directories in current directory)
→ find / -type f (to see all files under top level root directory)
→ find / -type d (to see all directories under top level root directory)
→ find / -type f -name <file_name> (to search specific file under top level root directory)
→ find / -type d -name <dir_name> (to search specific dir under top level root directory)

Grep Command:-

The grep command which stands for "global regular expression print".

By default, grep displays the matching lines

Use grep to search for lines of text that match one or many regular expressions, and outputs only the matching lines.

Using grep for Search files

grep is a powerful file pattern searchers in Linux

less:- The less command is used to see the output line wise or page wise

Note:-

Press Enter key to scroll down line by line

Use d to go to next page

Use b to go to previous page

Use / to search for a word in the file

Use v to go Vi mode where you can edit the file and once you save it you will back to less Command

Press q to quit from the Prompt

more:- The less Command and more Command are almost Same

head:- It is used to display the top 10 lines of the file

By default it will top 10 lines

To see top 5 lines of the file

```
head -5 /etc/passwd
```

tail:- It is used to display the last 10 lines of the file

To see last 5 lines of the file

```
tail -5 /etc/passwd
```

Sort:- This command is used to sort the output in numeric or alphabetic order

```
sort filename↵
```

hostname:- hostname is the machine name

```
# hostname
```

IPaddress:- To know the ip address of the System

```
ipconfig in windows
ifconfig in Linux
hostname -i (it gives the ip address)
```

OS-release:- OS information is stored

```
cat os-release
cat /etc/os-release
```

* represents the all

```
cat /etc/os-rele* (It open starts with os rele)
cat /etc/*release (It opens any file ending with release)
cat /etc/*rele* (It opens the word rele)
```

To install packages:-

yum install httpd -y

To update packages:-

yum update httpd -y

To know the installed packages:-

yum list installed

which:-

Linux which command is used to identify the location of a given executable that is executed when you type the executable name (command).

which httpd
which tree
which mysql

To remove packages:-

yum remove httpd -y

Again install:-

To know the package is running or not

Service httpd status

To start:-

Service httpd start

To Restart:-

Service httpd restart

restart (or) reload

To stop:-

Service httpd stop

To stop the service of the httpd

chkconfig:-

Automatically starts the service when system restarted also.

chkconfig httpd on

To off:-

chkconfig httpd off

Redirection:-

echo "Hello" > filename

The filename is already existed the content is stores in the file..

the filename is not present it creates the file

echo "Hello" >> filename

to append the data

whoami:- It will give the user

To shows the type of the user

Find:-

find / -type f :- To see all the files in the System

find -type f :- To see the files in Current directory

find / -type d :- To see all the directories in the System

find -type d :- To see the directory in the current directory

find / -type f -name passwd

find / -type f -name os-relase

find / -type d -name dirb

find / -type d -name dir*

Shortcut keys:-

→ To clear the Screen ctrl+l

→ To repeat the previous Command ↑ (UP Arrow)

→ without executing the Command ctrl+C

→ To come out of the cat file ctrl+d

→ Use the "tab" button automatically appears the file name

We want to copy the content in linux highlight the wanted information and click on the cursor right button

→ To come out of the nano : ctrl+x

→ Creating directories & removing
mkdir -p dir1/dir2 & rm -rf

→ Sort: Contents inside file (sort <filename)

→ Softlink: ln -s <mainfile> <linkfile>

→ hardlink: ln <mainfile> <linkfile>

→ Adding single/multiple users to group: a - single M - Multi user
gpasswd -a/M <user>, <user1> <group>

→ Removing users from group:
gpasswd -d <user>, <user1> <group>

→ tar: tar -cvf <new.tar> <old>,
tar -xvf <new.tar> (du -h file name)
du - disk usage

→ zip: gzip <new.tar>, gunzip <new.tar.gz>

Adding user:-

→ useradd is used to add the user

passwd

cat /etc/passwd

Creating group, adding users to group:-

→ groupadd devops

group

cat /etc/group

→ gpasswd -a krishna devops

username groupname

To adding the single user to the group

→ gpasswd -M siva, hari, ajay devops

To add the multiple users to the group

Deleting users from group:-

→ gpasswd -d raj devops removing the user from that group

→ userdel username to delete the single user

→ groupdel groupname to delete the group

Softlink:-

ln - link

ln -s hari softhari (hari → main file, softhari → link file) in same directory

ln -s /etc/hari softhari in other directory

cat softhari

To view

cat >> softhari

To append

ls -l

softhari → hari

rm -rf harifile
deleting the original file

cat softhari
No such file or directory

→ Main file is, deleted then softlink file is also deleted then it is not present

Softlink is shortcut
Hardlink is back copy

ln -s indicates softlink

without -s indicates hardlink

Hardlink:-

ln – link

ln file3 hardfile3 → filename
ls
cat >> hardfile3
rm -rf file3
cat hardfile3

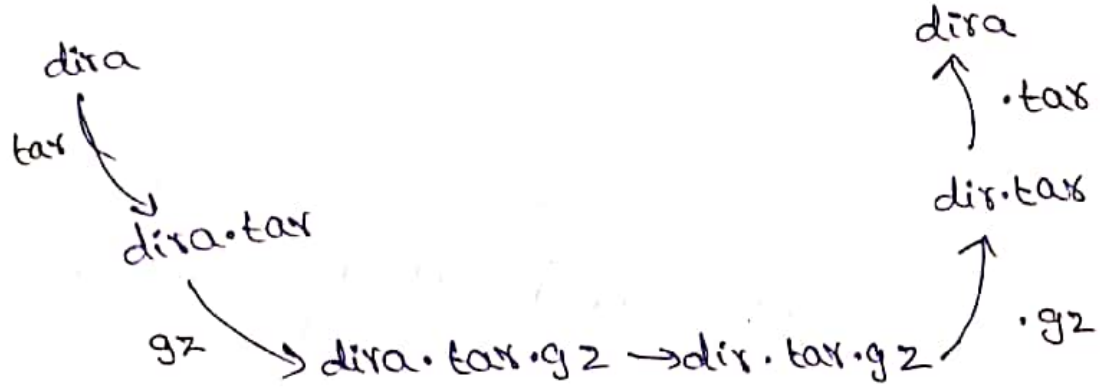
The file "file3" is deleted (or) removed hardfile3 contains the information

* → In windows zip is a file.
→ the zip file is to compress and share

→ In linux two concepts ie., two commands.

→ tar
→ gzip

→ tar command is used to pack
→ gzip command is used to compress



→ tar -cvf dira.tar dira
file directory

c - create
v - verbous
f - forcefully

→ gzip dira.tar
→ ls
→ dir.tar.gz
→ rm -rf dira

To remove gzip:-

gunzip dira.tar.gz
ls
* dira.tar

To remove tar:-

tar -xvf dira.tar
x - extract
v - verbous
f - forcefully

Wget:-

Wget Command is used to download the packages to the linux

Wget paste the link (package name)

Yum:-

Yum Command is used to install the packages

yum install chef (Press the tab button to the -y Continue address)

Which:-

rm:- rm -rf *

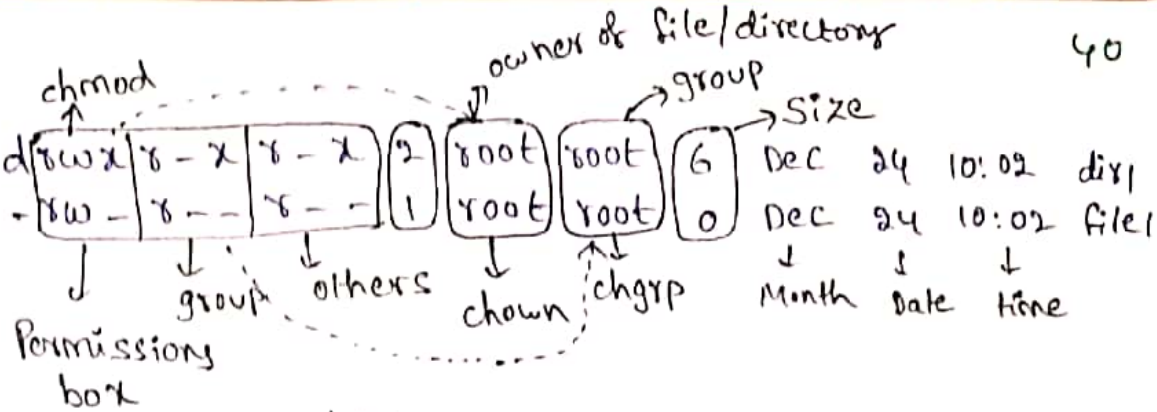
The above Command is used to delete all the files (or) directories (or) packages in Single Command.

Creating file & directory:-

touch file1

mkdir dir1

ls -l



r - read (4)
w - write (2)
x - execute (1)

read-only seeing purpose can't write (or) can't edit
write - to edit purpose

rwx	r-x	r-x
421	4 1	4 1
7	5	5

→ 755 default directory Permissions

rw-	r--	r--
42	4	4
6	4	4

→ 644 default file Permissions

chmod:- This command changes the permissions

chmod 777 file1 → file name (or) directory name
re, wr, ex
666 group re, w, group
444 re others

To change the ownership of the files:-

chown hari file1
chgrp devops file to change group

chmod - To change permissions
chown - To change ownership chown hari file1
chgrp - To change group chgrp devops file1

Git

26/12/19

SCM (Source Code Management)

→ Storage (Any code)
- Developers (Code)
- Testing team (Test)
- Devops (Scripts)

→ Different people from different teams can store simultaneously (Save all changes Separately)

→ Pipeline between off shore & on shore teams

→ Helps in achieving team work

→ Track changes (Minute level)

Git is for everyone, it is not a core devops tool

SCM Tools:-

- Git (Most advanced tool)
- SVN
- Perforce
- clearcase

Important Terminology:-

- Depository (git) / Depot (SVN)
- Server Mymachine
- WorkSpace / workdir / work tree (where you create files to work)
- Branch (git) / Trunk (SVN) / Codeline
- Commit (git) / check-in (SVN) (Moving storage to local repository)
- Version / Version-ID / Commit-ID (40 long alpha-Numeric characters)
- Tag (Nothing but a name)

→ Repository
  • Storage (folder)
→ Server (EC2 instance)
  • stores all repository

Branch V.V. imp Section
→ Product is same, so on repository, but different tasks
→ Each task has one separate branch
→ finally merge (code) all branches
→ for parallel development
→ Can create any number of branches
→ changes are personal to that particular branch
→ Can put files only in branches (not in repo directory)
→ Default branch is "Master"
→ files created in workspace will be visible in any of the branch workspace will you commit. Once you commit, then that file belongs to that particular branch

Workspace / working Directory :-
  • where you work
  • where you see files physically & do modifications

Version / version-ID / Commit-ID :-
  Reference to identify each change and who did that change

Tag :-
  Meaningful name (We cannot remember Commit-ID)

Why Only Git

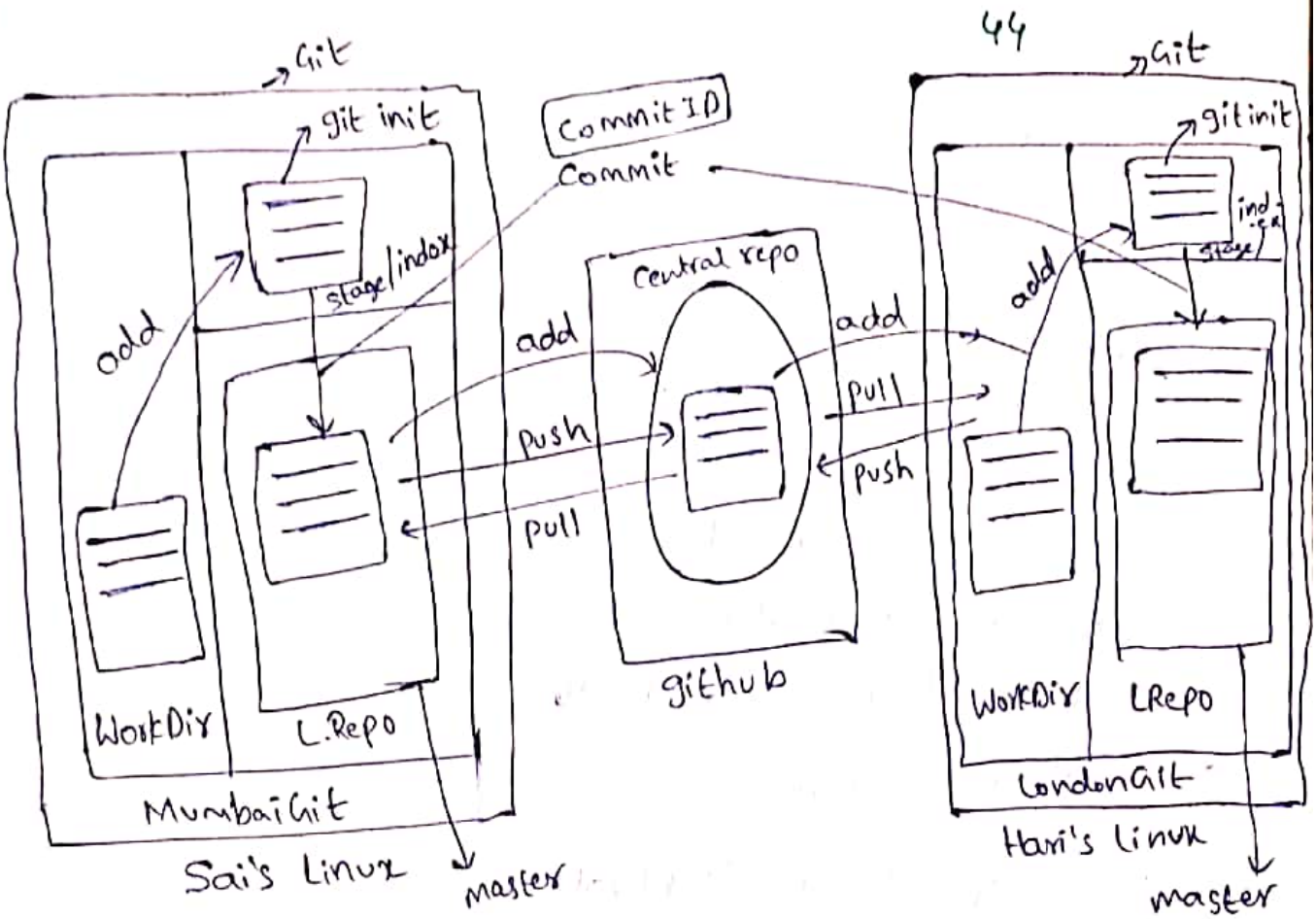
Git Advantages:-

→ Speed
- snapshots Concept

→ Parallel branching
- Multiple branches at a time unlike other scm tools

→ Fully Distributed
- Backup copy is available in multiple locations
- No need internet connection. So no network latency.
- No need central server Separately
- Each work space will have its own repo internally
- Can create any number of branches
- Can share code without using central repo
- That's why we call GIT as DVCS

DVCS - Distributed Version Control System



27/12/19

Git - DVCS - Distributed Version Control System
SVN - CVCS - Centralized Version Contool System

Snapshots:-

→ Get any Previous Version (Backup) 11A
→ Represents Some data of Particular ~~data~~ time
→ Stores the changes (append data) only. Not whole copy

Commit:-

→ Store changes in repo (will get commit ID)
→ 40 Alpha-Numeric characters
→ Concept - checksum (It's a tool in linux generates binary value equal to data present in the file)
→ Even if you change one dot, Commit-ID will be changed
→ Track the changes

Git Stages

→ Work Space
- physically see file & modify

→ Staging / Indexing area
- Buffer area
- Takes snapshot

→ Repository (Local)
- Store changes locally

→ Repository (Central)
- Store changes centrally

Types of Repositories:-

→ Base Repositories (central)
- Store & share only
- All central repositories are base repositories

→ Non-Base Repositories (Local)
- where you can modify the files
- All local repositories are non-base repositories

Install & Configure git:-

→ launch 2 EC2 instances in 2 regions (Mumbai & London). Run below commands in both machines

→ Sudo su
→ yum update -y
→ yum install git -y
→ git --version
→ git config --global user.name "sai/Hari"
→ git config --global user.email "sai/hari6cs@gmail.com"
→ git config --list
→ date
→ date +%.T -s "20:58:00"

cs group → SSH

Git Hub (Central Repository)

→ Goto www.github.com & create account
→ Create a new repository (central git) & choose public

Git Commands (in Mumbai EC2 machine)

→ Create Directory & go inside that
→ mkdir mumbaigit, cd mumbaigit
→ git init (to initialize git)

• Create new file, see status, put in staging area & commit into local repo

→ touch myfile
→ git status
→ git add .
→ git commit -m "1st commit from mumbai"
→ git log
→ git show <commit-id>

Git Commands (in Mumbai EC2 machine)

→ git remote add origin <central git repo url>
→ git push -u origin master (give github credentials)

. Verify in github. Can see the file

→ (add some content to the file and repeat the same process. But need not to add again to central repo)
→ (observe the differences b/w untracked file & modified file)
→ git log
→ git show <commit-ID>

Git Commands (in London EC2 instances (or) machine)

→ Create directory & go inside that
- →mkdir londongit
- →git init (to initialize git)
- →git remote add origin (Central git repo url)
- →git pull origin master
- →git log
- →git show <commit-ID>
- →cat >> myfile (append with some content)
- →git status
- →git add
- →git commit -m "1st commit from london git"
- →git push -u origin master

(or)

git push origin master

-u is not mandatory

To pull latest changes

→git pull origin master

Again append with some content add, commit & path

- →cat >> myfile (append with some content)
- →git status
- →git add
- →git commit -m "Any commit msg"
- →git push origin master

Untracked file modified file

Untracted file

Modified file

2/1/2020

Git Commands

To ignore the file while Committing (Retrospective effect is here)

- **.gitignore:-**

Steps:-

→ Create .gitignore file and give pattern matching
ex: *.class

→ Add and Commit .gitignore file

→ Create Some class files and java files for testing and them by running "git add."

git add - add only one file
git add . - adds all files

Practical:-

```
ls
cd mumbaigit/
ls
vi .gitignore↵
*.class
git add .gitignore
git Commit -m ".gitignore"
ls
touch 1.txt 2.txt 3.class 2.class
ls
```

git status
git add . ↵
git status
git commit -m "txt"
touch 3.class
git status

Git log options

git log

It shows all commits

git log -1

Ex: git log -2

It shows 2 latest commits

q - quit

git log --oneline

(It shows summarize manner)

To pic Commit based on Commit msg

git log --grep "any word of Commit msg"

To See the Content of Particular Commit

git show <Commit-ID>

Ex: git log --grep "london"

It shows the full details of the Commit

Branching

To see list of available branches (most have some thing)

→ git branch

Create a new branch

→ git branch <branchname>

To switch branch
→ git checkout <in which branch you want to go>
→ verify the content in both branches
→ Add content in both branches
→ files are not personal to branch until you commit

Branching (merge)

→ You cannot merge branches of difficult repositories
→ To merge branches (Pulling mechanism)
→ git merge <branch name>
→ verify the merge
→ git log

3/1/2020

Push to central repository
→ git push origin master

Merging is a pull mechanism → Copy & paste
git branch (Practical)
git log --oneline
git merge newbranch
ls
git log --oneline
git branch
git push origin master

username:
Password:

Git Conflict

→When same file having different content in a different branches, if you do merge, conflict occurs (Resolve Conflict, then add and Commit)

→Conflict occurs when
  →merging branching

Git Stashing

→It removes content inside file from working directory and puts in stashing store and gives clean working directory so that we can start new work freshly

→later on you can bring back that stashed items to working directory and can resume your work on that file

→To stash an item (only applies to modified files. Not new files)
  •git stash

Practical

```
git branch
git checkout newbranch
ls
vi saifile
hello
ESC:wq!
git add .
git commit -m " Commit into new branch
git checkout master
vi saifile
hai
ESC:wq!
```

```
git add .
git commit -m "commit into master"
git branch
git merge newbranch
vi saifile
<<<< HEAD
Hi
=====
hello
>>>> newbranch
```

Esc:wq!

```
git add.
git commit -m "commit new branch"
git log --oneline
```

Git Stash

→ To see stashed items list
  → git stash list
→ To apply stashed items
  → git stash apply stash@{number}
→ Then you can add & commit
→ To clear the stash items
  → git stash clear

steps to be followed

→ create any blank file add and commit
→ put some content, then run "git stash" (repeat twice)
→ Goto stash repo for verification
→ Get from stash repo and then add and commit (if conflict occurs, resolve conflict)

HEAD - repersents latest Commit
Branch - To implement new idea
Stash - Temporary Repository

No time (or) expiring

```
git branch
ls
touch harifile
git add.
git commit -m "harifile"
vi harifile
   first task
esc:wq!
git stash↵
cat harifile↵
vi harifile
 second task.
esc:wq!
git stash
vi harifile
git stash list
git stash apply stash@{1}
cat harifile
vi harifile
  first task
  finished
  git add.
  git commit -m "first task"
  vi harifile
git add .
git commit -m
git stash clear
git stash list
git shows
```

12 tools are in DevOps
2 (or) 3 are Main in Real Time

Resetting changes (Before Commit)

→ To reset from staging area
  →git reset <filename>
  →git reset

→ To reset the changes from both staging area and a working directory at a time (not individually)

→git reset - hard

Untracked file - which file is not added and committed

→ git reset --hard

Reverting changes (After Commit)

→ To revert the changes
  →git revert <commit -id>

→Revert will create a new commit -id (all changes will be tracked)

→You need not to add and commit again. Automatically new commit - ID will be generated

git show (Commit id) 7 letters of Commit id

Removing files:-

To Remove files
→ git rm <filename>
(Then Commit (no need to add))
(Gets deleted from that Particular branch)

Delete all Untracked files:-

→ git clean -n (dry run)
→ git clean -f (forcefull)

⇒ Files are deleted even commits are not deleted. Commits are safe in the central repository & is Present in the github

Creating 10 files at a time:-

touch file{1..10}
file 0 to file 10

Creating 100 files at a time:-

touch file{1..100}
file 0 to file 99

git status command to know the status

git status
git clean -n

Tag:- Tag is a Meaningful name

→ To apply tag
→ git tag -a <tagname> -m <tag messages> <commit-id>

→ To see the list of tags
→ git tag

→ To see Particular Commit Content by using tag
→ git show <tagnames>

→ To delete a tag
→ git tag -d <tagname>

1. git tag -a test -m my_imp_commit 1c8809c (Commitid)
2. git tag - To show the tags
   (or)
   git log --oneline
3. git show test
4. git tag -d test
   git log --oneline

Github(clone) clone - Taking the Complete Copy

Goto www.github.com
Choose an existing repository
Run below command in local machine's Current directory

git clone <url of github repo>
Go inside that repo
Then onwards, we can push & pull ---

GitHub (Branches & Merging)

- Create new branch from master (new branch)
- Switch to new branch, create an new file & Commit in new branch
- observe the differences between master branch and new branch
- To merge branches in github itself, click on
  - New Pull request
  - Create pull request
  - Merge pull request
  - Confirm merge
- Goto master branch. You can find new file that is Created in newbranch

BitBucket - Atlashe
GitHub - Microsoft
} Both are same tools but Companies are different

add
Commit
Push
Pull

CHEF

Configuration Management:-

→ It's a method through which we automate admin tasks
→ Configuration management tool turns your code into infrastructure
→ So your code would be testable, Repeatable & Versionable

IT infrastructure refers to the Composite of:-

→ Software → Network → People → Process

CM

1, Users
2, groups
3, Permissions
4, files
5, Folders
6, Packages
7, Services
8, Trouble shooting

→ Manual — Admins — Windows / Linux
→ Automatic — Automatic — Devops — chef / Ansible / Puppet / saltstack — CM Tools

IAC - Infrastructure As Code

1, Repeatable Code
2, Code is Versionable
3, Code is testable

Pain Points:-

→ Managing user & group accounts
→ Dealing with packages
→ Taking backup
→ Deploying all kinds of applications
→ Configure Services

Why CM Tools?

→ Complete Automation
→ Increase uptime
→ Improve Performance
→ Ensure compliance
→ Prevent Errors
→ Reduces cost

Why chef

→ chef is an administration tool. whatever System admins (Linux/windows) used to do manually, now we are automating all those tasks by using chef (Any CM Tool)

Ex:- Water Pots

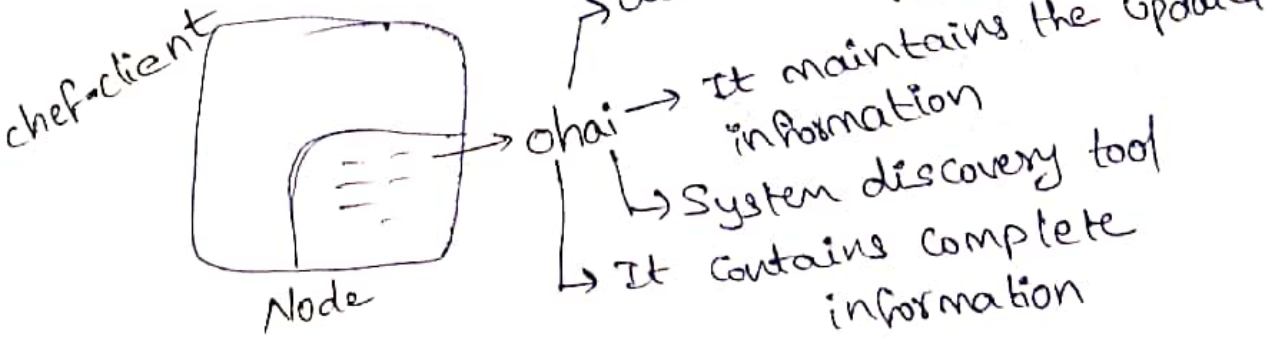
→ Can use this tool whether your servers are in on-Premises or in the cloud

→ It turns your code into infrastructure ie., your Computing environment has some of the same attri-butes as your application

- → Your code is versionable
- → your code is repeatable
- → your code is testable

→ you only need to tell what the desired configuration should be, not how to achieve it

→ Through automation, get desired state of Server



chef super market

↓

Custom Cookbook

Important Terminology:-

→ chef workstation - where you write code
→ chef server - where you upload code
→ chef Node - where you apply the code
→ knife - Tool to establish communication among workstation, Server & node
→ chef-client - Tool runs on every chef node to pull code from chef server
→ ohai - Maintains current state information of chef node (System Discovery Tool)
→ Idempotency - Tracking the state of system resources to ensure that the changes should not re-apply repeatedly
→ chef supermarket - where you get Custom code

Install chef:-

→ Goto chef.sh
→ Goto Downloads
→ click on chef workstation
→ Goto Red Hat enterprise linux
→ copy url on RHEL-7
→ verify chef installation
→ chef -v
→ chef --version

Cookbook:-

→ Cookbook is a group of recipes and some other files & folder

→ Each cookbook define a scenario

Ex:- Web Server Cookbook

→ Create a new directory (Don't change the name of directory)

.mkdir cookbooks

→ Go inside & create a cookbook (whatever you do, do inside cookbooks folder)

.chef generate cookbook test-cookbook

.tree test-cookbook

Inside Cookbook??

→ chefignore: like .gitignore

→ kitchen.yml: for testing cookbook

→ Metadata.rb: name, author, version.. etc... of cookbook

→ ReadMe.md: info about

→ Recipe: where you write code

→ spec: for unit test

→ Test: for integration test

Create your first Cookbook & Recipe

→ Already created cookbook (test-cookbook)

→ Go inside test-cookbook (But not inside recipe folder) & create a recipe.

- chef generate recipe test-recipe

→ Vi test-cookbook/recipes/test-recipe.rb

```
file '/myfile' do
  Content 'Hello Dear student!!'
  action: create
end
```

→ To verify the code (be inside cookbook folder)

- chef exec ruby -c test-cookbook/recipes/test-recipe.rb

→ Apply that recipe locally (be inside cookbooks folder)

- sudo chef-client -zr " recipe[test-cookbook::test-recipe]"

→ verify

- ls/

Create and Write your Second Recipe

→ Go inside cookbook (But not inside recipes folder) & create a recipe

- chef generate recipe demo-recipe

→ vi test-cookbook/recipes/demo-recipe-rb

```
package 'tree' do
  action:install
end
file '/myfile2' do
content 'this is my second file'
```

Deploying an Apache web server

→ Go inside cookbooks folder & create a cookbook
.chef generate cookbook apache-cookbook

→ Go inside cookbook (But not include recipes folder) & create a recipe
.chef generate recipe apache-recipe

→ Be inside apache-recipe

```
package 'httpd' do
 action:install
 end
file '/var/www/html/index.html' do
  content 'Hello Dear students!!'
  action: create
 end
service 'httpd' do
 action [:enable, :start ]
 end
```

→ Apply that recipe locally (be inside cookbooks folder)

```
sudo chef-client -zr "recipe[apache-cookbook::
      apache-recipe]"
```

verify: paste Public IP in browser.

```
action: create
owner 'root'
group 'root'
end
```

→ Apply that recipe locally (be inside cookbook folder)

• Sudo chef-client -z 'recipe[test-cookbook::demo-recipe]"

→ Verify

ls/

Part 2

Ohai

→ It is a System discovery tool
→ It gathers System information
  • Ohai
→ Now run
  • Ohai ipaddress
  • Ohai hostname
  • Ohai memory/total
  • Ohai cpu/0/mhz

Attributes

→ We have a web application to be deployed into 1000 nodes & we need to know some details of each server
→ Because we need to mention that in configuration file of each node. Then information is vary from System to System
→ These details we call "Attributes"
→ chef-client tool gather these attributes from ohai store and put in configuration file
→ Instead of hard coding these attributes, we mention as variables

```
file '/robofile' do
  content " this is to get Attributes
```

```
HOSTNAME:#{node['hostname']}
IPADDRESS:#{node['IP address']}
CPU:#{node['CPU']['0']['mhz']}
MEMORY:#{node['memory']['total']}
owner 'root'
group 'root'
action: create
end
```

Execute

To execute Linux Commands

```
execute "run a script" do
  command << EOH
    Mkdir /saidir
    touch /saifile
  EOH
end
```

User & Group

- We can create users and Groups

```
user "raj" do
  action: create
  end
```

Verify: cat/etc/passwd

chef client

→ We run chef-client to apply recipe to bring node into

→ We call this process as "Convergence"

Runlist

→ Run the recipes in a sequence order that we mention in Run list

→ To apply each recipe from each cookbook (not multiple recipes from same cookbook) at a time by using chef-client (be in cookbooks folder to run this below command)

- Sudo chef-client -zr "recipe[cookbook-name::recipe_name]"
- Sudo chef_client -zr "recipe[cookbook-name::recipe_name],recipe[cookbook_name::recipe-name]"

Include_recipe:-

→ To call recipe/recipes from another recipe with in same cookbook

→ To run multiple recipes from same cookbook

→ Here Comes the default recipe into action (

→ Add below line in default.rb

```
include recipe "cookbook::recipe"
```

→ Now run (Be inside cookbooks folder)

- Sudo chef-client -zr "recipe[cookbook name::recipe_name]"

chef Server

→ chef server is going to be a hub for the code (cookbooks)

→ Nodes pull code (cookbooks) from chef-Server

→ Sign up for chef-Server account

→ Attach your workstation to chef Server

→ upload to cookbook to server
→ Bootstrap nodes
→ Apply cookbooks to nodes

Goto manage.chef.io. for getting server

10/1/2020

Getting started with chef Server

→ Sign up in https://manage.chef.io
→ Get verification email. So get verified
→ Create a new organization
- Give full name
- Give short name (must be unique) (we get url on this short name)
- Now you can see the dashboard

→ To establish communication between chef Server & workstation
→ Goto
- Administration
- Organization
- Download starter-kit
- Unzip it. You can see chef-repo
- Move chef-repo to chef workstation (under EC2-user directory) (by using winscp tool)
- See the components of chef-repo
- To verify whether communication establi-shed or not (be include chef-repo)
- knife ssl check

Bootstrap a Node

→ Attaching a node to chef server is called Boot strapping (make sure, both workstation & noze are in same Az)

→ Now onwards, you have to be inside chef-repo directory to run any command

→ Two action will be done while bootstrapping
- Adding node to chef-server
- Installing chef-package

(copy pem key of node into chef-repo folder in as workstation)

knife bootstrap <node-private-IP> -ssh-user ec2user -sudo -i <pem> -N <any node-name (avoid-hyphen node name)>
- Bootstap 2 nodes
- To see bootstrapped nodes
  - knife node list

upload Cookbook to Server and apply to nodes

→ All cookbooks (apache & test..) must be inside cookbooks folder comes by default with chef-repo folder to upload to chef-server

→ Never ever delete the default cookbooks folder comes by default with chef-repo folder

→ To upload Cookbook to chef-Server

knife cookbook upload <cookbook>

→ To verify cookbook uploaded or not (to see list of Cookbooks which are present in chef server)

- knife cookbook list

→ To add cookbook's recipe to node (see run list) 7L
(do if both nodes)

- knife node run list set <node name>
  "recipe[cookbook recipe]" (to remove "remove")

→ To verify whether cookbook's recipe added to node or not (do if both nodes)

- knife node show <node name>

→ Go inside node & run (do it to both nodes)

- chef-client

Create the Node:-

Instance
subnet 1b

3. Instance
user details #!bin/bash
sudo su
yum update -y

chef-repo]# knife bootstrap <node-Private-IP> --ssh -user ec2-user --sudo -i <.pem> -N <any-node-name (avoid hyphens in node name)> → pem file name

- pem file name to chef repo

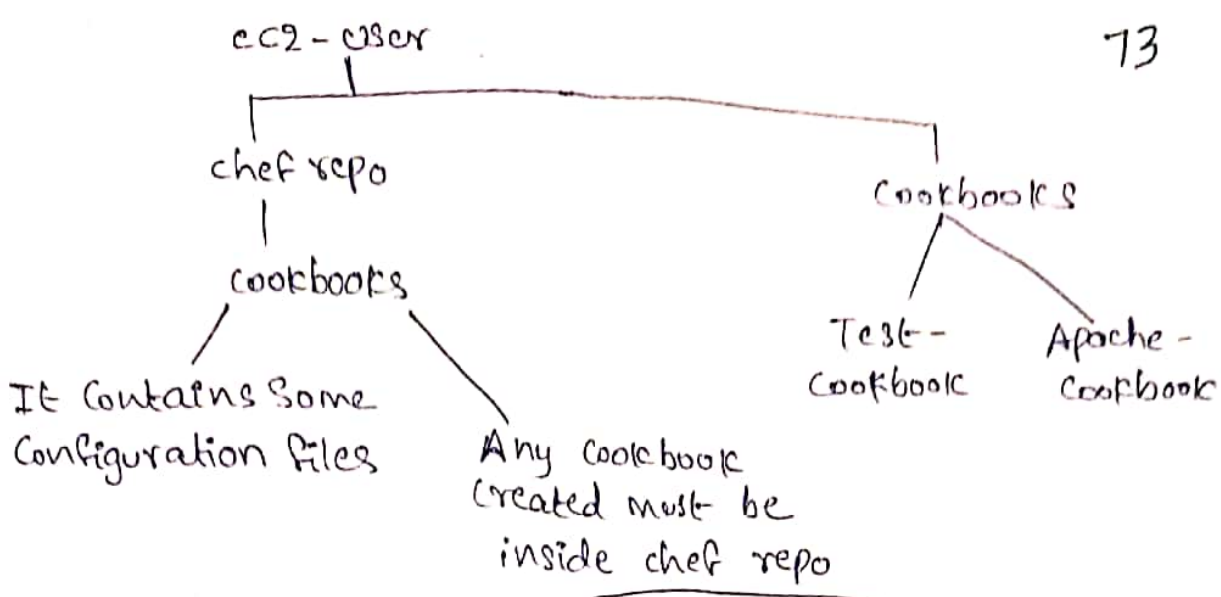
These are run inside the chef-repo

Node goto to Connected or not
knife node list
for check goto chef server and click on node

Upload:-
chef-repo ls
chef-node.pem cookbooks README.md roles
cd --
ec2-user]# ls
chef-repo chef-ws cookbooks nodes
↑ we created

cookbooks
↳ default
↳ apache.recipe
↳ sample.recipe

chef-repo folders
.chef
cookbooks
roles ↳ default



mv cookbooks/test-cookbook/chef-repo/cookbooks/

rm -rf cookbooks

To Upload:-

chef-repo)# knife cookbook upload apache-cookbook

To verify cookbook uploaded or not

knife cookbook list

Attaching the recipe to the node:-

apache-recipe)# knife node run_list set node1

"recipe[apache-cookbook::apache-recipe]"

Edit runlist - To See the node list

knife node show node1

Enter the node:-

sudo su

chef-client

1. Beready with cookbook
2. upload the Cookbook
3. Attach recipe to the node
4. Go and run the node

Running chef-client in regular interval

→ Goto node, open "crontab" file inside /etc directory & give

***** root chef-client

(in bootstrap script, can give below command to automate completely)

echo "***** root chef-client" >> /etc/crontab

→ Modify the recipe and upload that cookbook to chef server for testing

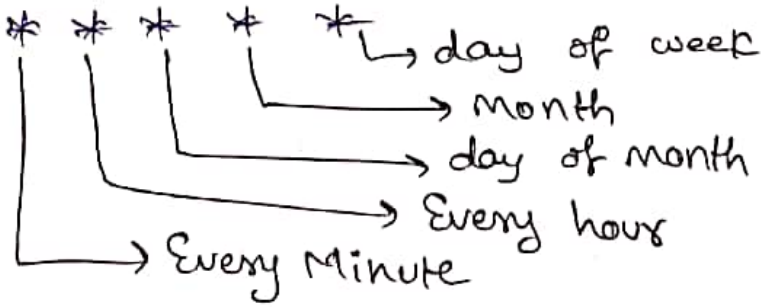
knife cookbook upload <cookbook>

(Adding cookbook's recipe to node's run-list is one time effort. Need not to add again and again)

21/1/2020

To Automate

ec2-user]# vi /etc/crontab ↵



* * * * * username Command to be executed

Space must | firing | root | chef-client

⇒ launch instance

```
#!/bin/bash
sudo su
yum update -y
echo "* * * * * root chef-client" >> /etc/crontab
```

Sample recipe_robo file cat /robofile (for open)

Roles:-

→ Custom run-list
→ We will create role & upload to chef server & assign them to nodes
→ If we have 1000 nodes, need to add cookbook to run-list of all 1000 nodes. Its very difficult. Instead, create role & attach that role to 1000 nodes once. Next time onwards add cookbook to that role. Automatically, all 1000 nodes will get that cookbook
→ So role is one time effort
→ Instead of adding cookbooks to each & every node run-list always, just create a role & attach that role to nodes
→ When we add cookbook to that role, it will be automatically apply to all nodes those assigned with that role

Roles

→ Terminate nodes & delete node and apache-cookbook from chef server
→ Be ready with 2 fresh nodes, boot strap them and automate chef client
→ Be inside chef-repo

- Vi roles/web.rb

```
name "web"
description "web server role"
run_list "recipe[apache-cookbook::apache-recipe]", recipe[test-cookbook::test-recipe]", "recipe[cookbookname]"
```

→Upload role to chef server (every time you modify the role) (run inside chef-repo)

- knife role from file roles/web.rb

→ Attach role to all nodes (one time effort) (run inside chef-repo)

- knife node run_list set <node_name1> "role[web]"
- knife node run_list set <node_name2> "role[web]"

→ To upload Cookbook (every time you modify) (run inside chef-repo)

- knife cookbook upload apache-cookbook

→ Paste Public IP of both nodes in web and verify (verify attributes & include recipe)

Test
- Modify recipe
- Upload cookbook

chef extra Commands

→ To upload all cookbooks to chef server in one go

knife cookbook upload -all

→ To deal with multiple resources at a time

```
['httpd','mariadb-server','unzip','git','vim'].each do |p|
  package p do
    action :install
  end
end
```

(or)

%w(httpd mariadb-server unzip git vim).
each do |p|

```
package p do
  action: install
  end
end
```

clean up the chef-server

→ To see list of cookbooks which are present in chef server

knife cookbook list

→ To delete cookbooks from chef server

knife cookbook delete <cookbook_name> -y

→ To see list of nodes which are present in chef server

knife node list

→ To delete nodes from chef server

knife node delete <node_name> -y

→ To see list of clients which are present in chef server

knife client list

→ To delete clients from chef server

knife client delete <client_name> -y

→ To ~~delete~~ see list of roles which are present in chef server

knife role list

→ To delete roles from chef server

knife role delete <role_name> -y

docker

Important differences

Container- A Container like a Virtual Machine

Docker - Docker is a tool to create those Virtual Machines

What is docker?

Docker is a tool that performs operating-system-level virtualization, also known as "containerization". It was first released in 2013 and developed by docker Inc.

→ Docker is a tool used to create virtual machines called "containers".

Docker:-

→ Took from shipping containers

→ Docker is a tool designed to make containers in which we can deploy any type of applications easily

→ Dockers use union file system (layered)

→ Docker performs OS-level virtualization

Docker Benefits:-

→ Containerization (os level virtualization) (No need guest os)

→ No pre-allocation of RAM

→ Can Replicate Same environment

→ Less Cost

→ Less weight

→ Fast to fire up

→ Can run on physical/virtual/cloud

→ Can re-use (same image)

→ Can create machines in less time

Docker Components:-

Docker image - Contains OS (very small) (almost negli-gible) + softwares

Docker Container - Container like a machine which is created from docker image

Docker file - Describes steps to create a docker image

Docker hub/registery - stores all docker images publicly

Docker daemon - Docker service

Ways to create docker images:-

→ Take image from Docker hub
→ Create image from existing docker Containers
→ Create image from docker file

24/1/2020

Docker Installation:-

Uname -r (os)
Yum update -y
Yum install docker -y
docker (to verify)
docker --version (to verify)
docker info (running or not)
Service docker start
docker info (running or not)
docker images (to see all images in our m-achine)
docker ps (to see only running Containers)
docker ps -a (to see all Containers)

How to Create a Container?

To Create Container

- docker run -i -t ubuntu //bin/bash
- i → Interactive mode
- t → Terminal
- // → S/by (Some OS's, / will be negated)
- hostname
- cat /etc/os-release
- exit

Hub.docker.com

Docker Basic Commands (Images)

→ To see all images Present in your local machine

- docker images

→ To search images in online docker registery

- docker search ubuntu

→ Download image from online docker registery to your machine

- docker pull jenkins
- docker pull chef/chefdk (Create container from this image)

Docker Basic Commands

→ Create Container by giving Container name

- docker run --name rkContainer -it ubuntu //bin/bash

→ To start Container

- docker start rkContainer

→ To go inside Container
.docker attach <Container>

→ To see all Containers
.docker ps -a

→ To see only running Containers
.docker ps

→ To stop Container
.docker stop <Container>

exit is used to come out of the Container

→ To delete Container
.docker rm <Container>

27/1/2020

Create Container from our own image:-

→ Create container
.docker run --name subcontainer -it ubuntu //bin/-bash
.touch /tmp/myfile
.docker diff subcontainer (run outside of container)

→ To create image from Container (Be inside base machine)
.docker commit saicontainer

→ To give a name to your image
.docker tag <Image ID> sivaimage
image name
.docker images

→ To give image name while creating image
.docker commit saicontainer <imagename>

→ Create Container from our image

•docker run --name haricontainer -it sivaimage /bin/bash

•ls /tmp

cd /tmp

tmp# ls

touch myfile

ls

myfile

exit

docker diff Container

diff

A Added

B deleted

C changed

Docker file

→ Docker file

•A text file with instructions to build image

•Automation of docker image creation

•FROM
•RUN
•CMD

} These instructions are in Capital letters

step-1:- Create a file named ~~Dockerfile~~ Dockerfile

step-2:- Add instructions in Docker file

step-3:- Build docker file to create image

step-4:- Run image to create Container

Docker file

→ vi Dockerfile

- FROM ubuntu
- RUN echo "Hi siva" > /tmp/testfile

→ To create image out of Dockerfile

- docker build -t test .
- docker PS -a
- docker images

→ create containers from the above image

- docker run -it --name testcontainer test //bin/bash
- cat /tmp/testfile

Docker file

→ vi Dockerfile

- FROM
- WORKDIR /tmp
- RUN echo "Hi sai" >/tmp/testfile
- ENV myname sai (echo $myname)
- COPY testfile1 /tmp
  (testfile1 must be in current directory in EC2)
- ADD test.tar.gz/tmp
  (test must be in current directory in EC2 in tar.gz format)

Volumes

→ Volume is a directory inside your container
→ First declare directory as a volume and then share volume
→ Even if we stop container, still we can access volume
→ Volume will be created in one container
→ You can declare as volume only while creating container
→ You can't create volume from existing container
→ You can share one volume across any number of containers
→ volume will not be included when you update an image
→ Map volumes in two ways
  - Share host - Container
  - Share Container - container

Volumes (declaration by using docker file) 28/1/2020

→ Docker volume
  - FROM ubuntu
  - VOLUME ["/data"]
→ docker build -t myimg
→ docker run -it --name mycont1 myimg //bin/bash
→ ls (you can see data folder)
→ touch /data/myfile
→ exit

Volumes (Container - Container)

→ To share volume while creating another container

- docker run -it --name mycont2 (new) --privileged
- true --volumes-from mycont1 (old) ubuntu //bin/bash

→ ls/data (can see myfile)

→ whatever u do in one volume, can see from other volume

- touch/data/test

→ docker start mycont1

→ docker attach mycont1

→ ls/data/ (can see test file)

→ exit

Volumes (declaration by using command)

→ docker run -it --name mycont3 -v /sai ubuntu //bin/bash

→ ls (can see sai folder, can mount like above same)

Volume (host - Container)

→ verify file in /home/ec2-user

→ docker run -it --name nit cont -v /home/ec2-user:/raj --privileged=true ubuntu //bin/bash

→ cd raj

→ ls (can see all files created in your host)

→ touch rajfile (in container/raj)

→ exit

→ ls (in host/home/ec2-user)

→ can see rajfile

→ To see full details of container (even if it is in stopped state)

→ docker inspect nitCont

Port expose

→ docker run -td (detached container) --name webserver (name of container) -p 80(host):80(container) ubuntu (d-daemon)

→ docker port webserver

→ docker exec -it webserver /bin/bash (to go inside container)

→ apt-get update

→ apt-get install apache2 -y

→ cd /var/www/html

→ echo "Hello" > index.html

→ service apache2 restart

→ In browser, public IP:80

(Can change host part. But not container port. make sure, you are taking default port for container)

→ Try with Jenkins image (8080:8080) (open 8080 Port in SG of EC2 machine)

(To unmap the port, stop the container or delete the container

29/1/2020

How to store docker images in Docker hub & how to share to others

1. Be ready with docker image (eg: krishnaimage)
2. Be ready with docker hub account and login to it
3. docker login (run this command in base machine & give docker hub credentials)
4. docker tag krishnaimage ~~krishna~~ chusrk13/newkrishnaimage
   (chusrk13 is docker hub id)
5. docker push chusrk13/newkrishnaimage
   (to push image to docker hub)

---

6. Be in some other machine
7. docker pull chusrk13/newkrishnaimage (to pull image to docker hub)

---

8. Can make docker image private also. So every time if some one wants to pull, they have to give your docker hub credentials. If it is public, no need to give credentials.

---

9. To logout, run "docker logout"

hub.docker.com → To create docker account

Important docker Commands

→ Stop all running Containers

```
docker stop $(docker ps -a -q)
```

→ Delete all stopped Containers

```
docker rm $(docker ps -a -q)
```

→ Delete all images:

```
docker rmi -f $(docker images -q)
```

Tools roles in DevOps

GitHub:- Github is an open source version control System with emphasis on data integrity and speed. GitHub - one of the world's largest Git repositories and a popular choice for developers collaborate and standardize code.

CHEF:- chef is a configuration management tool for dealing with machine setup on physical servers, virtual machines and in the cloud. Many companies use chef software to control and manage their infrastructure including facebook, Etsy, cheezburger and indiegogo.

Ansible:- Ansible is an open source IT configuration Management, Deployment & Orchestration tool. It aims to provide large productivity gains to a wide variety of automation challenges. This tool is very simple to use yet powerful enough to automate complex multi-tier IT application environments.

Puppet:- Puppet can define infrastructure as code, manage multiple servers, and enforce system configuration. Puppet is used by 42 percent of businesses that use DevOps methodologies, followed closely by chef with 37 percent. Puppet is an open source software configuration management and deployment tool.

Docker:- Docker is a tool designed to make it easier to create, deploy and run applications by using containers, Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Jenkins:- Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.
→ Jenkins achieves Continuous Integration with the helps of plugins. Pluging allows the integration of various Devops Stages

Maven:- Maven is a Powerful project management tool that is based on POM (Project object model). It is used for Projects build, dependency and documentation. It simplifies the build process like ANT.
In short terms we can tell maven is a tool that can be used for building and managing any java-based Project

Kubernetes:- Kubernetes is Google's solution to automate the deployment, scaling and management of applications in Containers: the tool adheres to the same principles Google uses to run Containerized applications and power their search engine along with a long list of Services.

Nagios:- Nagio is a free to use open source software tool for continuous monitoring. It helps you to monitor System, network and infrastructure. It is used for Continuous monitoring of systems, applications, service and business process in a Devops culture.
Nagios runs plugins stored on the same Server

Ansible

→ Ansible is an administration tool. whatever System admins (Linux/windows) used to manually, now we are automating all those tasks by using ansible (Any CM Tool).

Ex:- Water Pots

→ Can use this tool whether your Servers are in on-premises or in the cloud.

→ It turns your code into infrastructure ie; your Computing environment has some of the same attributes as your application

- → Your code is versionable
- → Your code is repeatable
- → Your code is testable

→ You only need to tell what the desired configuration should be, not how to achieve it

→ Ansible is same as chef both are Configuration management tools

Configuration Management:-

→ It's a method through which we automate admin tasks

→ Configuration management tool turns your code into infrastructure

→ So your code would be

- → testable
- → repeatable
- → versionable

IT infrastructure refers to the Composite of:-

- → Software
- → Network
- → People
- → Pixels

Configuration Management:-

Users
Groups
Permissions
files
folders
Packages
Services
Trouble Shooting

→ Manual — Admins — Windows / Linux

→ Automated — DevOps Engineer — Chef / Ansible / Puppet / Salt stack

IAC- Infrastructure As Code:-

1, Repeatable Code
2, Code is Versionable
3, Code is Testable

Pain Points:-

- → Managing user & group accounts
- → Dealing with packages
- → Taking backup
- → Deploying all kinds of applications
- → Configure Services

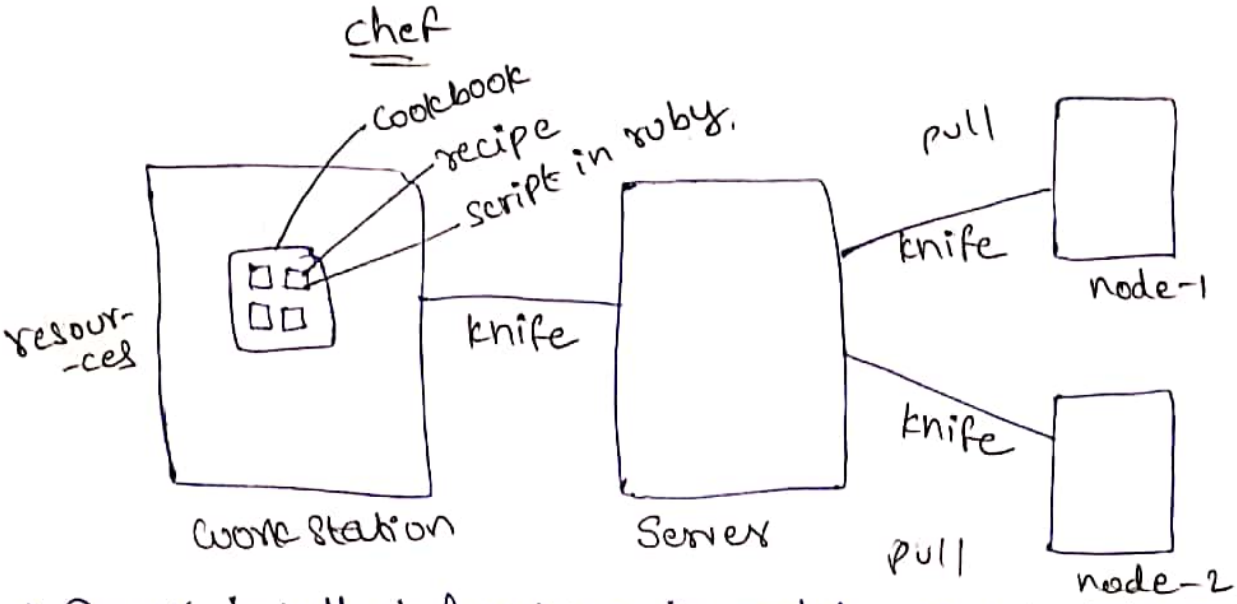
Why CM Tools?

- → Complete Automation
- → Increase uptime - Highly available
- → Improve performance → Prevent Errors
- → Ensure Compliance → Reduces Cost

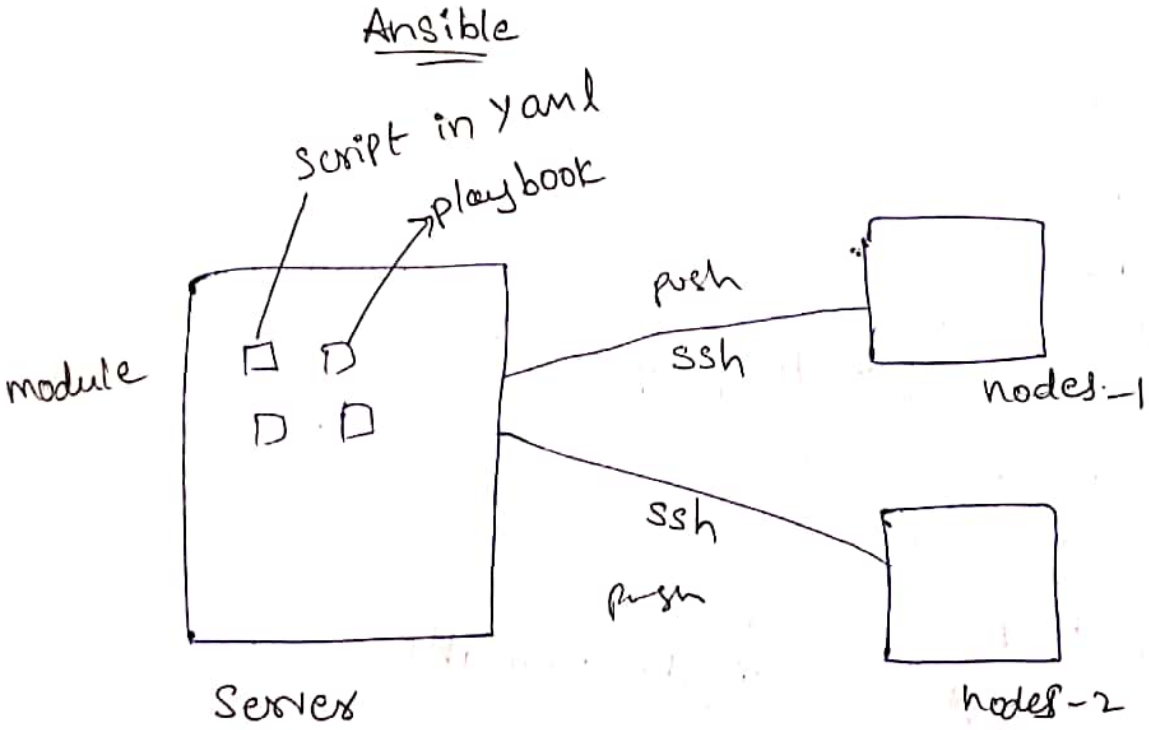
Configuration Management

Configuration details - Each and every detailed information

Comparison between chef and Ansible



-> Server install chef package in nodes using bootstrap



→ Chef is belongs to the chef company
→ Ansible is belongs to the ~~Ansible~~ Redhat company
→ Redhat is maintains by the IBM

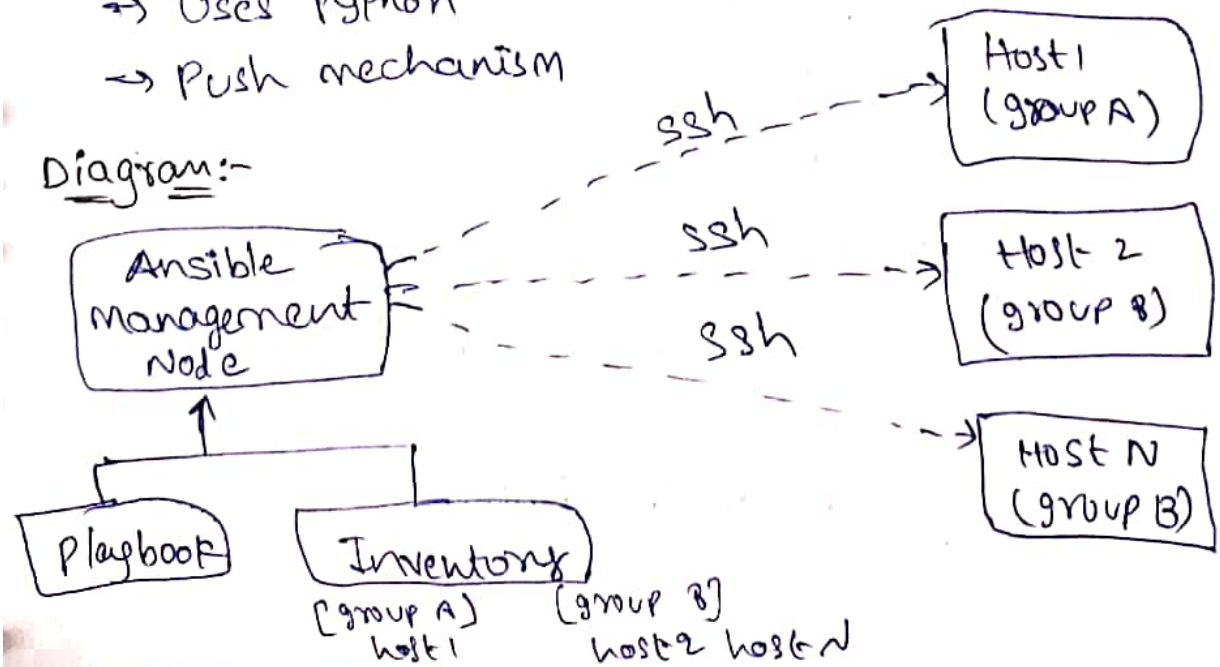
Why Ansible

→ Other tools in the market can be really complicated
- Huge overhead of infrastructure setup
- Complicated setup
- Pull mechanism
- Lot of learning required

Advantages of Ansible

→ Agentless
→ Relies on ssh
→ Uses Python
→ Push mechanism

Diagram:-



#history

1. yum update -y
2. wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
3. ls
4. yum install epel-release-latest-7.noarch.rpm -y
5. yum update -y
6. yum update -y
7. yum install git python python-devel python-pip openssl ansible -y
8. which ansible
9. ansible --version
10. ansible --version
11. history

5/2/2020

Install & Configure Ansible (Servers)

Launch Amazon Linux (need not be install ansible in nodes)

yum install wget -y

wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

yum install epel-release-latest-7.noarch.rpm -y

sudo yum update -y

sudo yum install git python python-devel python-pip openssl ansible -y

ansible --version

sudo vi /etc/ansible/ansible.cfg & enable the below lines

inventory = /etc/ansible/hosts

sudo_user = root

Ansible Inventory

→ Ansible recognizes systems listed in Ansible's inventory file (/etc/ansible/hosts)

→ Better to launch nodes as well in same AZ where server is there

the format for /etc/ansible/hosts looks like this:

(vi /etc/ansible/hosts) (add under Ex-1)

```
[groupname]
machinename (or) machine IP
```

Ex:

```
[demo]
<host name /private-ip of node>

[local]
localhost
```

Test Environment Setup(All machines)

→ adduser ansible (in all machines)

→ passwd ansible (in all machines)

→ visudo (add a line as below) (in all machines)

(adding ansible user into sudo users list)

→ ansible ALL=(ALL) NOPASSWD:ALL

To establish ssh connection among all nodes (do it in all machines)

→ vi /etc/ssh/sshd_config (all machines)

* ~~PermitRootLogin~~

→ PermitRootLogin yes (uncomment line, set to "yes")

→PasswordAuthentication yes (uncomment line, set to "yes")
→PermitRootLogin no(comment)
→PasswordAuthentication no(comment)
→service sshd restart

Test Environment Setup(Server)

→ check sudo works without asking password. (In Server)

→ su -ansible
→ whoami
→ sudo yum update
→ ssh <node private IP> (it prompts for password)

Run the following an ansible user

→ ssh-keygen (can see .ssh/ both keys in same directory)
→ copy the ssh keys to all the nodes (be in master) (be in .ssh folder) (asks for password for the last time)
→ ssh-copy-id ansible@<node-private-ip>
→ Test ssh to test machine, it should not ask password
→ ssh <node-private-ip>

Host Patterns

→ vi /etc/ansible/hosts
"all" pattern refers to all the machines in an inventory
→ ansible all --list-hosts
→ ansible <group_name> --list-hosts

ansible <group_name>[0] --list-hosts

You can refer to hosts within the group by adding a subscript to the group.

→ groupname[0] -- picks the first machine in the group

→ groupname[1] -- picks the second machine in the group

→ groupname[-1] -- picks the last machine in the group

→ groupname[0,1] -- picks the first 2 machine in the group

→ Groups separated by a colon can be used to use hosts from multiple groups

groupname1:groupname2

Ansible Ad-Hoc Commands

→ Use ad-hoc tasks really quick & don't want to save for later

→ these are quick one-liner without writing a playbook

→ To run an arbitary cmd use (-a)

→ ansible all --list-hosts

→ ansible demo -a "ls" (if all hostes, all)

→ ansible demo -a "ls -al"

→ ansible demo -a "touch anyfile" (re run and verify idempotance)

→ ansible demo -a "yum install httpd -y" (shows permission denied)

To run anything with sudo, use -b

→ ansible demo -b -a "yum install httpd -y"

→ ansible local -b -a "yum remove httpd -y"

Ansible Modules

→ To run an arbitary cmd use -a & use -m to run a module

→ ansible [group|host] -m <module> -a <cmd>

→ ansible demo -m ping

→ Copy a file test.txt (first create) from server to node

ansible demo -b -m copy -a "src = test.txt dest = /tmp/" (no need file name here)

Install/Remove a package

ansible demo -b -m yum -a "pkg=httpd state=present"

ansible demo -b -m yum -a "pkg = httpd state = latest"

ansible demo -b -m yum -a "pkg = httpd state = absent"

state = present will install it

state = latest will update

state = absent will remove it

Start/Stop a Service

ansible demo -b -m service -a "name = httpd state = started"

ansible demo -b -m service -a "name = httpd state = restarted"

ansible demo -b -m yum service -a "name = httpd state = stopped"

Create/Delete a User account

ansible demo -b -m user -a 'name=krishna"

ansible demo -b -m user -a "name=krishna
state=absent"

Gathering facts (Idempotence)

ansible demo -m setup

ansible demo -m setup -a 'filter=*ipv4*'

6/2/2020

Playbooks

→ playbook is like a file where u write code

→ ansible-playbook <playbook>.yml

Playbooks are expressed in YAML format

Ansible vs Chef

Playbook - Recipe

Module - Resource

Host - Node

Setup - Ohai

ssh - knife

Push - Pull

YAML (Yet Ain't Markup Language) Basics

→ For Ansible, nearly every YAML file starts with a list

→ Each item in the list is a list of key/value pairs, commonly called a "dictionary"

→ All YAML files have to begin with "---" and end with "..."

→ All members of a list lines must begin with same indentation level starting with "- "

```
---# A list of tasty fruits
Fruits:
   -Apple
   -Orange
   -Strawberry
   -Mango
...
```

→ A dictionary is represented in a simple key: value form (the colon must be followed by a space)

```
---# An employee record
Employee:
    name: KRISHNA
    Job: DevOps Engineer
        skill: Elite
...
```

Playbooks :-

→ Each playbook is composed of one or more 'modules' in a list

→ playbooks are divided into many sections like..

Target Section - Defines the hosts against which playbooks tasks has to be executed.

Variable Section - Defines variables

Tasks Section - List of all modules that we need to run, in an order

Our First Playbook

→ All sections begin with "-" & its attributes & parameters beneath it.

→ Indentation is imp, use only spaces & not tabs

→ Create a folder (playbooks) & go inside that (vi test.yml)

test.yml

```
---# My first YAML playbook
-hosts: demo
 tasks:
   -name: Install httpd on server
    action: yum pkg=httpd state=installed
```

→ Run ansible-playbook to call the playbook

→ ansible-playbook test.yml

Target Section

Create a file (vi first.yml)

Ex:-

```
--- #My First YAML playbook
- hosts: demo
  user: ansible                    #.yes or no
  become: yes
  Connection: ssh                  # ssh or paramico
  gather_facts: yes                #yes or no
```

→ Run ansible-playbook to call the playbook

→ ansible-playbook first.yml

Task Section:-

Ex:-

```
--- # My first YAML playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
   - name: Install HTTPD on centos 7
     action: yum name=httpd state=installed
   - name: Install MYSQL on centos 7
     action: yum name=mysql state=installed
```

→ Run ansible-playbook to call the playbook

ansible-playbook first.yml

(Remove httpd package).

variables: Inclusion Types

→ Create a section called vars within a playbook
→ Put vars above tasks so that we define it first & use it later

```
--- # My first YAML playbook
- hosts: demo
  user: ansible
  become: yes
  Connection: ssh
  vars:
    pkgname: httpd
  tasks:
   - name: Install HTTPD Server on centos 7
```

action: yum name = '{{ pkgname }}' state = installed
(Remove httpd package)

Handler Section

→ Consists the ability to notify when something happens
→ Also call another set of tasks
(Remove apache from node)

Ex:

```
---# My first YAML playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: Install HTTPD server on centos 7
      action: yum name=httpd state=installed
      notify: restart HTTPD #this is called only
if the action is ran & successful #
  handlers:
    - name: restart HTTPD #this has to match the
notify name. Otherwise throws error #
      action: service name=httpd state=restarted
```

(Install httpd in both nodes & don't start service. Then try running above script. It won't start service)

→ Run ansible-playbook to call the playbook
→ ansible-playbook first.yml (Remove httpd from node)

Dry Run

→ Check whether the playbook is formatted correctly
→ Test how the playbook is going to behave without running the tasks

```
ansible-playbook webserver.yml --check
```

Loops

→ Often you'll want to do many things in one task, such as create a lot of users, install a lot of packages, or repeat a polling step until a certain result is reached

Ex:-

```
---  # Loop Playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: add a list of users
      user: name='{{ item }}' state=present
      with_items:
        - Siva
        - Rama
        - krishna
```

Conditionals :-

Few tasks might be needed to execute only a Specific scenario

When statement

sometimes you will want to skip a particular step on a particular host

Ex:-1

```
---  # When playbook example
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
   - name: Install apache for debain
     command: apt-get -y install apache2
     when: ansible_os_family == "Debian"
   - name: Install apache for redhat
     command: yum -y install httpd
     when: ansible_os_family == "RedHat"
```

```
cat /etc/os-rele*
```

Vault

→ Ansible allows keeping sensitive data such as passwords or keys in encrypted files, rather than as plaintext in your playbooks

→ Creating a new Encrypted playbook (Put ---) (verify file permissions) (Open playbook now)

```
ansible-vault create playbook.yml
```

→ Edit the Encrypted playbook

```
ansible-vault edit playbook.yml
```

→ change the password

```
ansible-vault rekey playbook.yml
```

→ Decrypt the playbook & verify by opening playbook

```
ansible-vault decrypt playbook.yml
```

→ Encrypt an existing playbook (verify by opening playbook)

ansible-vault encrypt playbook.yml

Roles (Theory)

Adding more & more functionality to the playbooks will make it difficult to maintain in a single file. We can organize playbooks into a directory structure called roles.

Creating Role Framework (eg: playbooks/roles/webserver/~~master.yml~~ → ↓ /master.yml

/master.yml

playbook/roles/webserver/tasks/main.yml
/vars/main.yml
/handlers/main.yml

Roles (Lab)

Master.yml (master.yml & roles folder must be in same directory level)

→ vi roles/webserver/tasks/main.yml

```
- name: Install Apache on CentOS
  yum: pkg=httpd state=latest
```

playbook/roles/webserver/tasks/main.yml
/master.yml
/ansible-playbook master.yml (command run place)

This is how we mention Role in play-books

Ex:-

Create master.yml (inside playbook folder)

```
--- # master playbook for web servers
- hosts: all
  user: ansible
  become: yes
  connection: ssh
  roles:
    - webserver
```

ansible-playbook master.yml (run being present in playbooks folder)

Practical:-1

```
--- # My first YAML playbook
- hosts: demo
  user: ansible
  become: yes                # yes or no
  connection: ssh            # ssh or paramico
  gather_facts: yes          # yes or no
```

```
--- # My first YAML playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: Install HTTPD on centos 7
      action: yum name=httpd state=installed
```

3.

```
--- # My first YAML playbook
- hosts: demo
  user: ansible
  become: yes
  Connection: ssh
  vars:
    pkgname: httpd
  tasks:
  - name: Install HTTPD server on centos 7
    action: yum name='{{pkgname}}' state=
                    installed
```

4.

```
--- #My First YAML playbook
- hosts: demo
  user: ansible
  become: yes
  Connection: ssh
  tasks:
    -name: Install HTTPD server on centos 7
    action: yum name=httpd state=installed
    notify: restart HTTPD # is called only if
    the action is ran & successful #
  handlers:
    - name: restart HTTPD # to match the notify
    name. Otherwise throws error #
    action: service name=httpd state=restarted
```

5.

```
--- # Loop playbook
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
```

```
tasks:
- name: add a list of users
  user: name = '{{ item }}' state = present
  with_items:
     - Siva
     - Rama
     - krishna
```

6.
```
--- # when playbook example
- hosts: demo
  user: ansible
  become: yes
  connection: ssh
  tasks:
  - name: Install apache for debain
    command: apt-get -y install apache2
    when: ansible_os_family == "Debian"
  - name: Install apache for redhat
    command: yum -y install httpd
    when: ansible_os_family == "RedHat"
```

7.
```
- name: Install Apache on CentOS
  yum: pkg=httpd state=latest
```

8.
```
--- # master playbook for web servers
- hosts: all
  user: ansible
  become: yes
  connection: ssh
  roles:
    -webserver
```

6/2/2020

Practical

→ The playbook creation is simple

vi target.yml

Here target.yml is the playbook to execute the playbook example

→ ansible-playbook target.yml

task = target + task

target + task + variables = variables

vi vars.yml

variable

key: value

→ Handler Section

vi handlers.yml

ansible-playbook handlers.yml

After package started then service will started

→ Dry Run

--check

It adds after the file when execute stage

Source Code Management

Source code Management (SCM) and version Control Systems ensure all members of a team stay on top of changes to source code and related files. these tools are also crucial in coordinating parallel work on different features and the integration of the features for software releases

Continuous Integration:-

Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, which automated builds and tests are run. Continuous integration most often refers to the builds and tests are run. Continuous integration most often refers to the build or integration stage of the software releases process and entails both an automation component (e.g. a CI or build service) and a cultural component. the key goals of continuous integration are to find and address bugs quicker, improve software quality and reduce the time it takes to validate and release new software updates

Configuration Management:- DevOps has several components that must work in unison for a team to meet its objectives. A key element, which usually serves as the center of the DevOps "machinery", is configuration management

DevOps spans across software development and operations phases. therefore, it is only fitting that configuration management spans across both the areas.

Continuous Deployment:- Continuous Deployment is a software development practice in which every code changes goes through the entire pipeline and is put into production, automatically, resulting in many production deployments every day.

Deployment is updating your codes on the servers. Servers can range from one to thousands. You need proper tools and strategies to deploy the code to server and refresh the site. To avoid breaking live website, code is developed to various stages.

Testing - Staging - PreProd - Prod

Continuous Monitoring:- Continuous monitoring refers to the process and technology required to incorporate monitoring across each phase of your DevOps and IT operations lifecycles. It helps to continuously ensure the health, performance, and reliability of your application and infrastructure as it moves from development to production

Continues Integration & Continues Delivery/Deploy

→ whenever developers write code, we integrate all that code of all developers at that point of time and we build, test and deliver/deploy to the client. this process is called CI & CD.

→ Jenkins helps in achieving this

→ So instead of doing night builds, build as and when commit occurs by integrating all code in scm tool, test and checking the quality of that code is what continues Integration.

→ So in a day, there will be so many integrations, builds, tests & deliveries/deployments.

→ So bugs will be reported fast and get rectified fast. So development happens fast.

Key Terminology:-

Integrate:- Combine all code written by developers till some point of time.

Build:- Compile the code and make a small executable package

Test:- Test in all environments whether application is working properly or not

Archived:- stored in an artifactory so that in future we may use/deliver again

Deliver:- Handling the product to client

Deploy:- Installing product in client's machines

Jenkins work flow

→ We attach Git, Maven, Selenium & Artifactory plug-in's to jenkins
→ Once developers put code in Git, Jenkins pull that code & send to Maven for build
→ Once build is done, Jenkins pull that built code and send to selenium for testing
→ Once testing is done, then jenkins pull that built code and send to selenium for testing
→ We can also deploy with Jenkins.

Ways of Continuous Integration

→ Can do manually
→ Can write scripts
→ Can use tool like Jenkins

Benefits of CI

→ Detect bugs as soon as possible
→ If you want you can stop the SDLC process. at any stage
Ex:- You can stop at test stage only or you can continue till deployment
→ Maintain history (logs) for reference

Why Only Jenkins

→ It has so many plug-ins
→ You can write your own plug-in
→ You can use Community plug-ins
→ Jenkins is not just a tool. It is a framework ie., You can do whatever you want. All you need in plug-ins.

→ We can attach slaves (nodes) to jenkins master. It instructs other (slaves) to do job. If slaves are not available, Jenkins itself does the job.

→ Jenkins also acts as crone server replacement, ie, can do repeated tests automatically

→ Running some scripts regularly

Ex: Automatic daily alarm

→ Can create labels (Can restrict where the Project has to run

Jenkins

→ Jenkins Architecture

→ Client - Server model

→ when you install jenkins it acts as master

→ Jenkins invoke clients to perform jobs.

Supported as

→ Can install in any os

→ you will be accessing Jenkins through web only.

→ choose long Term Support release (to get support)

→ Install Java

→ Install web server

Jenkins

→ Enterprise edition - Hudson

→ Open source - Jenkins

Three types of Configurations

→ Global Configuration

→ Node Configuration

→ Job Configuration

Git for Windows

→ git downloads
→ Download
→ Additional icons
→ Use git from Command line and also from 3rd party Software
→ Use OpenSSH
→ Use OpenSSL Library
→ checkout as-is, commit Unix-style line endings
→ Use MinTTY
→ Uncheck view release notes
→ git version
→ git config --global --list
→ git config --global user.name "sai"
→ git config --global user.email "saiecs@gmail.com"
→ git config --global --list
→ cat ~/.gitConfig

Java for Windows

→ Java development kit download
→ Oracle Technology Network (website)
→ JDK (Download)
→ Accept License Agreement
→ Download for windows
→ Run the java downloaded application
→ restart the PC
→ java -version

Configure JAVA

→ Go to c drive
→ Program files
→ Java
→ Jdk (go inside JDK folder)
→ Copy the path
→ This PC
→ Properties
→ Advanced system settings
→ Advanced
→ Environment variables (in both blocks)
→ New
→ JAVA_HOME & paste that you have copied (path to Java JDK)
→ Go inside bin folder of jdk & copy the path
→ System variables - path - edit - new - paste that you have copied - ok
ok, ok, ok

→ Restart PC
→ To verify:-
Go to command prompt
set JAVA_HOME (in windows cmd)
echo %JAVA_HOME% (in windows cmd)

Maven Installation

→ maven.apache.org
→ Download
→ Maven binary zip link
→ Extract (C:\DevTools) the downloaded file
→ Go inside folder
→ Copy the path
→ This PC - Properties
→ Advanced System Settings
→ Advanced
→ Environment Variables
→ System Variables
→ New
→ M2_HOME & paste that you have copied (path to Maven) & OK
→ Go inside bin folder of maven & copy the path
→ System Variables - path - edit - new - Paste you have copied - OK
→ Restart PC (Mandatory)
→ Go to Command Prompt
→ mvn -version (In windows cmd prompt)
→ echo %M2_HOME% (in windows cmd) (to verify)

Jenkins Installation

→ jenkins.io
→ Download
→ LTS Release - windows
→ extract here
→ run the .exe
→ Restart PC
→ localhost:8080
→ Copy the path to password & paste on notepad++ (open) & paste
→ Install suggested plugins
→ Username - admin
→ password -
→ Confirm password
→ Full name - admin
→ Email - sai4cs@gmail.com
→ http://localhost:8080

Jenkins

→ Get familiar with Jenkins dashboard
→ Jenkins services on windows
  localhost:8080/restart
  localhost:8080/stop
  localhost:8080/start
→ Create sample job & familiar with all options & build
→ Job/Project/Item: Projects are some times referred to as jobs or items. Jenkins appears to use these terms interchangeably

→ Workspace:- the workspace is the location on your computer where Jenkins place all files related to the Jenkins project. By default each project or job is assigned a workspace location and it contains Jenkins-specific project metadata, temporary files like logs and any build artifacts, including transient build files.

Jenkins (Free Style Project)

→ Windows Batch Project
Name item - Script project
Free style project - ok
build
execute windows batch command
echo "Hello Dear Students!!"
save
Build now
Console output

→ Simulate a fail build

→ Delete a project

→ Fix broken build

→ Copy a project

Jenkins

→ Manage Jenkins
Update plugins
Plugin research
Add new plugins & remove (Green Ball)
Maven integration plugin

Global Tool Configeration

JDK (Give installed path)

Git (No need)

Maven (Give installed path)

Jenkins

Plugin:-

→ With jenkins, nearly everything is a plugin and that nearly all functionality is provided by plugins. You can think of jenkins as little more than an executor of plugins

→ plugins are small libraries that add new abilities to jenkins and can provide integration points to other tools.

→ Since nearly everything jenkins does is because of a plugin, jenkins ships with a small set of default plugins - some of which can be upgraded independently of jenkins.

Maven Project (by Maven)

→ Go to https://github.com/SaiDevOpsFaculty/time-tracker

→ click on time tracker repo

→ fork

→ Sign in to git hub

→ click on time-tracker repo

→ clone

→ Go to c drive

→ git clone <url of time tracker project>

→ cd time-tracker
→ mvn clean package

Maven Project (By Jenkins)

→ Copy the main url (top) of time-tracker project
→ new item - Time Tracker - maven project - OK
→ Git Hub project (Paste link)
→ Source code management - git
→ Paste link of git http (http/ssh) url of time-tracker
→ build
→ clean package (pom.xml is already there)
→ save
→ Build now
→ Verify workspace contents with github side
→ See console output (Git clone)
→ Click on github icon on Jenkins
→ verify modules

Troubleshooting maven projects

→ Go to git hub time tracker
→ Time-tracker - core - src ---- core
→ Tracker.java - edit
→ 3, 4 //
→ Causing problems by commenting out code (Commit msg)
→ Commit changes

→ Build now
→ Can see failed build (Refresh page)
→ See console output (Cannot find symbol)
→ Remove // from both lines from github (Time-Tracker)
→ Undoing committing changes to fix build
→ commit changes
→ Build now
→ Can see Build is successful

Scheduled projects

→ click on any project - configure - build triggers - build periodically
* * * * *
→ save
→ Can see automatic builds every 1 min
→ You can manually trigger build as well

Source code Polling

→ New Job (Polling example)
→ Copy from Time-Tracker project - ok
→ Build Triggers
→ Poll SCM (* * * * *)
→ First time it will trigger build immediately (main page)
→ Go to time tracker repo in git hub in your account
→ edit ReadMe file
→ Give commit message & click on Commit changes

→ Can see Build triggered (Main page)
→ See changes tab to see last change

Related/Linked Projects

Upstream Projects:- Trigger other Projects

Downstream Projects:- Triggered by other Projects

UpStream Project Practice

→ Enable Auto Refresh & create 2 jobs (Job A & Job B)
→ Go to Job A & Job B
→ Source code management - git
→ Paste link of git http (http/ssh) url of time-tracker.
→ Post build actions (Job A) - Build other projects - Job B - Save
→ Build now (Job A)
→ Verify dashboard & Console Output (Job A triggers Job B) (Push)

Down Stream Project Practice

→ Go to Job B
→ Build Triggers
→ Build after other projects are build - Job A - Save
→ Build now (Job A)
→ Verify dashboard & console output (Job B triggered by Job A) (pull)

→ Up stream (Job A takes initiative) (JobA Push JobB)

→ Down stream (Job B takes initiative) (JobB pull JobA)

Views

→ List of Related Projects
→ Default views - All
→ Click on + - (free style) - List Views - OK
→ Job filter - 1, 2, 4, 6 - OK
→ Click on + - (Maven) - List Views - OK
→ Job filter - 3, 5 - OK

Deleting Views

→ Delete view - yes
→ Projects get dispersed
→ can add description

User Management

Create New Users:-

→ Manage jenkins - Manage users - Create user(2) (user1, user2)
→ Login as user (will have all privileges) - Logout

Manage users: (Login as Admin)

→ Install "Role-based authorization Strategy" plug-in
→ Manage jenkins - Configure global security - enable security - Role based strategy - Save

→ login as users (No privileges) - logout
→ login as Admin - Manage Jenkins - Manage & Assign roles - Manage roles
→ Role to add (Global) - employee - add - Read & View access - apply
→ Role to add (Project) - developer, Dev.* (pattern) - add - give all permissions (check) - apply
→ Manage jenkins - Manage & Assign roles - Assign Roles - add both users to Global roles - check users employee boxes - apply
→ add both users to Item roles - user1 (developer) & user2 (tester) - apply - Save
→ Create 2 new projects (DevProject, TestProject) - logout
→ login as users & verify

Master Slave setup - Windows (using command)

→ ManageJenkins - ManageNodes - NewNode - slave1 - Permanent Agent - OK
→ Remote Root Directory (C:\Jenkins)
→ LaunchMethod - Launch agent via execution of command on master - download agent.jar & paste in C drive - Launch Command (Java -jar C:\agent.jar) - SAVE
after few seconds, slave will be up

→ Build any job - can see job running on slave agent

→ In Node configuration(Slave1) - labels - give any label name (Mynode)

→ In job Configuration - Restrict where this Project can be run - give label name(Mynode)

→ Now, build. Can see job running on Slave1.

12/2/2020

→ In Jenkins Build means running the job

→ the Job is fail then build is fail

→ Job is Success then build is Success

* Groovy script for plugin
* windows - batch
* Linux - bash/shell commands

Grey - Not build

Red - build failed

Blue - build Success

Blinking - building in Progress

Maven

129 17/2/2020

What is Build?

Build: Compile + Assembly + Create deliverable

Compile: Convert source code to machine readable format

Assembly (linking): Grouping all class files

Deliverable: .war .jar

Advantages of Build tool:-

→ Automated tasks (Mention All in pom.xml)
pom-

→ Multiple tasks at a time

→ Quality Product

→ Minimize bad builds

→ Keep history

→ Save time - save money

→ Documentation

→ Gives set of standards

→ Give define Project life cycle (Goals)

→ Manage all dependencies

→ Uniformity in all Projects

→ Re-usability

Why Separate build team?

→ To match customer's environment

→ Build team worry about whole product

Build tools

→ C, C++ = make file
→ .Net = Visual Studio
→ Java = Ant, Maven
(Ant: older; Maven: Apache; Advance)

Architecture of Maven:-

→ Main Configuration file is pom.xml
→ One Project - one work space - one pom.xml

Requirements for build:-

→ Source code (Present in work directory)
→ Compiler (Remote Repo - local Repo - workspace)
→ Dependencies (Remote Repo - local Repo - workspace)

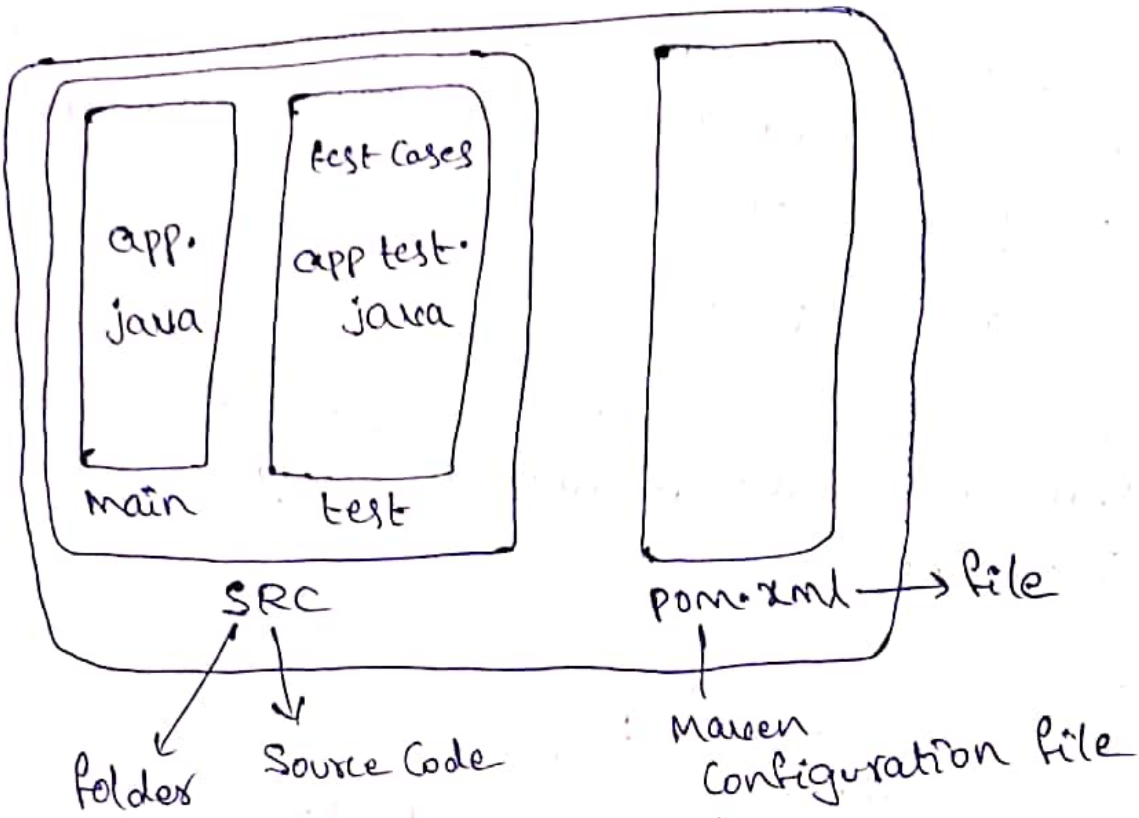
Maven Build life cycle:-

→ Goals:-
1. Generate resources
2. Compile code
3. Unit test
4. package (Build)
5. Install (into local repository & artifactory)
6. Deploy (to Servers)
7. clean (delete all run time files)

Ex: mvn install
mvn clean package

1-6 → Default & Sequence order
7 → Not default & It won't follow Sequence

Maven Directory Structure:-



Maven Repositories:-

→ Local (.M2)

→ Remote (https://repo1.maven.org/maven2/)

Maven hub

Pom.xml Contains

→ Metadata - Complete details

→ Dependencies

→ kind of Project

→ kind of output (.jar, .war)

→ Description

Important Points

-> Maven is all about plug-ins

-> Snapshot:- Indicates development copy of your project. Not the one which you are going to release

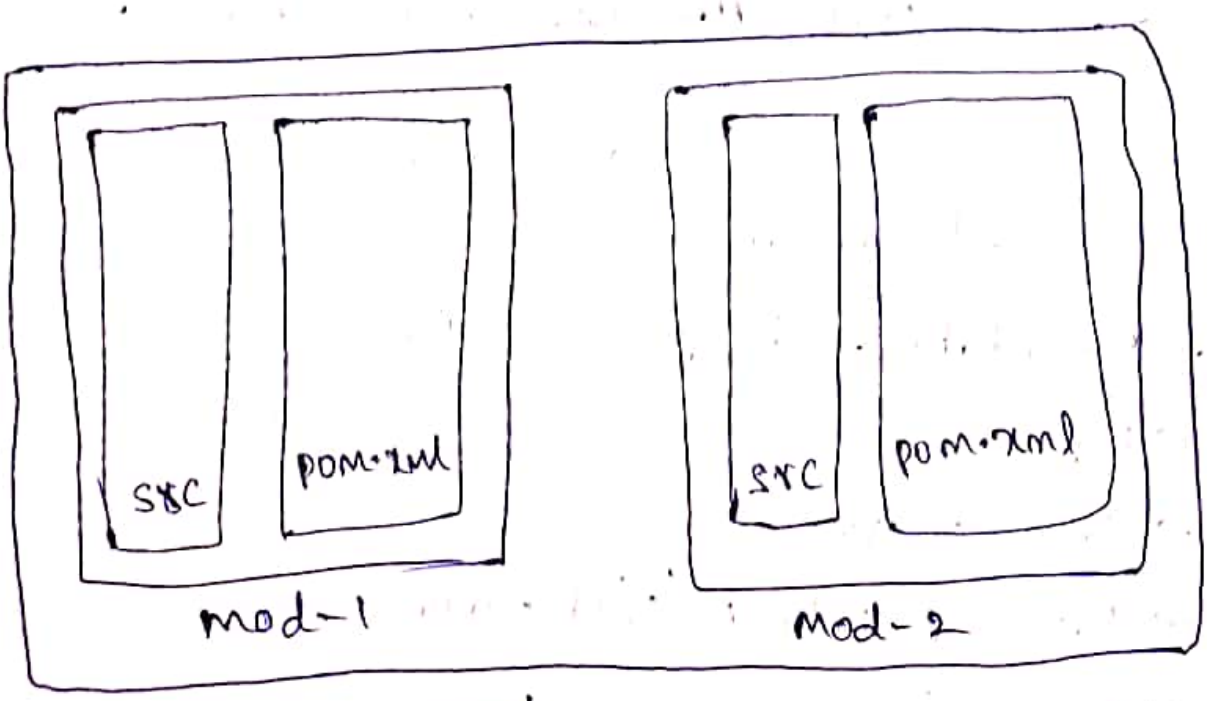
Ex: 1.0 - SNAPSHOT

-> If you see version no in place of snapshot, that it means product is ready to give Customer.

Version => 8.0.0

- 8 → major
- 0 → change in applications
- 0 → minor

Multi-Module Project:-



demo

→ Simply dividing Project into modules
→ Each module must have its own SRC folder & pom.xml so that build will happen separately
→ To build all modules with one command, there should be a parent pom.xml file. This calls all child pom.xml files automatically.
→ In parent pom.xml file, need to mention the child pom.xml order

Why Nagios

→ Why monitoring tool?
High Availability
Reduce downtime

→ We can monitor by using?
Scripts
Tool

Why Nagios??

→ Oldest & Latest
→ Stable
→ So many plug-ins
→ Own Database
→ Monitoring & Alerting

Important to note

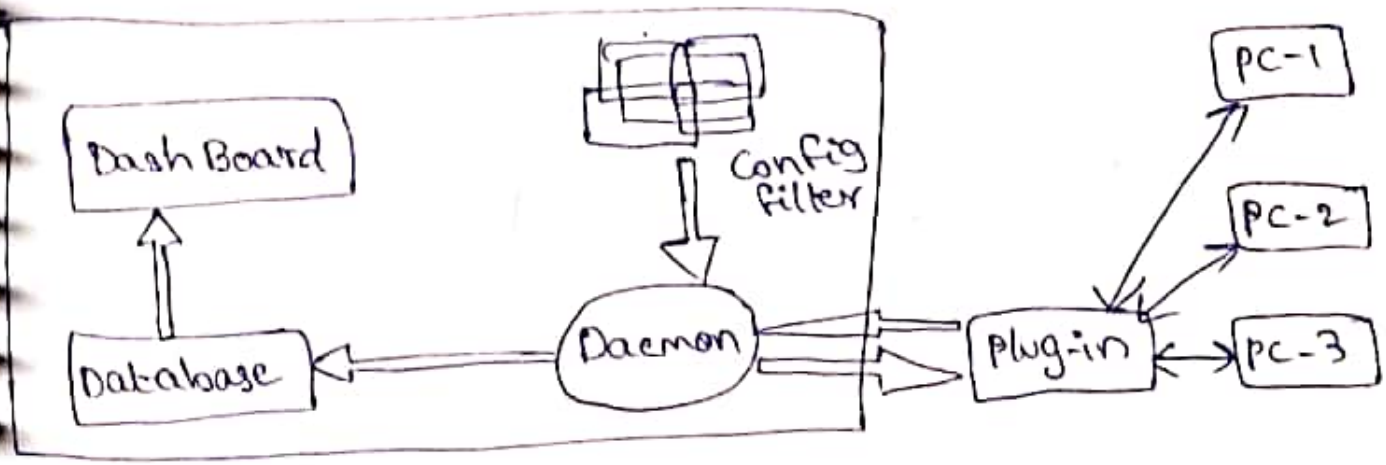
Plug-ins

→ Can use open source (Community plug-ins)
→ Can write your own

Pre-requisites

→ httpd (Browser)
→ php (dashboard)
→ GCC & GD (Compilers) (To Convert raw code into binaries)
→ makefile (to build)
→ perl (script)

Architecture of Nagios



How does it works?

→ Mention all details in config files
→ Daemon read those details what data to be collected
→ Daemon use NRPE plug-in to collect data from nodes and stores in its own database
→ Finally displays in dashboard

Important points

→ Main configuration file
/usr/local/nagios/etc/nagios.cfg
→ All monitoring things called as "Services"
eg:- 5 servers - 3 checks each
→ You have to monitor 5 * 3 = 15 services
→ While define plug-ins, can set upper & lower limit of monitoring range

→ Things to mention for each server in config files

- Username & Password
- Service (Plug-in)
- IP address
- Upper & Lower threshold

Dashboard overview

In dashboard, you can see

Hosts
- down
- unreachable
- recovery
- none

Services
- warning
- unknown
- critical
- recovery

How does it works??

→ To monitor remote machines

Install NRPE (Nagios Remote Plug-in Executor) on server

Use check-by-ssh plugin (this plugin has to be in remote machines)

libexec folder (where plugins will be stored)

NRPE plugin in nagios server will go to client by ssh & invoke check-by-ssh plugin

Grouping
- servers (Host group)
- Services (Service group)

Nagios Installation

→ yum install -y httpd httpd-tools php gcc
→ glibc glibc-common gd gd-devel make net-snmp
→ useradd nagios
→ groupadd nagcmd
→ usermod -G nagcmd nagios
→ usermod -G nagcmd apache
→ mkdir /root/nagios
→ cd /root/nagios
→ wget https://assets.nagios.com/downloads/nagios-core/releases/nagios-4.3.4.tar.gz
→ wget https://nagios-plugins.org/download/nagios-plugins-2.2.1.tar.gz
→ tar -xvf nagios-4.3.4.tar.gz
→ tar -xvf nagios-plugins-2.2.1.tar.gz
→ ls -l
→ cd nagios-4.3.4/
./configure --with-command-group=nagcmd
→ make all
→ make install
→ make install-init
→ make install-Commandmode
→ make install-config
→ make install-webconf

→htpasswd -s -c
→ /usr/local/nagios/etc/htpasswd.users
→nagiosadmin
→service httpd start
→systemctl start httpd.service
→cd /root/nagios
→cd nagios-plugins-2.2.1/
./configure --with-nagios-user=nagios --with-nagios-group=nagios
→make
→make install
→/usr/local/nagios/bin/nagios -v
→/usr/local/nagios/etc/nagios.cfg
→chkconfig --add nagios
→chkconfig --level 35 nagios on
→chkconfig --add httpd
→systemctl enable nagios
→systemctl enable httpd
→service nagios start
→Systemctl start nagios.service
→Public IP/nagios (in browser)
→nagiosadmin (user name)
→Password

/usr/local/nagios/bin - binary files
/usr/local/nagios/sbin - CGI files (to get web page)

/usr/local/nagios/libexec - plugins

/usr/local/nagios/share - PHP files

/usr/local/nagios/etc - configuration files

/usr/local/nagios/var - logs

/usr/local/nagios/var/status.dat (file) - database

---

main configuration file

→ /usr/local/nagios/etc/nagios.cfg

→ /usr/local/nagios/etc/objects/localhost.cfg
(hosts information)

→ /usr/local/nagios/etc/objects/contacts.cfg
(whom to be informed (emails))

→ /usr/local/nagios/etc/objects/timeperiods.cfg
(at what time to monitor)

→ /usr/local/nagios/etc/objects/commands.cfg
(plugins to use)

→ /usr/local/nagios/etc/objects/templates.cfg
(templates)

Install Jenkins on AWS EC2

→ Prerequisites
  . EC2 with Port 8080 open

→ yum -y install java-1.8.0-openjdk
→ java -version
→ JAVA_HOME = /usr/lib/jvm/java-1.8.0
  -openjdk-1.8.0.191.b12-1.el7_6.x86_64
→ export JAVA_HOME
→ PATH = $PATH : $JAVA_HOME
→ source ~/.bash_profile
→ yum -y install wget
→ wget -O /etc/yum.repos.d/jenkins.repo
→ https://pkg.jenkins.io/redhat-stable/jenki-
  -ns.repo
→ rpm --import https://pkg.jenkins.io/redhat-
  stable/jenkins.io.key

→ yum -y install jenkins

→ Start Jenkins
→ Systemctl start jenkins
→ systemctl enable jenkins

(http://YOUR-SERVER-PUBLIC-IP:8080)
  in browser

Kubernetes

→ K8S is a Container orchestration technology that creates, deploy and manages clusters (bunch of docker containers)

→ It schedules, runs and manages isolated containers which are running on virtual/physical/cloud machines

→ Convert isolated containers running on different H/W into cluster.

All 3 clouds support kubernetes

→ K8S originated at Google

Features of kubernetes

→ Orchestration (clustering of any number of containers running on different H/W)

→ Auto-Scaling (more clients? More demand)

→ Auto-Healing (new containers in place of crashed containers)

→ Load-Balancing (Distribute client requests)

→ platform Independent (Cloud/virtualization/phyllical)

→ fault tolerance (Node/pod failures)

→ Roll back (Going back to previous versions)

Kubernetes

→ Working with kubernetes

→ we create manifest (.yml)

→ Apply this to cluster (to master) to bring into desired state

→ Pod runs on node, which is controlled by master.

Role of master node

→ Kubernetes cluster contains containers running on Bare metal/VM Instances/cloud instances/all mix.

→ kubernetes designates one or more of these as master and all others as workers

→ The master is now going to run set of K8S processes. These processes will ensure smooth functioning of cluster. these processes are called "Control plane".

→ Can be Multi-master for high availability

→ Master runs Control plane to run cluster smoothly.

Constituents of Control plane

kube-apiserver:-

→ this apiserver interacts directly with user (ie, we apply yaml or json manifest to kube-apiserver)

→ this kube-apiserver is meant to scale automa-tically as per load.

→ kube-api server is front end of Control plane

etcd: (cluster Store)

→ Stores metadata and status of cluster

→ etcd is consistent and high available store (key-value store)

→ source-of-truth for cluster state (info about state of cluster)

Kube-scheduler:-

→ When users make request for the creation & management of pods, kube-scheduler is going to take action on these requests.

→ Handles pod creation and management

→ kube-scheduler match/assign any node to create and run pods

Controller-manager:-

→ Make sure actual state of cluster matches to desired state

→ Two possible choices for controller manager

1. if K8S on cloud, then it will be cloud-controller-Manager.
2. if K8S on non cloud, then it will be kube-Controller-Manager

Nodes (kube-proxy & Kubelet):-

What runs on each node of cluster?

→ node/minion is going to run 3 imp pieces of software

Kubelet:-

→ Agent running on the node

→ Listens to kubernetes master (ex:- pod creation request)

→ Port 10255

→ Sends success / fail reports to master

Container engine: (Dockers):-

→ Works with kubelet
→ Pulling images
→ Start/stop containers
→ Exposing containers on ports specified in manifest.

Nodes, Kube-proxy

Kube-Proxy:- (assigns IP to each pod)

→ It is required to assign IP addresses to pods (dynamic)
→ kube-proxy runs on each node & this make sure that each pod will get its own unique IP address.
→ Above 3 components collectively consists "node"

K8S for Hybrid & Multi-cloud

→ Hybrid:- On premise datacenter + Public cloud
→ Multi cloud:- More than one public cloud provider

What is a POD

→ Atomic unit of deployment in kubernetes
→ Consists of 1 or more tightly coupled containers
→ Pod runs on node, which controlled by master
→ kubernetes only knows about pods (doesn't know about individual containers)
→ Cannot start containers without a Pod

→ 1 pod usually contains 1 Container
→ Multi-container pods are possible too
→ Such containers are tightly coupled

PODS

Multi container pods:
→ Share access to memory space
→ Connect to each other using localhost:<container port>
→ Share access to the same volumes
Atomic unit of kubernetes
→ Containers within pod are deployed in an all-or-nothing manner
→ Entire pod is hosted on the same node (scheduler will decide about which node)

Pod runs on which node - Schedulers will decide

POD Limitations

→ No auto-healing or scaling
→ Pod crashes? must be handled at higher level
- Replica set
- Deployment

Higher level kubernetes objects

Replication set:-
- Scaling & Healing

Deployment:-
- Versioning and rollback

Service:-
- Static (non-ephemeral) IP and networking

Volume:-
- Non-ephemeral storage

Important

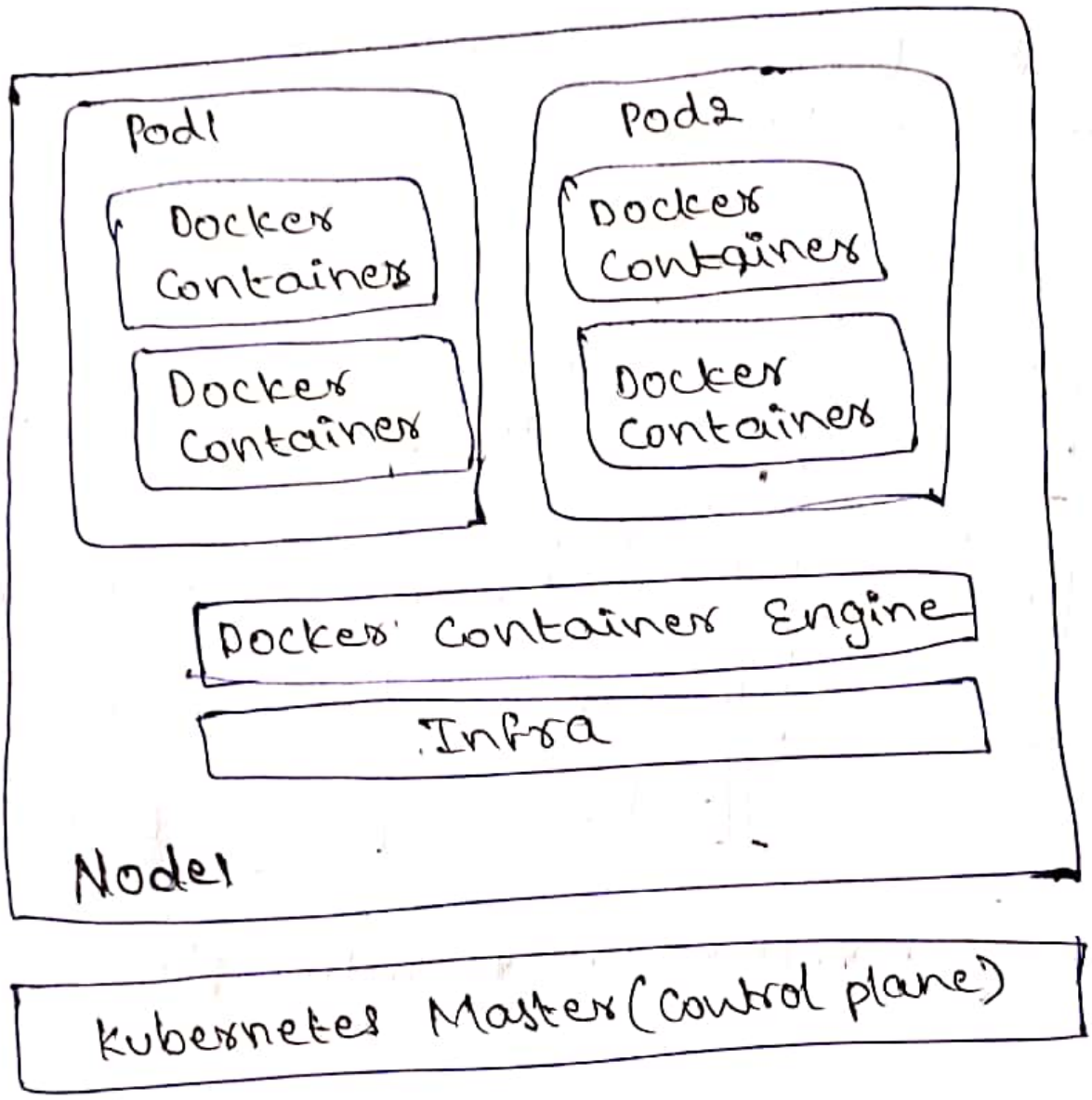
kubectl - single cloud
kubeadm - on premise
kubefed - federated

2 object management methods

→ Imperative method
→ Declarative method

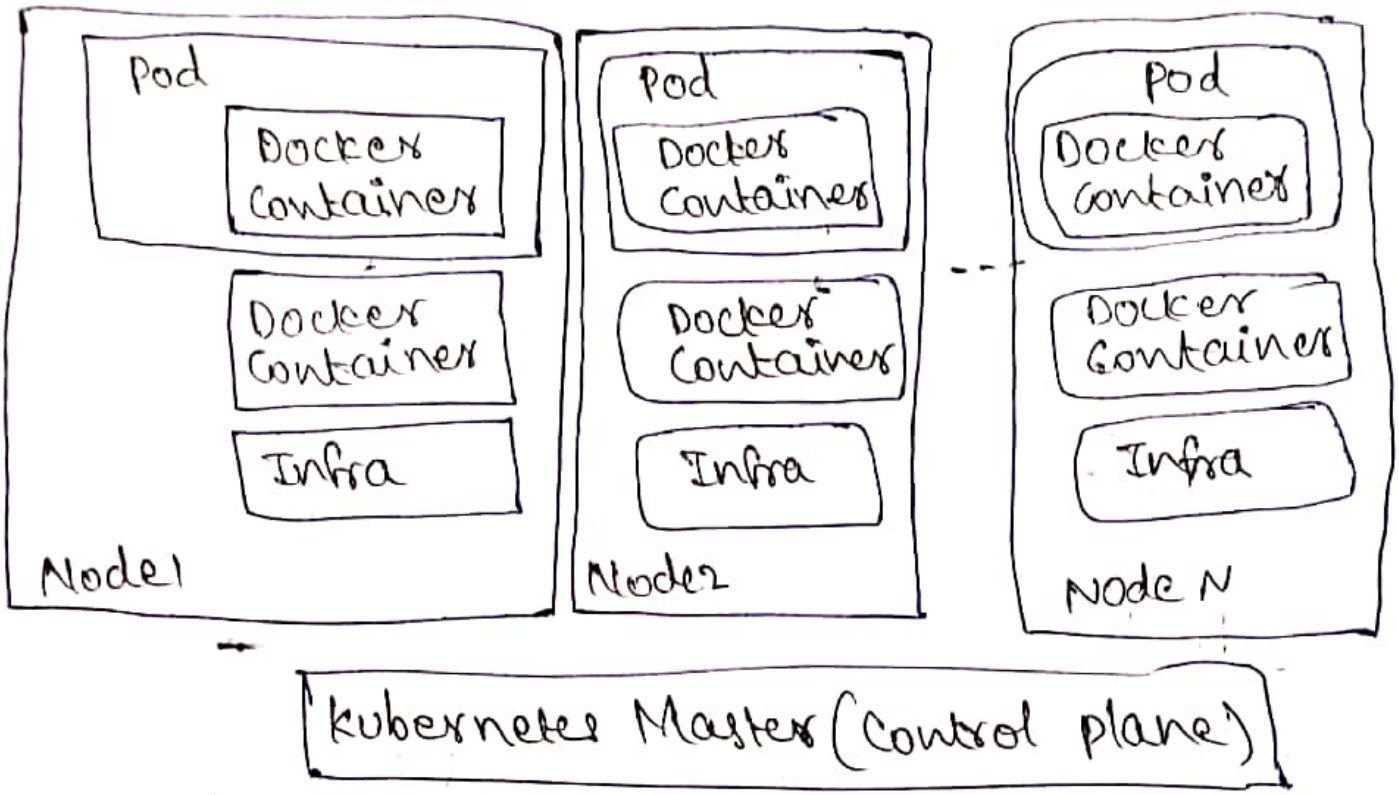
Imperative:- We say actions that we want kubernetes to take

Declarative:- We tell only require output, won't tell how to get

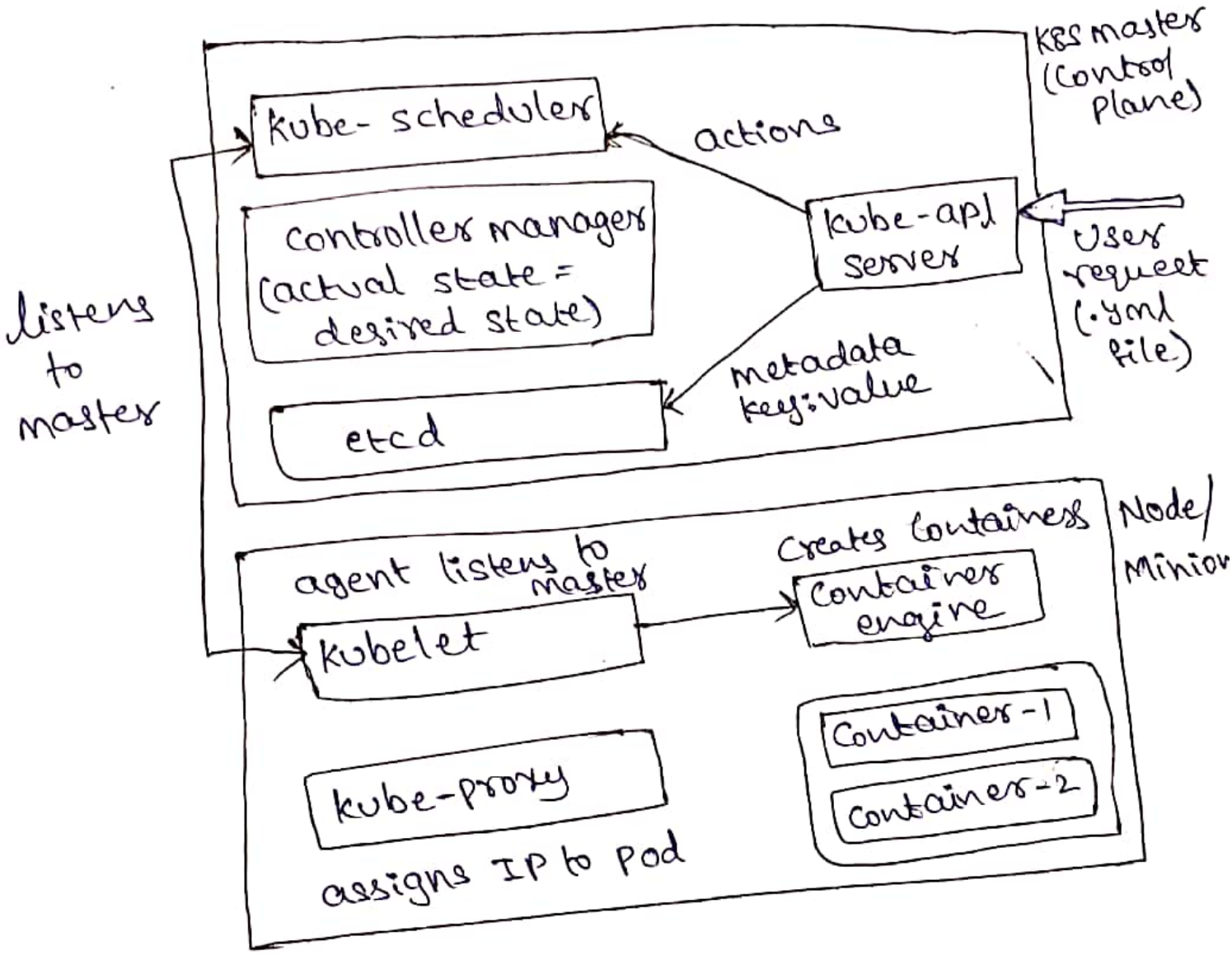


Kubernetes: Cluster Orchestration

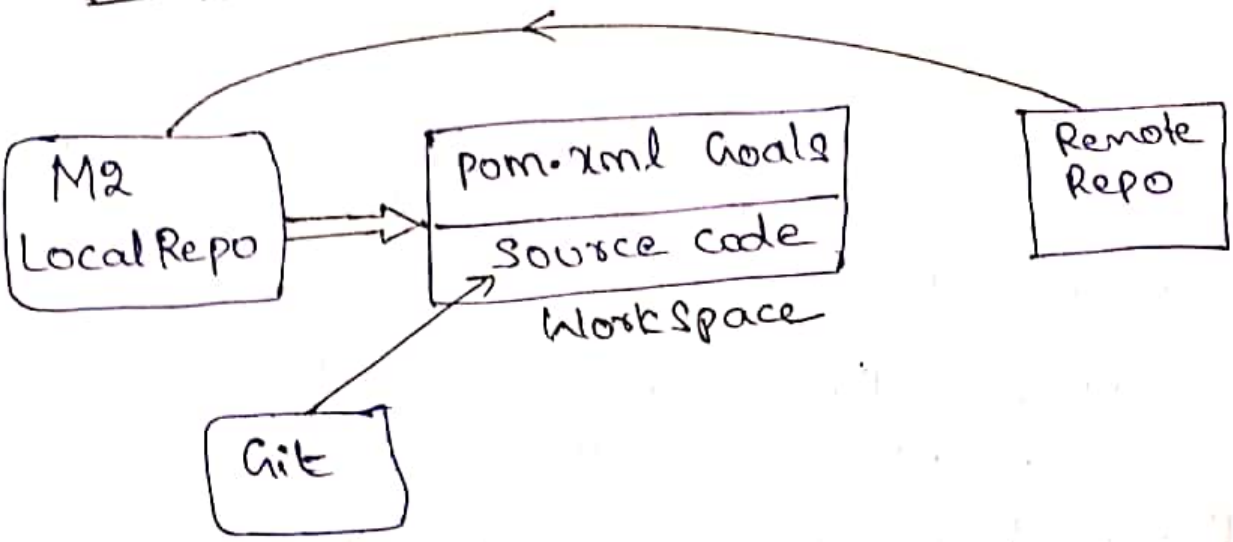
Potentially thousands of containers on hundreds of VMs



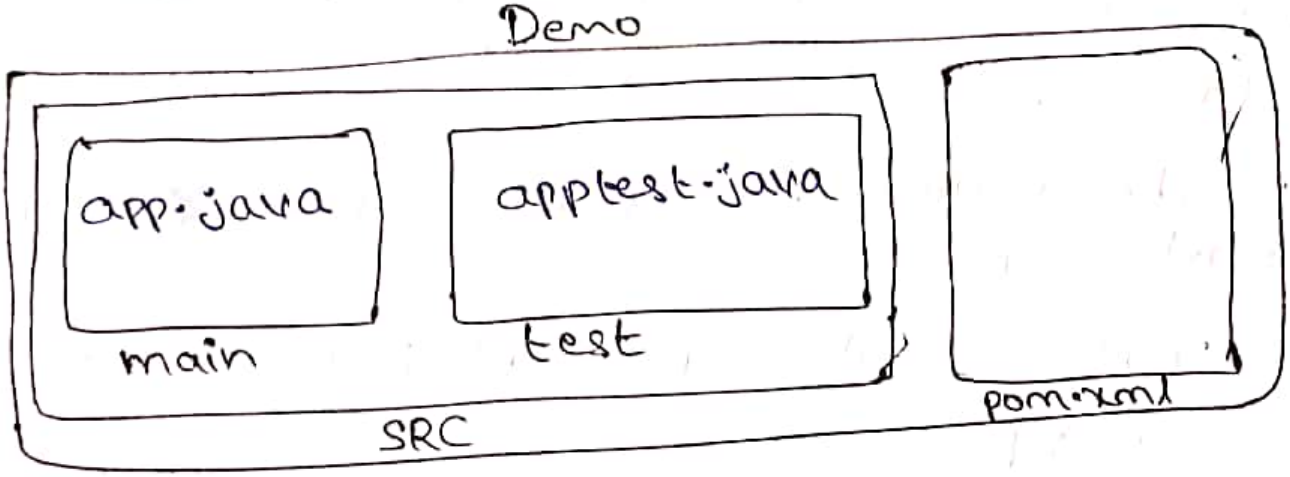
Kubernetes Architecture



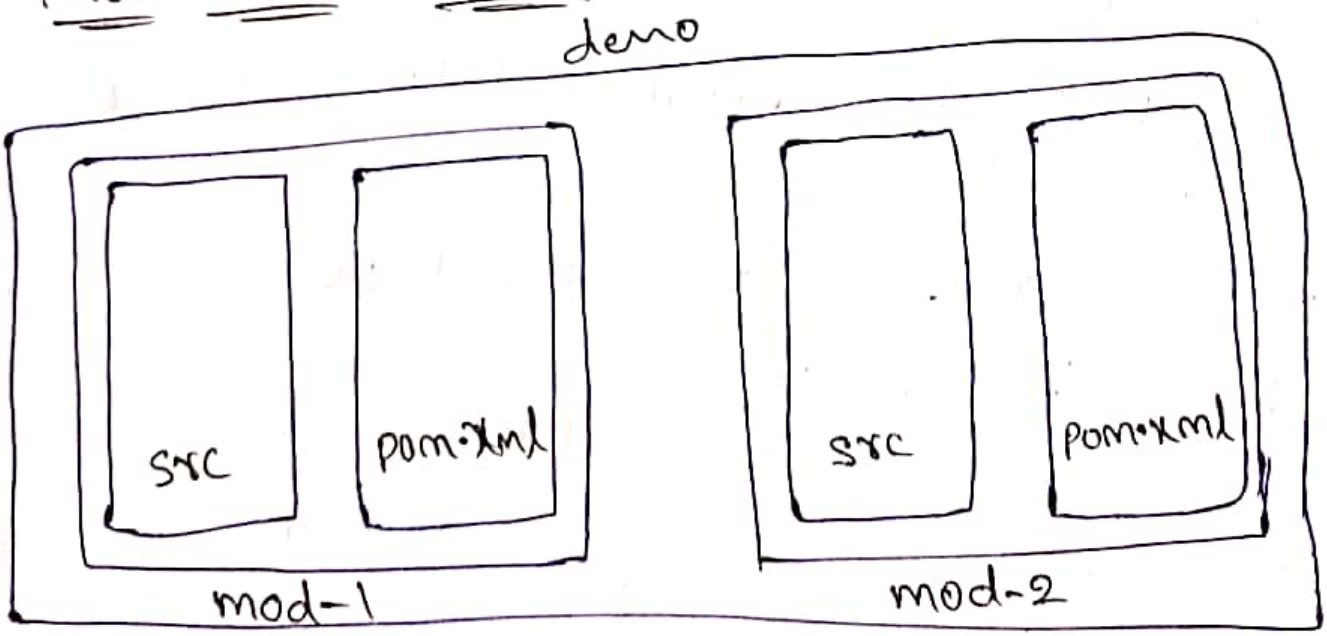
Architecture of Maven



Maven Directory Structure



Multi-module project

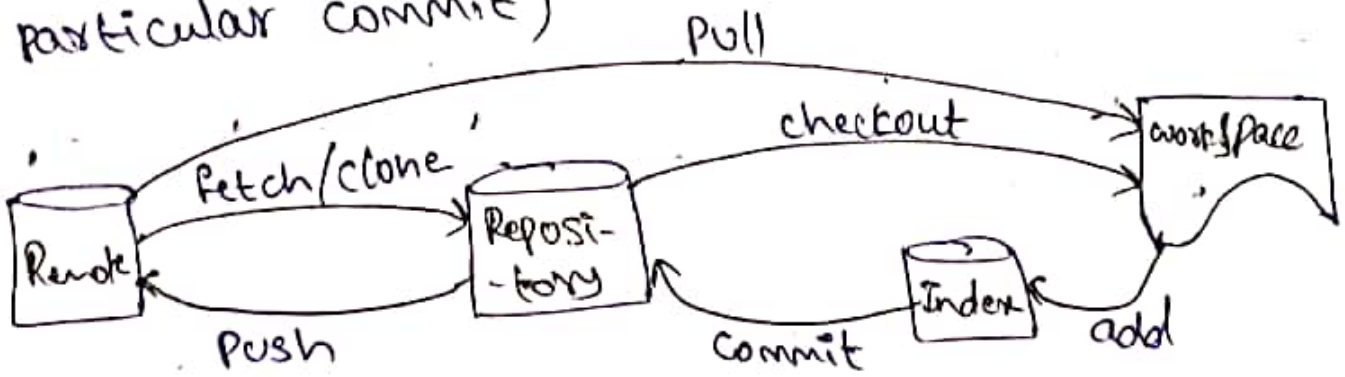


→ Pull vs Fetch

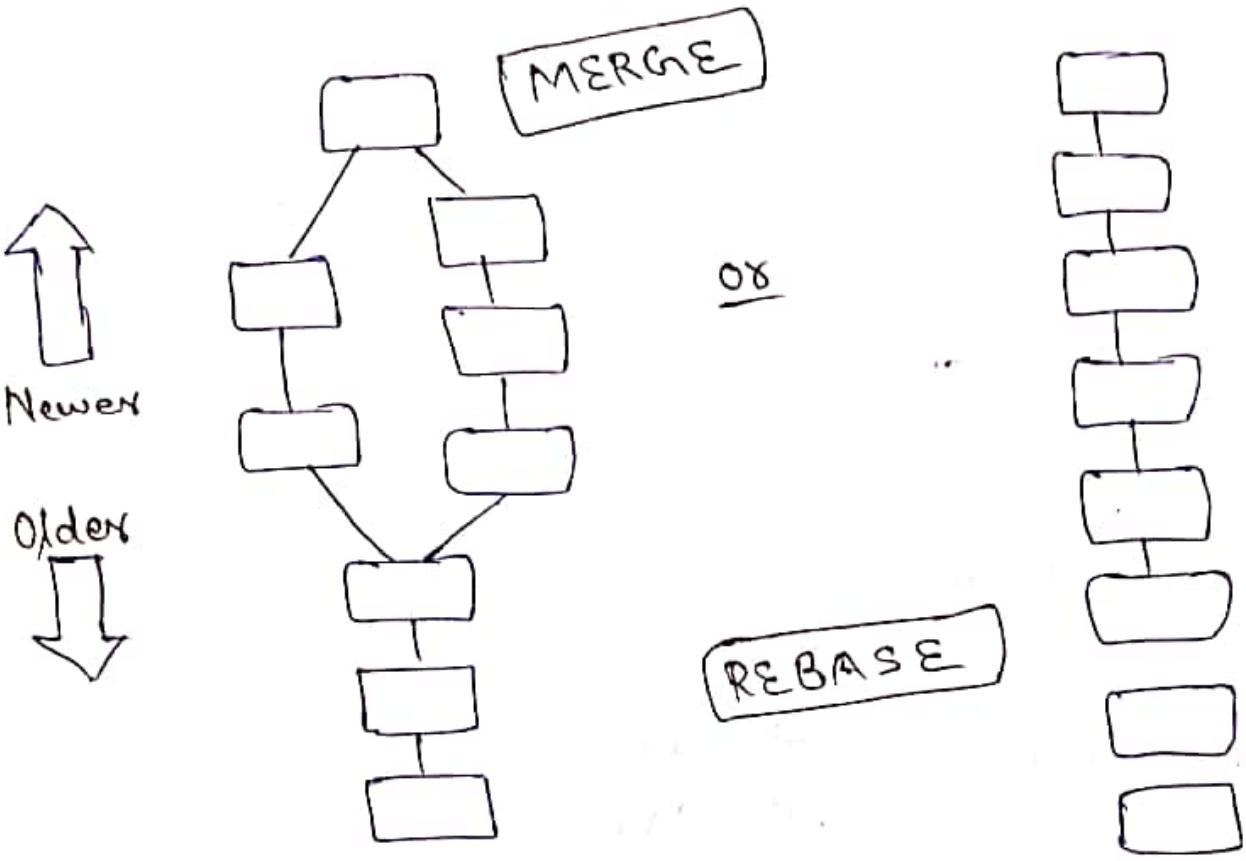
→ Pull = Fetch + Merge
→ Fetch = Download from central repo
→ Merge = Installed the downloaded one
→ Pull = Download + Install

→ Run git fetch (verify content in file in workspace & commits. there won't be any new change)
Fetch:- Download objects and refs from another repository
pull:- Fetch from and integrate with another repository or a local branch

→ git diff master origin/master (before pull, if want to verify)
→ git push -u -all (to push all branches to central repo)
→ git branch -D new (to delete branch (be in other branch))
→ git push origin newbranch (to push to newbranch)
→ git cherry-pick <commit-id> (to merge particular commit)



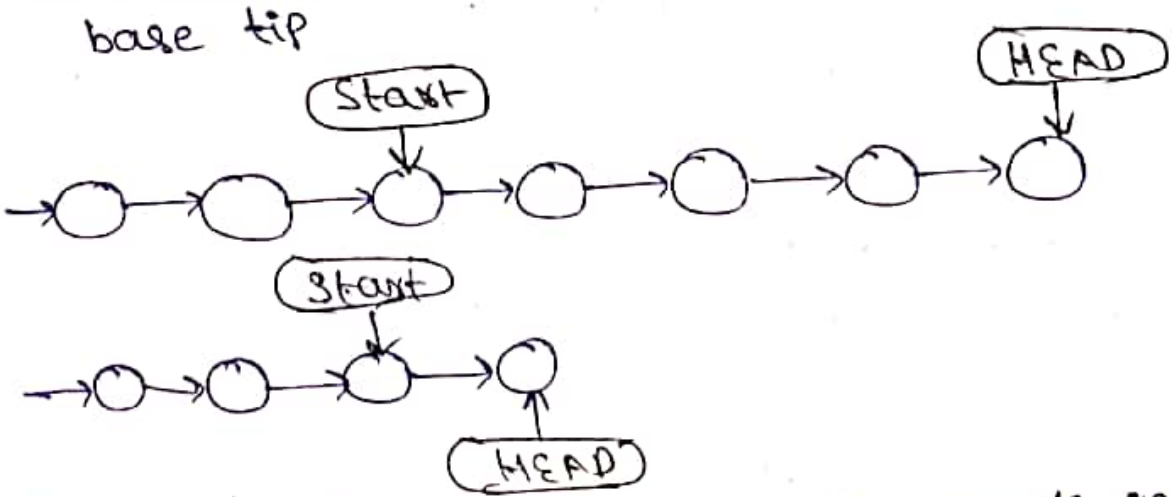
- Git hooks (Web hooks):- To set permissions & configure email notifications



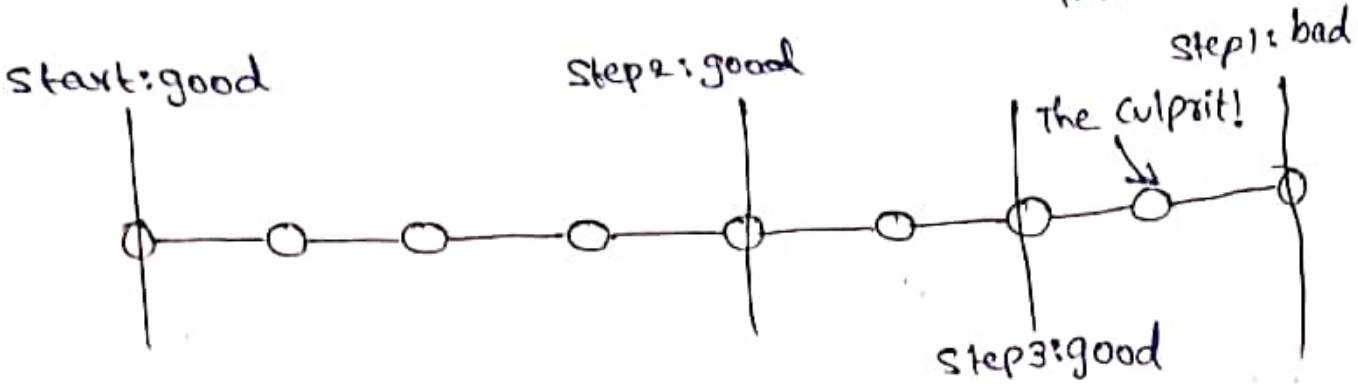
Git merge vs rebase

merge:- Join two or more development histories together

rebase: Reapply commits on top of another base tip



Git Squash:- To move the multiple commits into its parent so that you end with one commit. If you repeat this process multiple times, you can reduce n commits to a single one



CMD

→ Vi Dockerfile
→ FROM ubuntu:16.04
→ CMD echo 'Welcome to docker'
→ docker build -t app
→ docker run -it app
→ can see welcome to docker
→ docker run -it app echo 'Welcome to AWS'
→ can see welcome to AWS
→ ie., runtime command will overcorites the argument mentioned in CMD in docker file
→ Vi Dockerfile
→ FROM ubuntu:16.04
→ CMD echo 'Welcome to docker'
→ CMD echo 'Welcome to Python'
→ docker build -t app1
→ docker run -it app1
→ can see welcome to Python(ie., takes latest)

Entrypoint

→ Vi Dockerfile

```
FROM ubuntu:16.04
ENTRYPOINT ["echo", "Hi Docker"]
```

→ docker build -t app3
→ docker run -it app3

(Can see Hi Docker)

→ ENTRYPOINT:- We can't overwrite ENTRYPOINT instructions by runtime command

→ docker run -it app3 echo "Hi Hari"

→ can see Hi Docker echo Hi Hari

→ ie., runtime command will overwrites the argument mentioned in CMD instruction in docker file

→ ie., runtime command will not overwrites the argument mentioned in ENTRYPOINT instruction in docker file

→ Combinely, we can use

```
FROM ubuntu:16.04
ENTRYPOINT ["echo"]
CMD ["Hi Docker"]
```

→ docker build -t app4
→ docker run -it app4

→ Can see Hi Docker

→ docker run -it app4 "Hi Hari"

(Can see Hi Hari)

Install and Setup Java environment

→ Oracle Java Installation (in Google) (choose oracle Technology Network)
→ Accept license Agreement
→ Download Windows x64 installer & Run
→ Go to Environment Variables
→ In Environment variables (both user variables & System) (JAVA_HOME & JDK installed path)
→ java -version (in cmd) (if unable to see, restart PC)

Download Jenkins

→ Jenkins Installer (In google)
→ Downloads
→ Long-term Support
→ Download Generic Java Package (.war)

Install Tomcat

→ Tomcat installer Windows (In Google)
→ apache.tomcat.... cgi (link)
→ Binary Distributions - Core - Windows Service Installer & Run
→ username & password - admin (Tomcat default port is 8080)
→ localhost:8080 (to verify)
→ Manager App - admin & admin
→ You can see apps come with tomcat by default

→ All these apps installed in program files - ApacheSoft-wareFoundation - Tomcat - webapps (this is where we are going to deploy jenkins war file) 156

Install Jenkins as a Webapp on Tomcat

→ Create JENKINSHOME folder in c:\drive
→ Go to Env variable, set system variable (JENKINS_HOME & path of (inside) JENKINSHOME)
→ In cmd, echo %JENKINS_HOME% (can see JENKINSHOME) If not, restart PC (mandatory)
→ Copy the jenkins.war file that you downloaded to this tomcat webapps folder & start tomcat service (you can see jenkins folder got created) (start Jenkins in tomcat browser)
→ In browser, localhost:8080/jenkins (can see jenkins page) & give initial admin password which is in JENKINSHOME folder. (If unable to see Jenkins page, start Jenkins in tomcat & restart PC)
→ Select plugins to install - none - install - admin

ANT Installation

→ Download apache ANT
→ (ant.apache.org/bindownload.cgi)
→ Download bin.zip & unzip (1.9.9) & copy ANT folder in c:\drive)
→ Environment variables (in user variables) (ANT_HOME & ant downloaded path)
→ System variables - in path, (ant's bin directory downloaded path)
→ restart & ant -version & echo %ANT_HOME% (in cmd) (if unable, then restart PC (mandatory))

Clone Sample Java Code:-

→ Create GitWorkspace folder in c:\ & initialize git.

→ inside GitWorkspace - git clone https://github.com/saiDevOpsFaculty/SampleWebApp.git (in cmd)

Continues Integration (Create a job to pull from GitHub)

→ Manage jenkins - manage plugins - GitHub - Install without restart

→ New job - GithubPull - Free style project - OK

→ GithubPull - Configure - General - Advanced - Use Custom Workspace - ${JENKINS_HOME} /workspace/SampleWebApp

→ Source Code Management - Git - Copy the down HTTP url of SampleWebApp repo - uncheck pollscm - save

Continues Building: Build, Code review and Publish results

→ Manage Jenkins - manage plugins - checkstyle & Ant plugin - Install without restart

→ New Item - BuildandCodeReview - FreeStyleProject - OK

→ General - Advanced - Use Custom Workspace - ${JENKINS_HOME}/workspace/SampleWebApp

→ Sourcecode management - git - paste the url (down) of samplewebapp from github

→ Build - Invoke ANT - Targets - checkstyle

→ Post Build Actions - Publish checkstyle Analysis Results - checkstyle - errors.xml - save

Continuous Testing: Run tests and publish Junit test reports

→ Install JUnit Realtime Test Reporter plugin

→ New Item - Unit Test - freestyle - OK

→ General - Advanced - Use Custom Workspace - ${JENKINS_HOME}/workspace/SampleWebApp

→ Sourcecode management - git - paste the url (down) of samplewebapp from github

→ Build - Invoke ANT - junit

→ Post Build Actions - Publish JUnit Test Result Report - Test Calculator - JUnit Result.xml - save

Set Up New Deployment Server (Tomcat)

→ Go to Tomcat/conf/server.xml
Connector port "8090"

→ Go to Tomcat/conf/tomcat-users.xml
<user username="admin" password="admin" roles="manager-gui, manager-script, admin" />

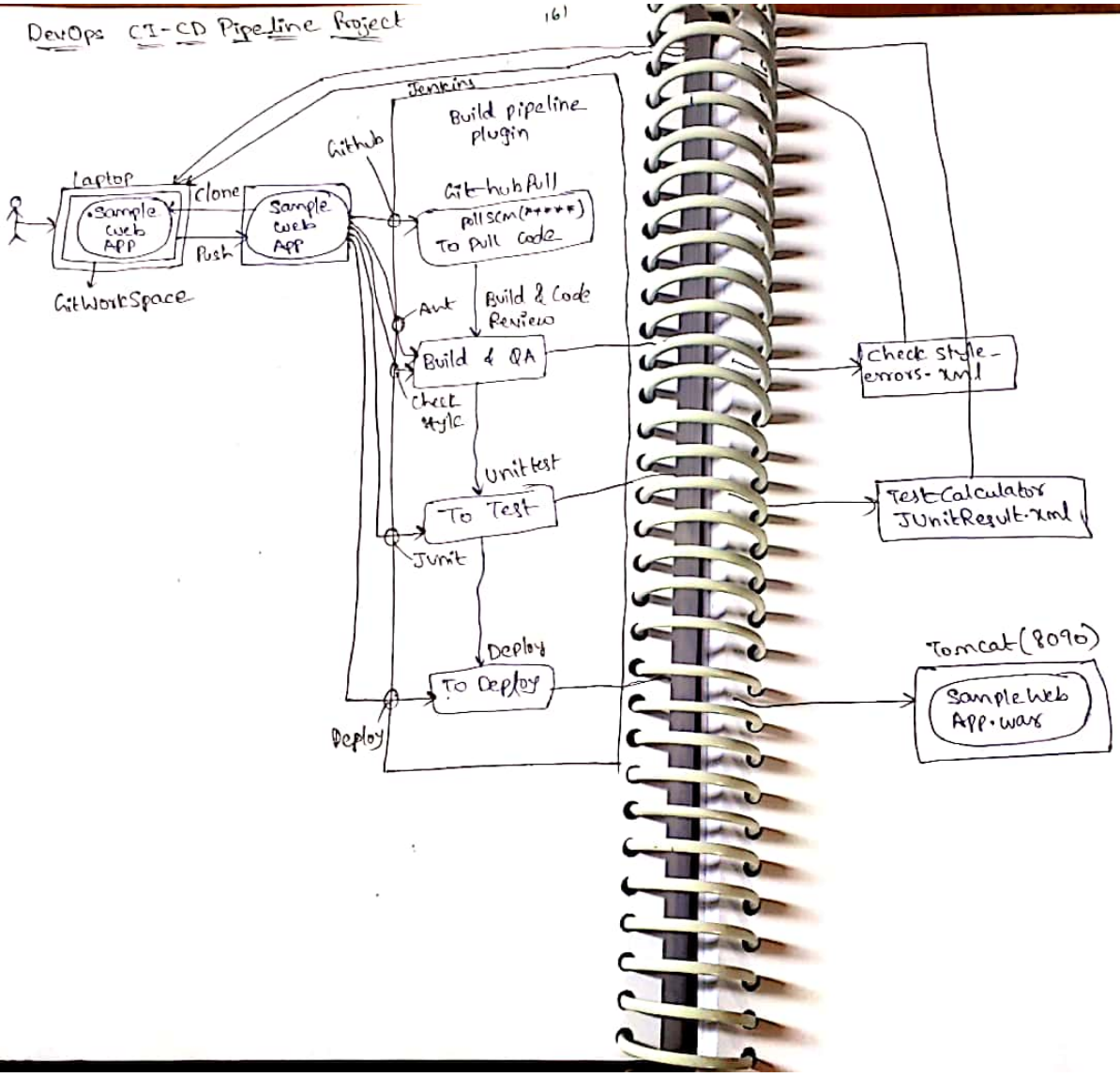
Continuous Deployment: Deploy code to production environment:-

→ Install Deploy to Container plugin
→ New Item - Deploy - freestyle - OK
→ General - Advanced - Use Custom Workspace - ${JENKINS_HOME}/workspace/SampleWebApp
→ git - paste the url of samplewebapp repo
→ Build - Invoke Ant - Target - war
→ Post Build Actions - Deploy War/Ear to container (WAR/EAR files - **/*.war & Context path - samplewebapp) - add container - Tomcat 8.x - admin - admin - (Credentials: admin) - http://localhost:8090 - save
→ Tomcat8.exe (Stop & Start tomcat service) to start tomcat

CI CD Build Pipeline

→ Install Build Pipeline plugin
→ Go to GitHubPull - configure - PostbuildAction - Build OtherProjects - CodeReview - save
→ Go to Build and CodeReview Job - Configure - Postbuild Action - BuildOtherProjects - Unit Test - save
→ Go to UnitTest Job : configure - PostbuildAction - BuildOtherProjects - Deploy - save
→ click on + - Complete_Pipeline - BuildPipeLine - View - ok - Select InitialJob - GithubPull - OK
→ Go to GithubPull - configure - PollSCM - * * * * *
(Can see all jobs in a pipe line view automa-tically being build)
→ localhost: 8090/samplewebapp
→ Can see web page is loading

DevOps CI-CD Pipeline Project



Installations

1. Git download
2. Git install
3. Java download
4. Java Install
5. Java Configure
6. Jenkins.war download
7. Tomcat Download
8. Tomcat Install
9. Jenkins Home
10. Restart PC
11. Place Jenkins.war in Tomcat
12. Download ANT
13. Configure ANT
14. Restart PC