10 January 2024 15:47

DEVOPS-NOTES-for-Preparation:

Session : 1

WHAT IS CLOUD?

let's assume that you are watching a cloud in the sky. But have you ever thought that you can watch the same cloud anywhere in your location. This is how it works, you can access the data/application via the internet at any time in anyway

WHY CLOUD?

If you are working on a project for 30 days and you saved all your data on your local PC. But unexpectedly your PC got broken. You lost all your project data.

NOTE: Whenever you saved your data in a physical device like Memory cards, CDs, and pen drives. If you lost these devices or if the devices gets crashed then the data should not be retrieved

But what if you saved all your files in to cloud.

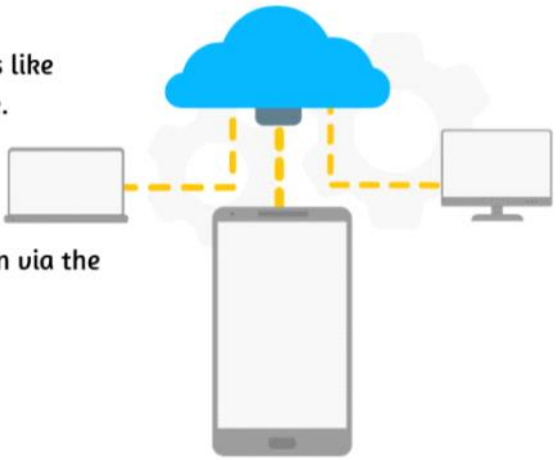
Now even if you lose your files now, no need to worry about data. Because all your files and folders are securely stored in the cloud.

WHAT IS CLOUD COMPUTING

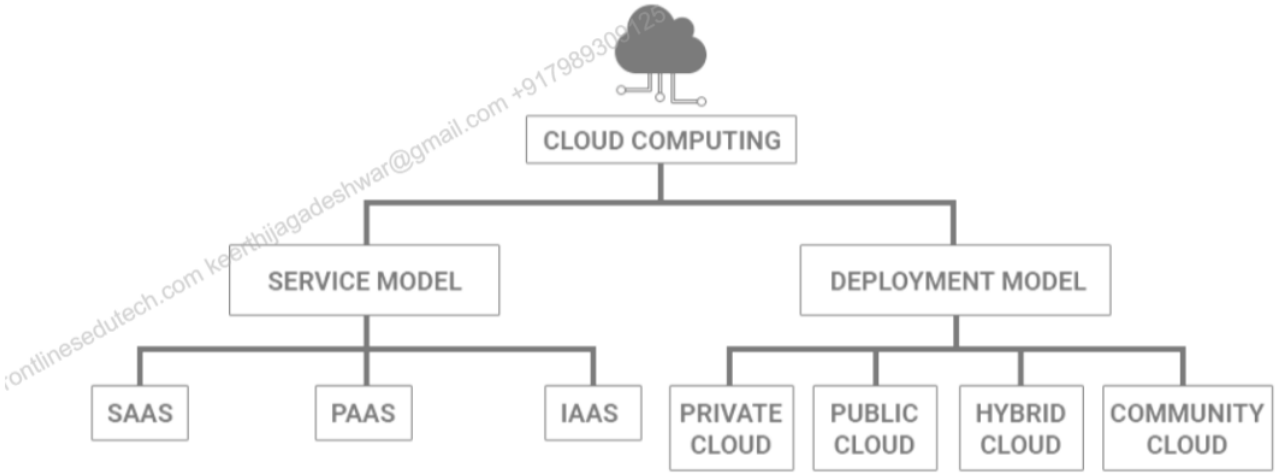
Cloud Computing is the delivery of computing services like Servers, Storage, Database, Networking and software.

or

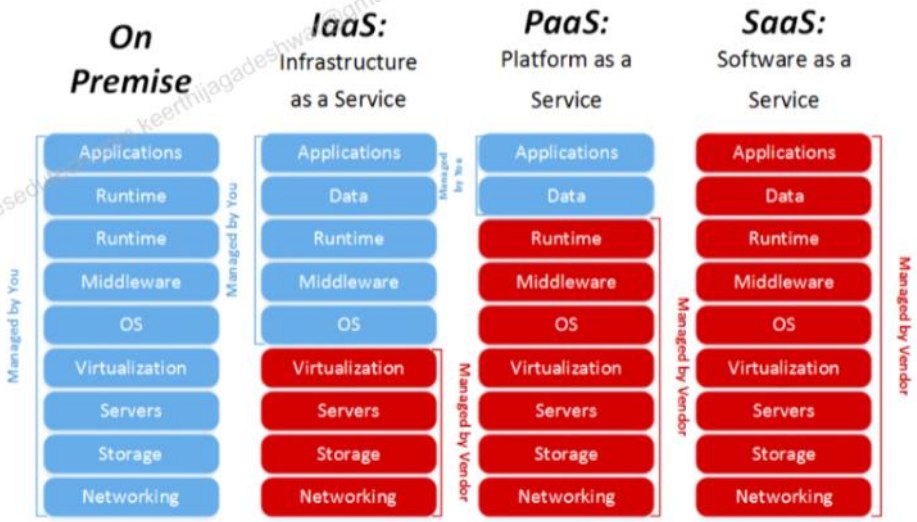
it is the processing of accessing the data or application via the internet



EX:



SERVICE MODEL:

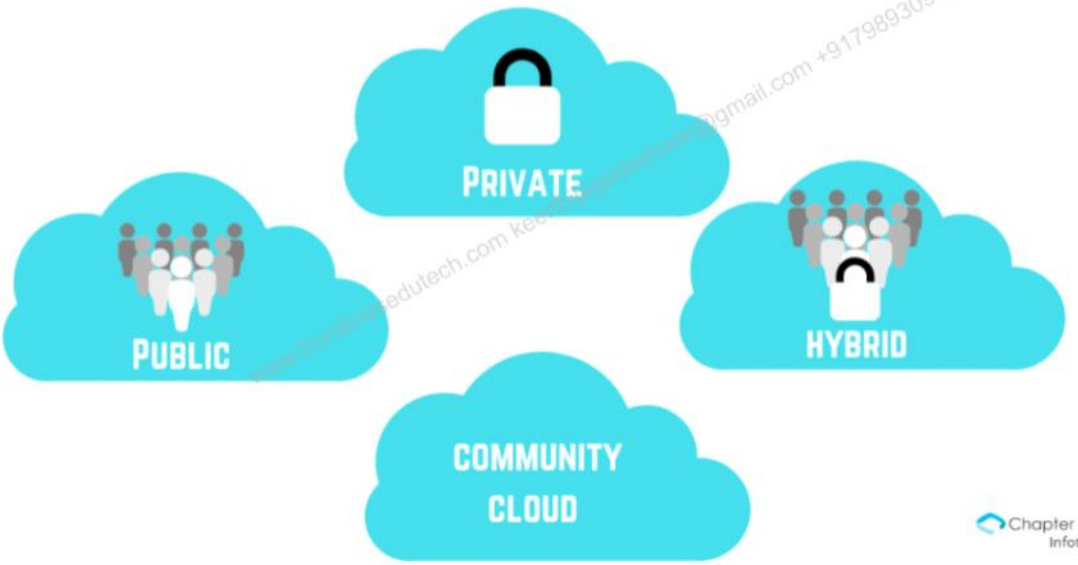


- **Infrastructure as a Service (IaaS):** In this model, the cloud provider offers virtualized computing resources, such as virtual machines, storage, and networking, to customers.
  Ex: **AWS** - used for infrastructure provisioned and managed over internet

- **Platform as a Service (PaaS):** This model provides a platform for customers to develop, run, and manage their own applications, without having to worry about the underlying infrastructure.
  Ex: **GoDaddy** - Someone manages the Hardware and OS, Someone will take care about security, patching, updates and maintenance. We need to handle the applications only.

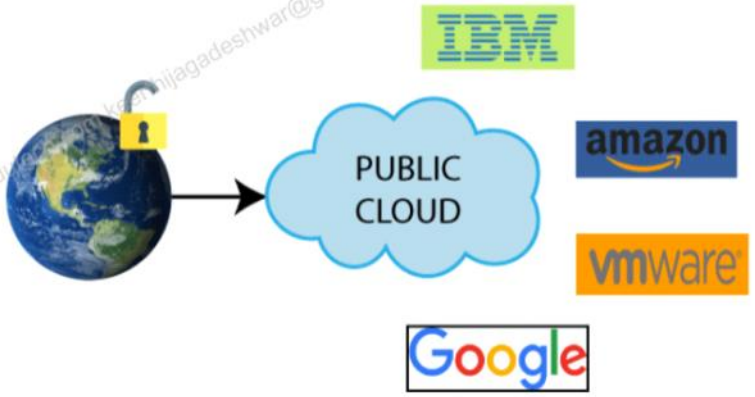
- **Software as a Service (SaaS):** This model provides software applications that are hosted and managed by the cloud provider, and can be accessed by users over the Internet.
  Ex: **Gmail** - You can manage inbox only, Google take cares of data centers, servers, network, storage maintenance etc .. All you need to worry about software and how you use it

DEPLOYMENT MODEL:



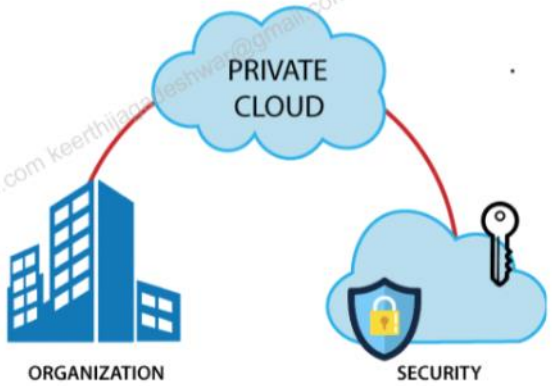
PUBLIC CLOUD:

Public cloud is open to all to store and access information via the Internet. Public clouds are managed by third parties. In public cloud, Security will be less when we compared to private and hybdrid.



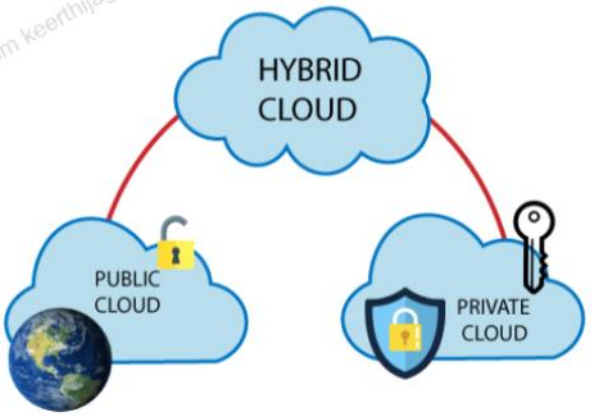
PRIVATE CLOUD:

A private cloud is also known as an internal cloud or corporate cloud. It is used by organizations to build and manage their data centers. In private cloud security will be high.



HYBRID CLOUD:

Hybrid cloud is the combination of both public cloud and private cloud.
If the services are running on public then it will not have much security.
If the services are running on private then the cloud have high security.



COMMUNITY CLOUD:

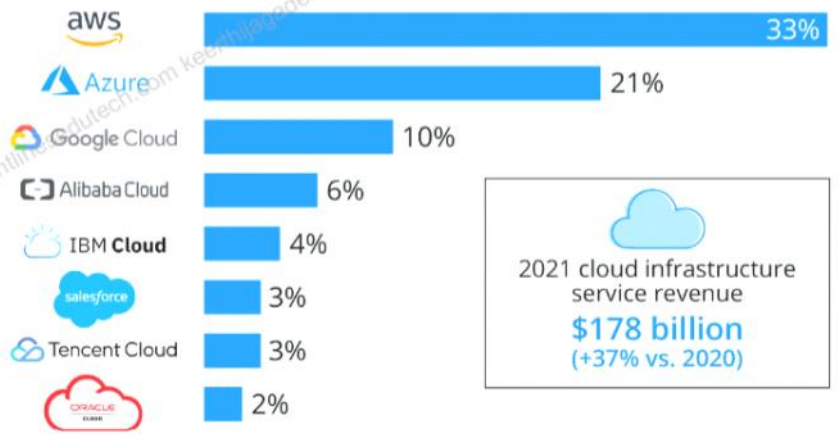
It allows multiple organizations to use same cloud to store their data.

www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125

TOP 10 CLOUD PROVIDERS

AMAZON LEADS $-180 BILLION CLOUD MARKET

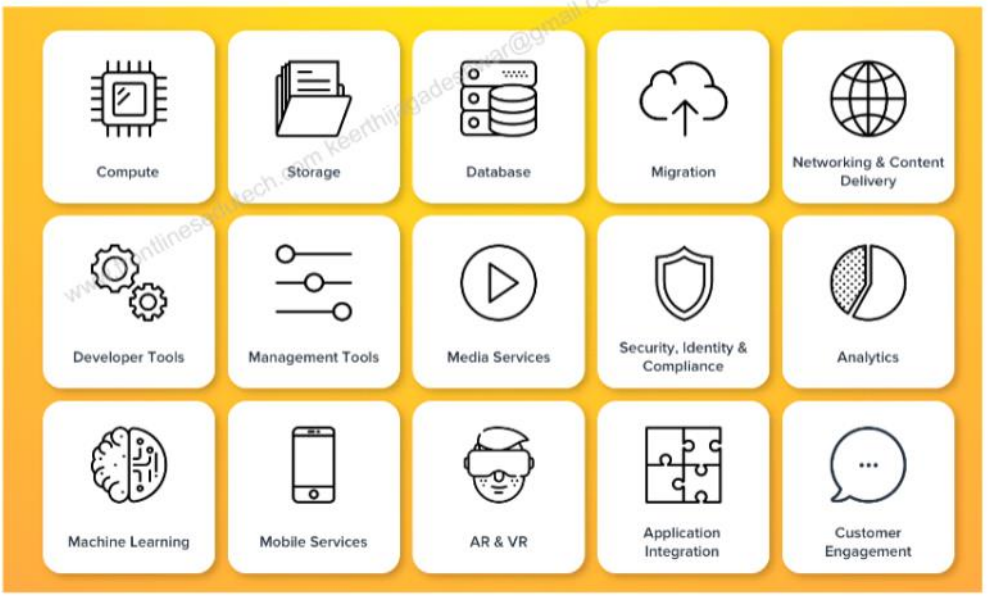
WORLDWIDE MARKET SHARE OF LEADING CLOUD INFRASTRUCTURE SERVICE PROVIDERS IN Q4 2021*



WHAT IS

- AWS is abbreviated as Amazon Web Services.
- It is the best cloud provider
- It is the first among all the clouds
- It offers multiple services on different domains.
- It is the combination of SAAS, PAAS, IAAS.
- It is one of the most popular cloud computing platform. Now a days most of the companies are using this cloud because of numerous that allow them to store their data easily without the need for a physical space.

AWS WEB SERVICES LIST



WHY AWS IS SO POPULAR?

- AWS provides a wide range of cloud computing services that can be used to build and run applications.
- It can offer 200+ services on multiple domains.
- AWS covers about 31 geographic regions around the world.
- AWS provides a wide range of security features and compliance certifications that help customers to secure their applications and data in the cloud.
- The AWS Cloud spans 99 Availability Zones.
- It has 7 years of experience compared to another cloud.
- AWS allows users to select different operating systems, databases, and languages.
- It follows the Pay as you go, Model.

KEY TERMS:

"AVAILABILITY ZONE "

..aka "Data Center"

"REGION"

..aka "Geographical Area"

***Each Region consists of 2 or more availability zones*

"EDGE LOCATION"

...aka "AWS Endpoint used to cache content"

**Reduces latency to end user

PAY AS YOU GO MODEL

IF YOU GO TO BARBIQUE NATION RESTURANT

AND ATE ONLY 1 PLATE OF BIRYNI

SERVER ASKS YOU TO PAY 800RS

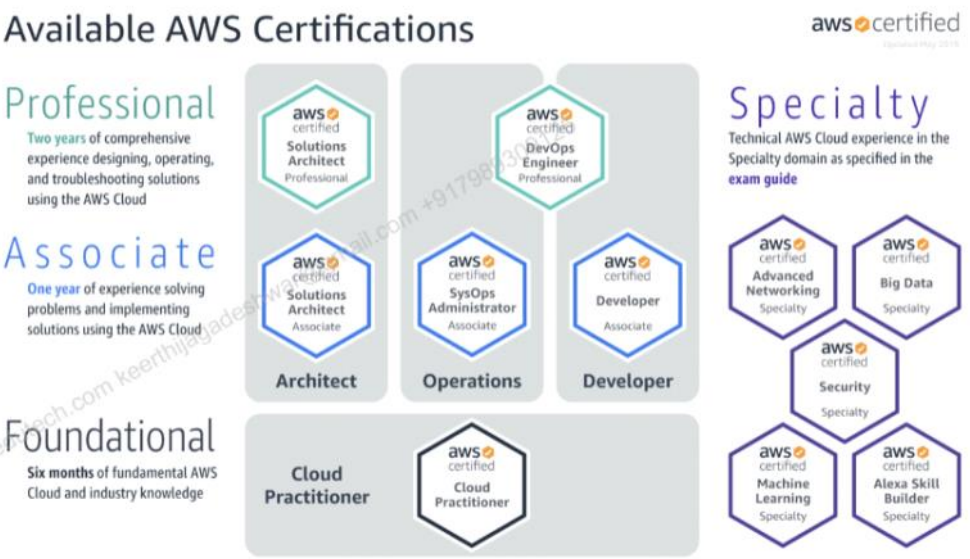
INSTEAD

IF YOU GO TO NORMAL RESTURANT THEY WILL CHARGE WHAT EVER YOU ORDER

ITEM	PRICE
BIRYANI	$60.00
STARTERS	$65.00
MILK SHAKES	$80.00

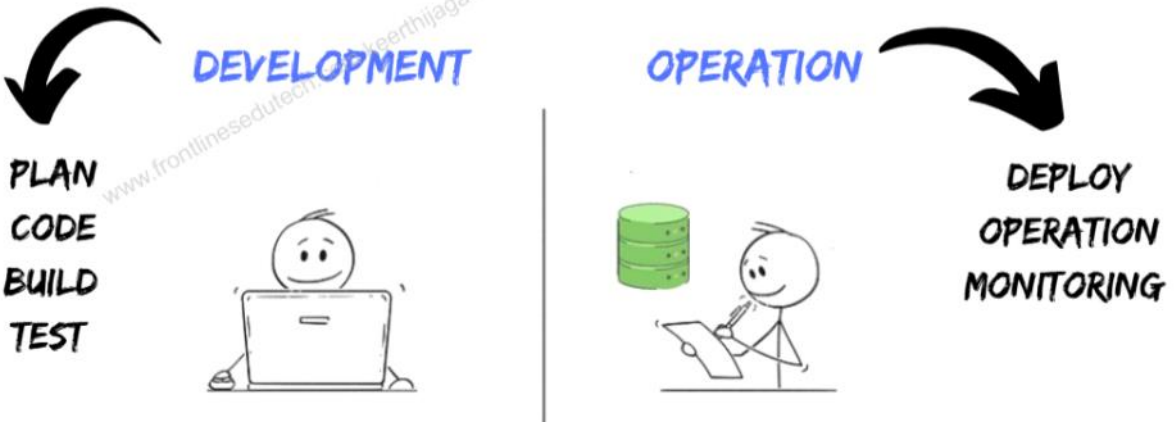
ADVANTAGES OF AWS

AWS CERTIFICATIONS



WHAT IS DEVOPS

IN SOFTWARE DOMAIN, WE HAVE 2 TEAMS FOR DEVELOPING THE APPLICATION



DEVELOPMENT

THE DEV TEAM WRITES THE CODE FOR THE ENTIRE APPLICATION AND TOSSES IT TO OPERATIONAL TEAM FOR PRODUCTION

OPERATIONAL

THE OPS TEAM DEPLOYS, MONITOR OPERATE & MAINTAIN THE APPLICATION

DEV TEAM

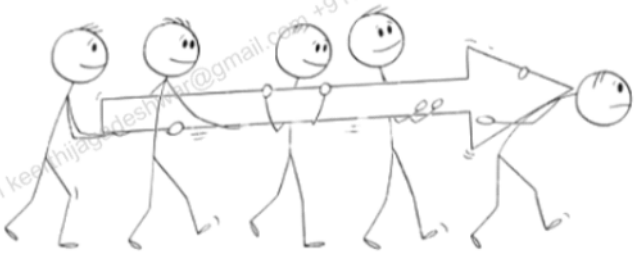
I Have given you the right code

OPS TEAM

I have deployed the code you given to me

DEVOPS IS NOT A TOOL, TECHNOLOGY OR FRAME WORK

DevOps is a cultural movement, mindset, philosophy to coordinate produce better, more reliable products

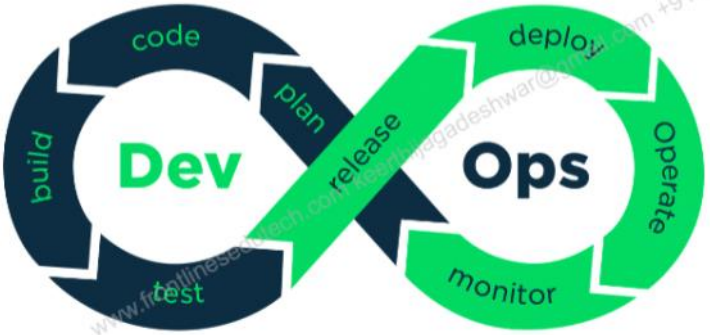


by automating infrastructure, workflow, and continuously measuring application performance for which they use a lot of tools

www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125

DEVOPS LIFE CYCLE

The Logo of DevOps represents infinite because its a never ending continuous process



Session : 2

Amazon Elastic Co...

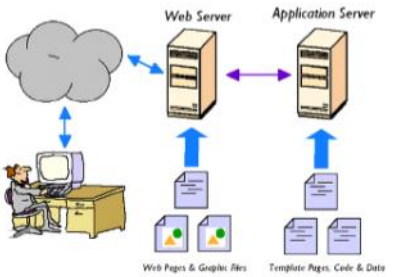
STEPS TO CREATE A...

SERVER:

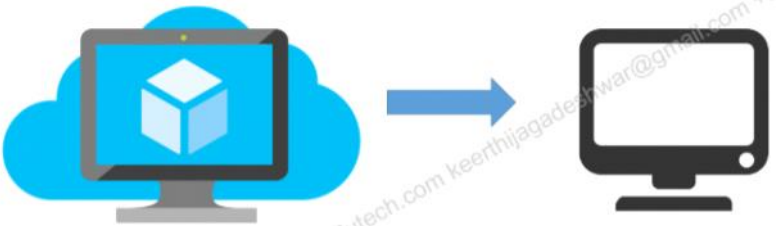
Servers are the computers that run services to serve the needs of other computers i.e. (to provide services to other computers). These server are used for various purposes like hosting websites, running applications, storing and managing data.

TYPES OF SERVERS:

- Web Server : Apache, Nginx, IIS (Internet information service), GWS(Google Web server)
- Application server : Apache Tomcat, F5 Nginix, IBM Websphere
- Database Server
- Email server
- FTP Server
- File server
- Proxy server
- Streaming server
- IRC Server (Internet relay chat)
- Fax server



VIRTUAL MACHINE:



A VM is a program on a computer that works like it is a separate computer inside the main computer. Each VM functions as a separate and isolated computer with its own operating system, storage, and network resources.

TYPES:

- SYSTEM VM'S
- PROCESS VM'S

WHAT IS EC2?

ELASTIC COMPUTE CLOUD IS A WEB SERVICE THAT PROVIDES SECURITY AND RESIZABLE COMPUTE CAPACITY IN THE CLOUD WHICH IS DESIGNED TO USED BY DEVELOPERS EASIER.

IT IS ONE OF THE SERVICES PROVIDED BY THE AWS WHICH WE CAN USE IT TO LAUNCH INSTANCES ON DIFFERENT OS.

TO LAUCH EC2 INSTANCE THERE WILL BE SEVEN STEPS NEEDS TO BE PERFOM.

STEP-1: Choose an Amazon Machine Image (AMI):

- An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance
- It consist of an AMI-ID which is region specific.
- We can buy, sell and share the AMI's.
- We need to use free tier AMI's only

STEP-2: Choose an instance type:

In the second step, we are providing CPU & MEMORY to our instance. Here we need to select the instance type which is under free tier eligible.

T2 MICRO (1 CPU AND 1 GB) AND T2 NANO (1 CPU AND 0.5 GB)

Total instance families are 90 available

All instance families	c5	c6id	g3s	im4gn	m5d	mac2	r5b	u-3tb1	z1d
t1	c5a	c7g	g4ad	inf1	m5dn	p2	r5d	u-6tb1	
t2	c5ad	cc2	g4dn	is4gen	m5n	p3	r5dn	u-9tb1	
t3	c5d	d2	g5	m1	m5zn	p3dn	r5n	vt1	
t3a	c5n	d3	g5g	m2	m6a	p4d	r6a	x1	
t4g	c6a	d3en	h1	m3	m6g	r3	r6g	x1e	
a1	c6g	dl1	i2	m4	m6gd	r4	r6gd	x2gd	
c1	c6gd	f1	i3	m5	m6i	r5	r6i	x2idn	
c3	c6gn	g2	i3en	m5a	m6id	r5a	r6id	x2iedn	
c4	c6i	g3	i4i	m5ad	mac1	r5ad	u-12tb1	x2iezn	

STEP-3: Configure your instance

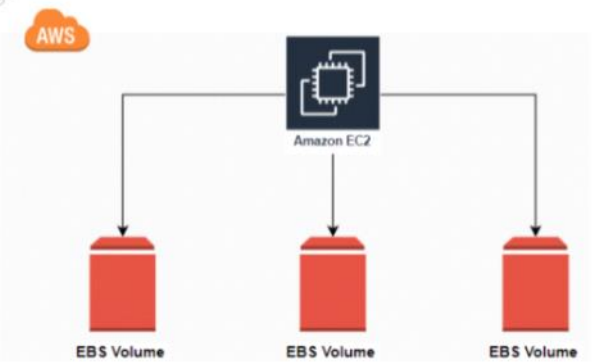
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

HERE YOU NEED TO CONFIGURE ALL YOUR INSTANCE DETAILS LIKE NO.OF INSTANCES , SUBNETS ,VPC , IAM ROLE , TENANCY ALL OTHER STUFF

STEP-4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes.

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage.



STEP-5: Add Tags

You can give a name to your instance

STEP-6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance.
These are region and network specific
The port range is 0 to 65535
It deals with the inbound and outbound traffic

STEP-7: Review & Launch

Session : 5

LINUX:

Linux is an free and Open-source operating system with high security. Linux is multi user based OS

OS: OPERATING SYSTEM

An Operating System (OS) is a software that acts as an interface between computer hardware components and the user.

Every computer system must have at least one operating system to run other programs. Applications like Browsers, MS Office, Notepad Games, etc., need some environment to run and perform its tasks.

The OS helps you to communicate with the computer without knowing how to speak the computer's language. It is not possible

TYPES:



KERNEL: manages the Hardware Components. CPU, memory, and peripheral devices. The kernel is the lowest level of the OS

DAEMONS: Manages are background services (printing, sound, scheduling, etc.) that either start up during boot or after you log into the desktop.

SHELL: is an environment in which we can run our commands, programs, and shell scripts. It gathers input from you and executes programs based on that input. When a program finish executing, it displays that program's output

LINUX OS DISTRIBUTIONS:

Many of the users taken the linux OS and modified according to their requirements and released into the market with different names called Linux distribution.

- RedHat
- Ubuntu
- Debian
- Centos
- Fedora
- Opensuse
- Kali Linux
- Amazon Linux
- Rocky Linux

HISTORY:

In 1991, Linus Torvalds a student at the university of Helsinki, Finland, thought to have a freely available academic version of Unix started writing its own code.
After this project became the Linux kernel.

The Linux kernel is written in C language.
He wrote this program specially for his own PC

Firstly he wanted to name it as 'Freax' but later it became 'Linux'.
In 1992, he released the kernel under GNU General Public License.

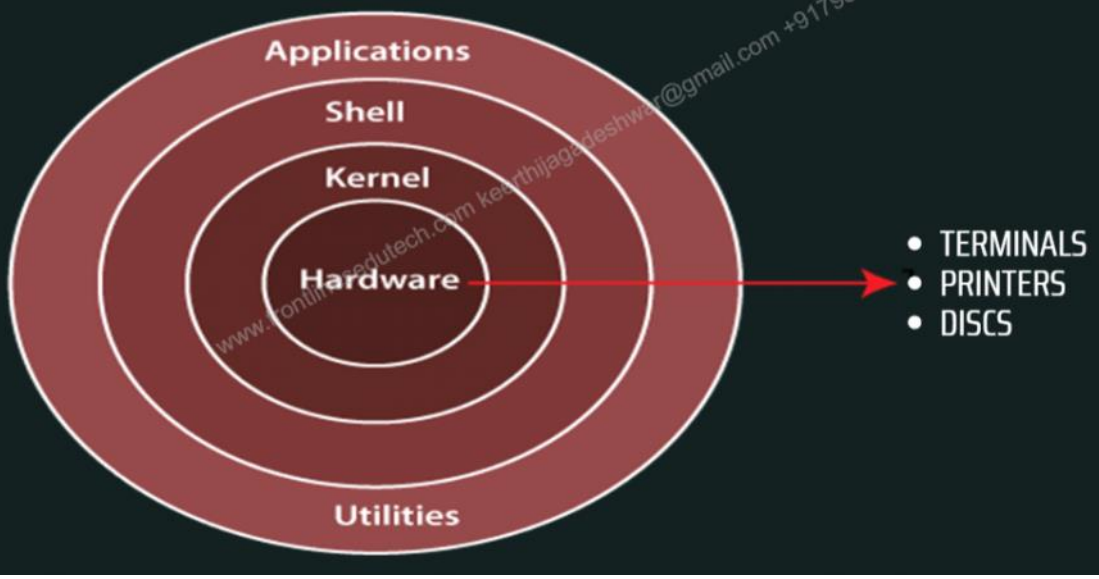
Today, supercomputers, smart phones, desktop, web servers, tablet, laptops and home appliances like washing machines, DVD players, routers, modems, cars, refrigerators, etc use Linux OS.

OPEN SOURCE

Linux is also distributed under an open-source license. Open source follows these key tenants:

- It is made freely available to the public, allowing anyone to view, use, modify, and distribute the code as they see fit.
- The freedom to run the program, for any purpose.
- The freedom to study how the program works, and change it to make it do what you wish.
- The freedom to redistribute copies so you can help your neighbour.
- The freedom to distribute copies of your modified versions to others.

ARCHITECTURE:



SHELL:

- A shell is a program that serves as an interface between the user and the operating system. It provides a way for users to interact with the computer by accepting commands and then executing them.
- Shells can be graphical (GUI-based), where users interact with the computer through windows, icons, and buttons, or they can be command-line-based, where users type text commands into a terminal.

COMMAND:

- A command is a specific instruction or request given to a computer's operating system or shell. It tells the computer to perform a particular action or task.

TERMINAL:

- A terminal, also known as a command-line interface (CLI), shell, or console, is a text-based interface for interacting with a computer's operating system. It provides a way for users to enter commands and receive text-based output.
- Terminals are commonly used in Unix-based and Linux operating systems, but they also exist in Windows (Command Prompt or PowerShell) and macOS.

LINUX
DAY-1

Session : 6

LINUX
CLASS-2

Session : 8

LINUX
DAY-3

Session : 9

LINUX
CLASS-4

Session : 10

LINUX
CLASS-5

LINUX
CHEAT SH...

Session : 11

GIT CLASS-1
(1)

Session: 12

GIT CLASS-2
(2)

Session : 17

GIT CLASS-5

GIT CLASS-5

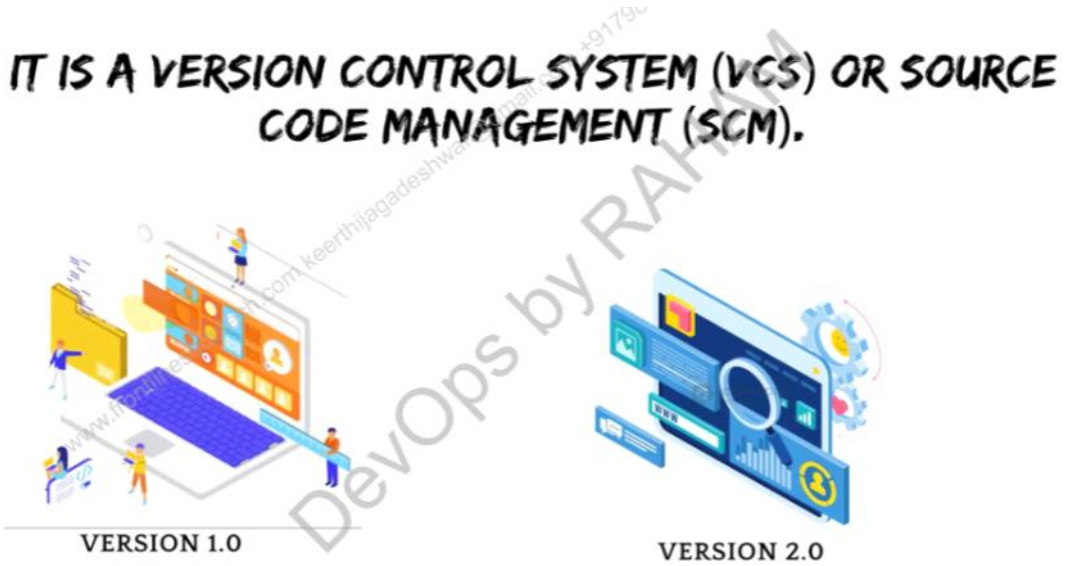
Session : 21

GIT INTERVIE...

git interview ...

GIT LAST CLASS

GIT CHEAT SHEET - G...



VERSION 1.0 CODE

VERSION 2.0 CODE

VERSION 3.0 CODE

VERSION CONTROL SYSTEM (VCS)

DevOps by RAHAM

It will maintain multiple versions of the same file.
It is platform-independent.
It is free and open-source.
They can handle larger projects efficiently.
They save time and developers can fetch and create pull requests without switching.

VCS HISTORY

SCCM:
To track only one file
[1972]

RCS:
Track multiple files
but not directories

CSV:
Track multiple files
and directories but
single user

SVN:
Track multiple files
and directories
[2000]

GIT:
Distributed Version
Control System
[2005]

WHY GIT?

If client asks you to develop application

You can develop according to client requirements.

you released version-1 of the application

One year later, the client comes to you and asks to change the options in the application

But the application got failed, In this case you can rollback to specific previous version.

GIT is a Distributed Version Control System

GIT tracks changes you made, so you have a record of what has been done, and you can revert to specific versions. it makes collaboration easy, allowing changes by multiple people to all merged into one source

DevOps by RAHAM
www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125

GIT ADD:

- Git add command is a straightforward command. It adds files to the staging area.
- We can add single or multiple files at once in the staging area.
- Every time we add or update any file in our project, it is required to forward updates to the staging area.
- The staging and committing are co-related to each other.

FILES GIT GIT FILES

TYPES OF REPOSITORIES:

LOCAL REPO:

The Local Repository is everything in your .git directory. Mainly what you will see in your Local Repository are all of your checkpoints or commits. It is the area that saves everything (so don't delete it).

REMOTE REPO:

The remote repository is a Git repository that is stored on some remote computer.

CENTRAL REPO:

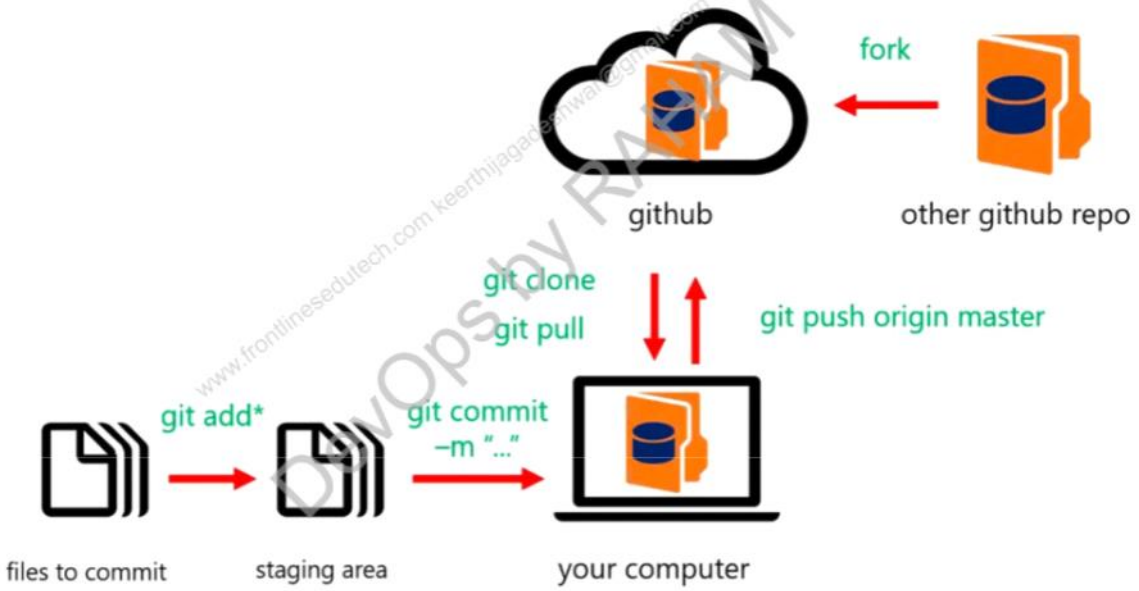
This will be present in our GITHUB

GIT ALTERNATIVES:

GIT LAB, SVN, BIT BUCKET, P4, STASH, HELIX

DevOps by RAHAM

GIT WORK FLOW:



GIT INSTALLATION:

- yum install git -y
- git init .

COMMIT A FILE:

- Create a file : touch filename
- Now add that file : git add . (Dot represents current directory)
- commit the file with message : git commit -m "commit message you want" filename
- To see details of that file : git log

GIT ADD:

- Git add command is a straightforward command. It adds files to the staging area.
- We can add single or multiple files at once in the staging area.
- Every time we add or update any file in our project, it is required to forward updates to the staging area.
- The staging and committing are co-related to each other.

DevOps by RAHAM

GIT COMMIT:

- It is used to record the changes in the repository.
- It is the next command after the git add.
- Every commit contains the index data and the commit message.

GIT STATUS:

- The git status command is used to display the state of the repository and staging area.
- It allows us to see the tracked, untracked files and changes.
- This command will not show any commit records or information.

GIT CONFIGURE:

- if you want to give your username and E-mail id to those commits then
  - git config user.name "username"
  - git config user.email "userxyz@gmail.com"

NOTE: now give the git log command to see changes, it won't work because after configuring we haven't done anything.
Now create a file and commit that file and give git log you will see changes as you configure.

GIT IGNORE:

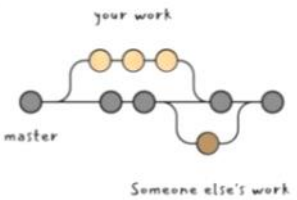
- It will be useful when you don't want to track some specific files then we use a file called .gitignore
- create some text files and create a directory with "jpg" files.
  - vi .gitignore
  - *.txt

Now all the txt files will be ignore

GIT BRANCHES:

- A branch represents an independent line of development.
- The git branch command lets you create, list, rename, and delete branches.
- The default branch name in Git is master.

To see current branch	:	git branch
To add new branch	:	git branch branch-name
To switch branches	:	git checkout branch-name
To create and switch at a time	:	git checkout -b branch-name
To rename a branch	:	git branch -m old new
To clone a specific branch	:	git clone -b branch-name repo-URL
To delete a branch	:	git branch -d <branch>



The -d option will delete the branch only if it has already been pushed and merged with the remote branch. Use -D instead if you want to force the branch to be deleted, even if it hasn't been pushed or merged yet. The branch is now deleted locally.

Now all the things you have done is on your local system.
Now we will go to GIT HUB.

DevOps by RAHAM

EXTRA TOPICS:

GIT MERGE CONFLICTS:

GIT makes merging super easy!
CONFLICTS generally araise when two people two people have changed the same lines in a file (or) if one developer deleted a file while another developer is working on the same file!
In this situation git cannot determine what is correct!

Lets understand in a simple way!

cat>file1 : hai all
add & commit
git checkout -b branch1
cat>file1 : 1234
add & commit
git checkout master
cat>>file1 : abcd
add & commit

git merge branch1 : remove it

IDENTIFY MERGE CONFLICTS:

see the file in master branch then you will see both the data in a single file including branch names

RESOLVE:

open file in VIM EDITOR and delete all the conflict dividers and save it!
add git to that file and commit it with the command (git commit -m "merged and resolved the conflict issue in abc.txt")

GIT REBASE:

if you have 5 commits in master branch and only 1 commit in devops branch, to get all the commits from master branch to devops branch we can use rebase in git. (command: git rebase branch_name)

GIT CHERRY-PICK:

if you have 5 commits in master branch and only 1 commit in devops branch, to get specific commit from master branch to devops branch we can use cherry pick in git. (git cherry-pick commit_id).

DevOps by RAHAM

GIT STASH:

Using the git stash command, developers can temporarily save changes made in the working directory. It allows them to quickly switch contexts when they are not quite ready to commit changes. And it allows them to more easily switch between branches.
Generally, the stash's meaning is "store something safely in a hidden place."

- git stash
- git stash apply
- git stash list
- git stash remove
- git stash pop
- git stash clear

SOME EXTRA COMMANDS:

git show <commit> --stat : you'll see the commit summary along with the files that changed and details on how they changed.
git commit --amend -m "New commit message" : to edit the commit message
git commit --amend --no-edit : used to add some files in previous commit. (--no-edit means that the commit message does not change.)
git update-ref -d HEAD : used to delete 1st commit in the git
git reset commit: used to delete all the commits (upto the commit id)
git commit --amend --author "Author Name <Author Email>" : used to change the author of latest commit

DevOps by RAHAM

MERGE VS REBASE:

When there are changes on the main branch that you want to incorporate into your branch, you can either merge the changes in or rebase your branch from a different point.

merge takes the changes from one branch and merges them into another branch in one merge commit.
rebase adjusts the point at which a branch actually branched off (i.e. moves the branch to a new starting point from the base branch).

Generally, you'll use rebase when there are changes are made in main/master branch that you want to include in your branch. You'll use merge when there are changes in a branch that you want to put back into main.

to merge: git merge branch_name
to rebase: git rebase branch_name

GIT-HUB

GitHub is a web-based platform used for version control.
It simplifies the process of working with other people and makes it easy to collaborate on projects.
Team members can work on files and easily merge their changes in with the master branch of the project.

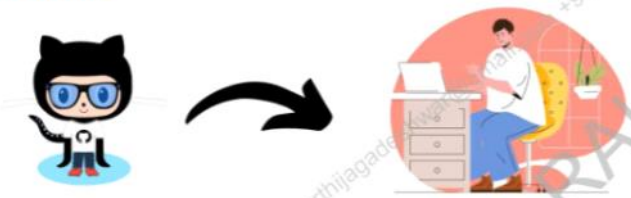
PUSH YOUR CODE TO GIT HUB:

- git remote add origin url

- git push -u origin branch-name

DevOps by RAHAM

GIT CLONING:



It means having same files in another folder.
To clone a git repo we need to have a repository and also check our present working directory.

- **git clone repo_url**

now we just cloned the files in repo-A to repo-B.
But before cloning we need to add and commit our files.

GIT MERGE:

- If you want to merge branch-1 with branch-2 switch to branch-1 first and give command git merge branch-2
- now that command had merged the content of branch-1 to branch-2.
- Whatever the content in branch-1 will be seen in branch-2 now.

git merge branchname

GIT FORK:

A fork is a rough copy of a repository. Forking a repository allows you to freely test and debug with changes without affecting the original project

GIT PULL:

push some from local to central and add some commits in central. you can see those central commits from local by using git fetch.
git fetch is used to get the updates from central repo.

git fetch : used to get the changes from central to local
git status : to see the differences of the commits
git log origin/branch_name : to see the logs of remote repo

to get those changes from central to local we use
git pull origin branch_name

DevOps by RAHAM

ADVANTAGES:

- Speed
- Simplicity
- Fully Distributed
- Excellent support for parallel development, support for hundreds of parallel branches.
- Integrity

DISADVANTAGES:

- Windows support issue.
- Entire download of the project history may be impractical and consume more disk space if the project has long history.

COMPARISION:

	HELIX TEAMHUB	GITLAB	GITHUB	BITBUCKET
Side-by-side view	✓	✗	✗	✗
Quick Action Buttons	✗	✓	✓	✓
Supports adding images	✓	✓	✓	✓
Supports adding other type of attachments	✓	✓	✗	✗
Search functionality for wiki pages	✓	✗	✗	✗
Support for multiple markup languages	✗	✓	✓	✓
Possibility to view the history on code level	✓	✗	✗	✓

DevOps by RAHAM

Session : 22

MAVEN

It is an Automation Project management tool developed by Apache software foundation.

It is based on POM (Project Object Model). (POM.xml) xml: extensible Markup Language.

- POM is nothing but Project Object Model
- xml is extensible markup language.
- POM.xlm files consists METADATA, DEPENDENCIES, KIND OF OUTPUT, KIND OF PROJECT, DESCRIPTION.

NOTE: One project contains only one workspace, each workspace consists of one POM.xml files.

MAVEN is a build tool and manage dependencies.

Manage Dependencies:

WHAT IS BUILD TOOL:

- it is used to set up everything which is required to run your java code This can be applied to your entire java project.
- It generates source code, compiling code, packaging code to a jar etc.
- POM refers the XML file that have all information regarding project and configuration details
- Main configuration file is in pom.xml.
- It has description of the Project details regarding version and configuration management.
- The XML file is in the Project home directory.

DevOps BY MUSTAFA

- Maven can build any number of projects into desired Output such as .jar, .war and .ear

  .jar = java archive file
  .war = web archive file
  .EAR = enterprise archive

- It is mostly used for java-based projects.
- It was initially released on 13 July 2004.
- Maven is written in java.

- Maven helps in getting the right jar file for each project as there may be different versions of separate packages.
- For downloading dependencies visit mvnrepository.com.

Dependencies: It refers to the java libraries that are needed for the project
Repositories: Refers to the directories of Packaged jar files.
Build tools:

C, C ++	:	make file
.Net	:	Visual studio
Java	:	Ant, Maven, Gradle

JAVA PROJECT STRUCTURE:

- SOURCE CODE
- TEST CODE
- PROJECT STRUCTURE (ASSERTS, DIRECTORIES, RESOURCES)
- DEPENDENCIES/LIBRARY
- CONFIGURATION
- TASK RUNNER
- REPORTING

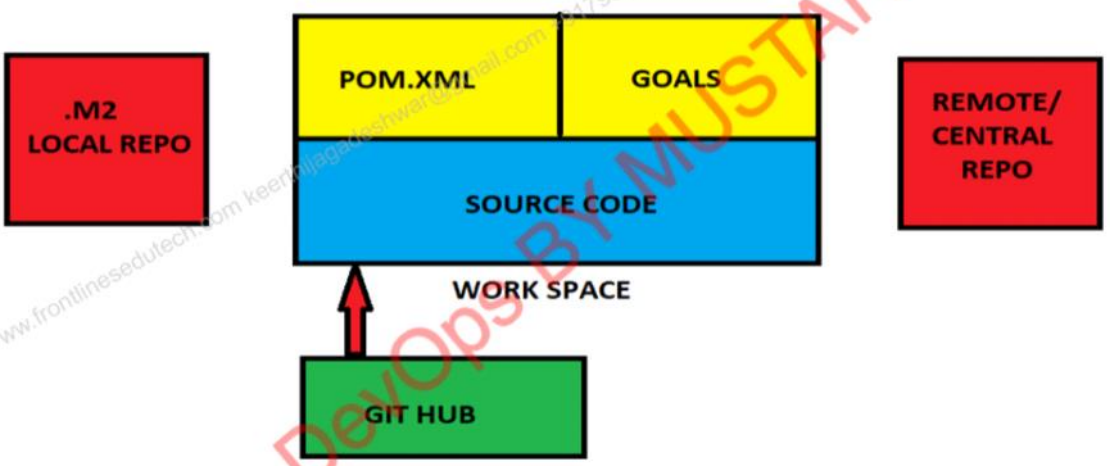


DevOps BY MUSTAFA

PROBLEMS WITHOUT MAVEN:

- **Adding set of jars in each project:** In the case of Struts, Springs we need to add jar files in each project. It must include all the dependencies of jars also.
- **Creating the right project structure:** We must create the right project structure in Servlet, Struts etc. Otherwise it will not be executed.
- **Building and Deploying the Project:** We must build and deploy the Project, so it may work.

MAVEN ARCHITECTURE:



COMPONENTS:

- **Local Repo:** Refers to the machine of the developer where all the Project materials are saved.
- **Remote Repo**: Refers to the repository present on a web server which is used when maven needs to download dependencies.
- **Central Repo**: Refers to the maven community that comes into the action when there is a need of dependencies and those dependencies cannot be found in the Local repository.
- **GOALS:** It is nothing but a task

DevOps BY MUSTAFA

HOW IT WORKS:

There are various components of Maven architecture – a local repository or the local machine that you work on (please refer to the diagram above). There is a central repository and then there is a remote repository or the remote web server.

So, whenever you specify any dependency in the POM.xml file, Maven will look for it in the central repository first. If the dependency is present in the central repository it will copy the same onto your local machine.

However, if that dependency is not present there, Maven will fetch it from the remote repository or remote web server using the internet. So, the internet is very much mandatory for using Maven.

MAVEN BUILD LIFE CYCLE:

DevOps BY MUSTAFA

1 Generate Resource = mvn archetype:generate

2. Compile Code = mvn compile

3. Unit Test = mvn test

4. Package (build) = mvn package

5. install (into Local repo or Artifactory) = mvn install

6. Deploy (to servers)

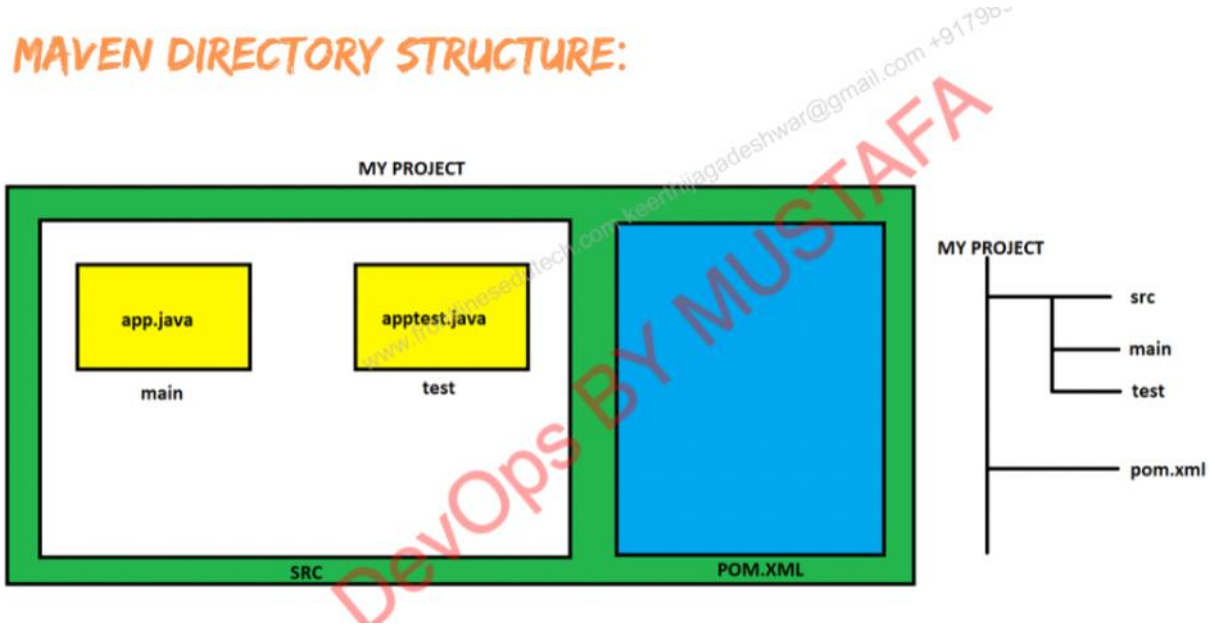
7. Clean (to delete all the runtime files) = mvn clean

- Build lifecycle consists of a sequence of build phases & each build phase consists sequence of goals
- Each goal is responsible for a particular task. Goals are nothing but commands.
- When a phase is run, all the goals related to that phase and its plugins are also compiled.

MAVEN VS ANT

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

DevOps BY MUSTAFA



MAVEN (7)

Session : 23

JENKINS CLASS-1 (J...

Session : 27

nexus (1)

NEXUS SETUP AN...

Session : 31

JENKINS DECALRA...

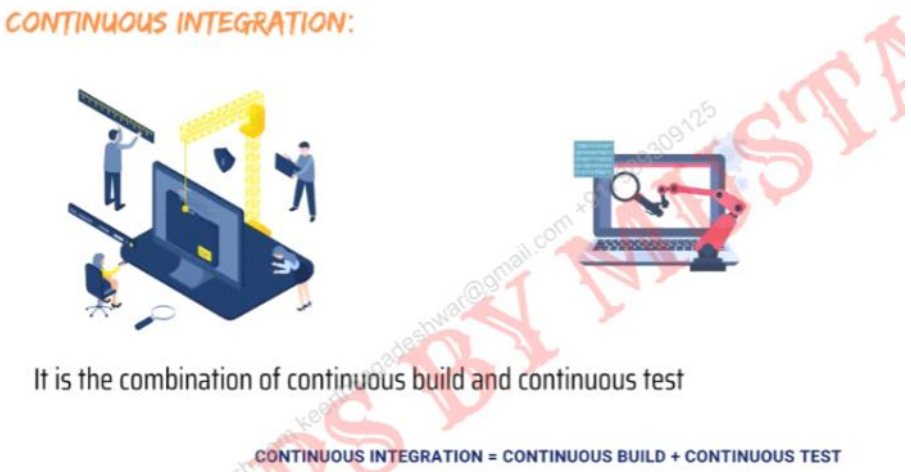
Session : 32

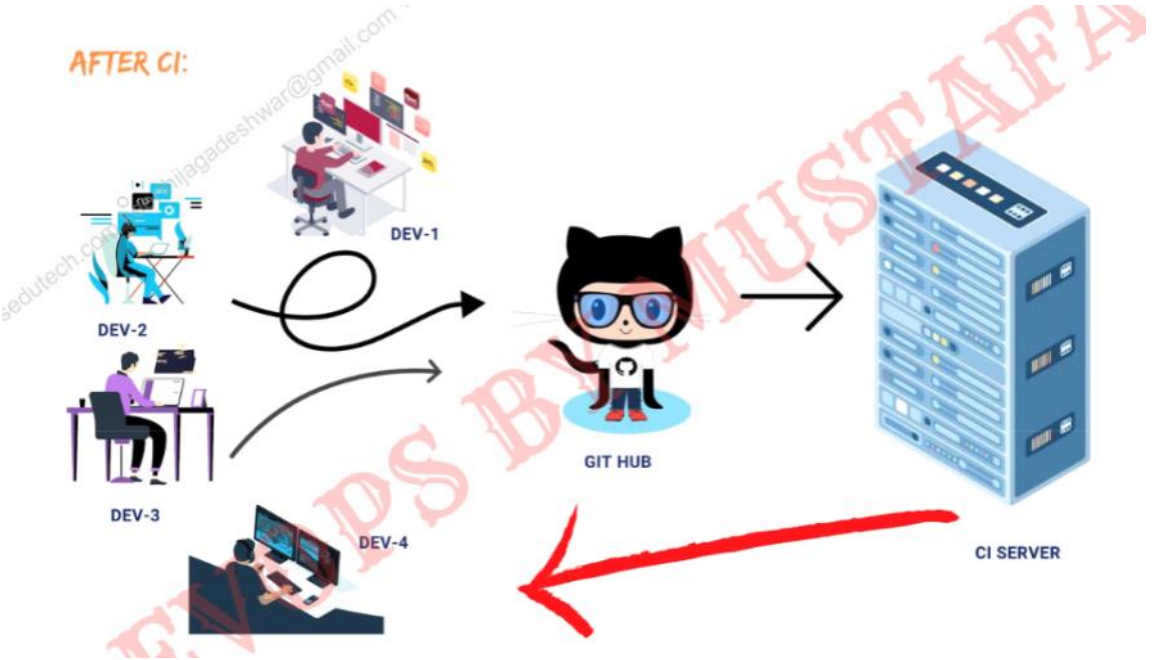
JENKINS DECALRA...

Session : 35

CICD IN JENKINS (1)

SONAR SETUP & I...





CI SERVER:

Here Build, Test & Deploy all these activities are performed in a single CI server

DEVOPS BY MUSTAFA

CONTINUOUS INTEGRATION:

Whenever a developer commits the code using source code management like GIT, then the CI pipeline gets the changes of the code runs automatically build and unit test.

- Due to integrating the new code with old code, we can easily get to know the code is a success or failure.
- It finds the errors more quickly
- Delivery the products to client more frequently
- Developers don't need to manual tasks
- Reduces the developers time 20% to 30%

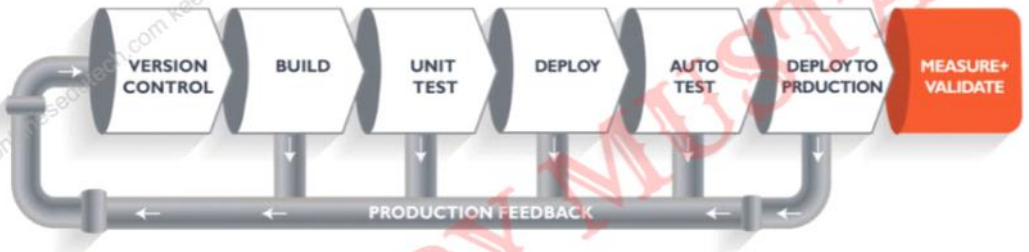
CD: CONTINUOUS DELIVERY/DEPLOYMENT

Continuous Delivery is making it available for deployment. Anytime a new build artifact is available, the artifact is automatically placed in the desired environment and deployed.

Continuous Deployment is when you commit your code then it gets automatically tested, build and deploy on the production server.

EX:

CI/CD PIPELINE:



It looks like Software Development Life Cycle, but let's see how it works.
Lets consider an example, if you are developing a web application

DEVOPS BY MUSTAFA

Version Control: Here developers need to write code for web applications. So it needs to be committed using version control system like GIT or SVN.

Build: Lets consider your code is written in java, it needs to be compiled before execution. In this build step code gets compiled.

Unit Test: If the build step is completed, then move to testing phase in this step unit step will be done.

Deploy: If the test step is completed, then move to Deploy phase in this step you can deploy your code in testing environment. Here you can see your application output.

Auto Test: Once our code is working fine in testing servers, then we need to do Automation testing using Selenium or Junit.

Deploy to Production: If everything is fine then you can directly deploy your code in production server.

NOTE: If we have error in Code then it will give feedback and it will be corrected, if we have error in Build then it will give feedback and it will be corrected, Pipeline will work like this until it reaches Deploy.

Because of this pipeline, Bugs will be reported fast and get rectified so entire development is fast.

JENKINS:

Jenkins is an **open source** project written in **java** that runs on the **Window, Linux and Mac OS**

It is community-supported, Free to use, and the First choice for Continuous Integration.

DEVOPS BY MUSTAF

- Consist of Plugins
- Automates the Entire Software Development Life Cycle (SDLC).

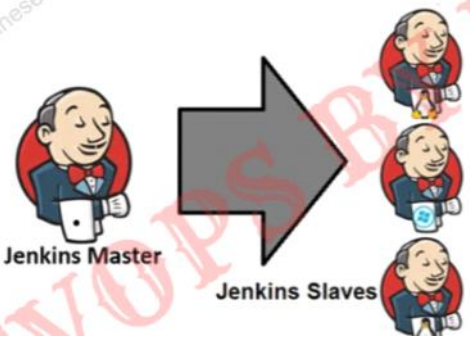
- It was originally developed by Sun Microsystem in 2004 as HUDSON.
- Hudson was an enterprise Edition we need to pay for it.
- The project was renamed Jenkins when Oracle brought the Microsystems.

- It can run on any major platform without Compatibility issues.

- Whenever developers write code, we integrate all the code of all developers at any point in time and we build, test, and deliver/deploy it to the client. This is called CI/CD.

ADVANTAGES:

- Jenkins follows Master-Slave Architecture.
- You can write your own plugin, can use the community plugin also.
- Can understand the process of what is going on.



DEVOPS BY MUSTAFA

- Jenkins master is going to assign a job to the slave..
- If Slaves are not available Jenkins itself does the job.
- By using the labels we can specify the jobs to the nodes.

JENKINS ALTERNATIVES:

Bamboo

circleci

GitLab

TC

Buddy

JENKINS SETUP:

- Go to **jenkins.io** and copy the links

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

- **Install EPEL** (Extra Package for Enterprise Linux)
1. sudo amazon-linux-extras install epel -y

- **Install java**
1. yum install java-1.8.0-openjdk -y

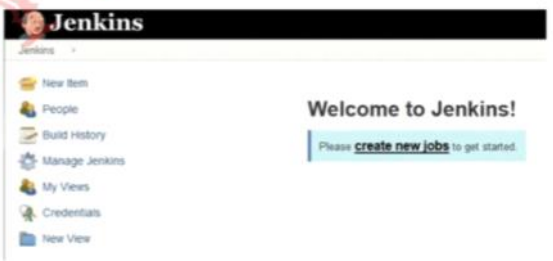
- **Install git, maven & Jenkins**
1. yum install git jenkins maven -y

- **Restart Jenkins**
1. systemctl restart/start jenkins

- Check Jenkins server is running or not:
1. systemctl status jenkins

- **Connect to dashboard:** copy the public IP address of the server and make a paste on the new tab with Jenkins port number (8080)
1. public_ip:8080

- **To unlock the jenkins:**
1. cat /var/lib/jenkins/secrets/initialAdminPassword
2. Install suggested plugins
3. Add user name, Password and mail id
4. connect to dashboard



DEVOPS BY MUSTAFA

JOB: To perform some set of tasks we use a job in jenkins.

PARAMETER TYPES:

- String: any combination of characters and numbers
- Choice: a pre-defined set of strings from which a user can pick a value
- Credentials: a pre-defined Jenkins credential
- File: the full path to a file on the filesystem
- Multi-line String: same as String, but allows newline characters
- Password: similar to the Credentials type, but allows us to pass a plain text param
- eter specific to the job or pipeline
- Run: an absolute URL to a single run of another job

FILE PARAMETER:

This is used when we want to build our local files.
General — > This Project is Parameterized — > Fille Parameter

CHOICE PARAMETER:

This parameter is used when we have multiple options to generate a build but need to use only on specific one.
General — > This Project is Parameterized — > Choice Parameter

STRING PARAMETER:

This parameter is used when we need to pass an parameter as input by default.
It can be any combination of characters and numbers.
General — > This Project is Parameterized — > String Parameter

MULTI STRING PARAMETER:

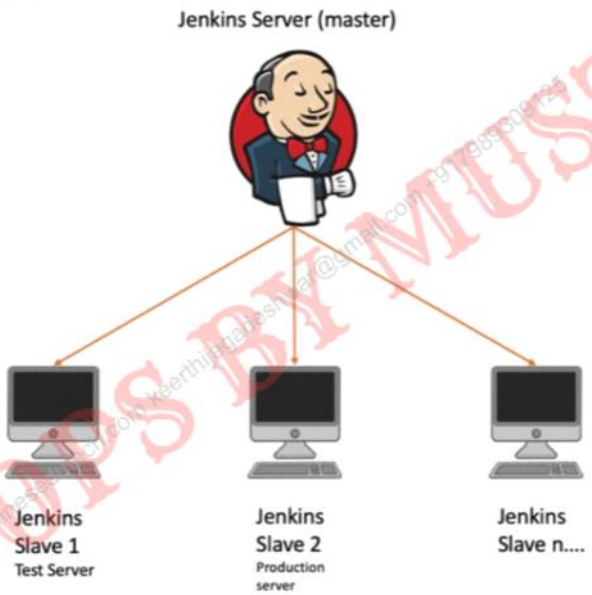
This will work as same as String Parameter but the difference is instead of one single line string we can use multiple strings at a time as a Parametres.
General — > This Project is Parameterized — > Multi-String Parameter

DEVOPS BY MUSTAFA

LINKED JOBS : This is used when a job is linked with another job

TYPES: Up stream and Down stream
Here for job-1 both job-2 & job-3 are Downstream
For job-2 upstream is job-1 and Downstream is job-3
for Job-3 Both Job-2 & job-1 are Upstream

MASTER & SLAVE:



SETUP:

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

sudo yum install java-1.8.0-openjdk git maven jenkins -y
sudo systemctl restart jenkins
sudo systemctl status jenkins
copy the IPV4 and paste it on browser like {ipv4:8080}
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

sudo vim /etc/passwd
sudo passwd jenkins
sudo visudo
sudo vim /etc/ssh/sshd_config
sudo systemctl restart sshd
sudo systemctl status sshd
```

DEVOPS BY MUSTA

LOGIN TO SLAVE

sudo useradd jenkins
sudo passwd jenkins
sudo visudo
sudo vim /etc/ssh/sshd_config
sudo systemctl restart sshd
sudo systemctl status sshd

LOGIN TO MASTER

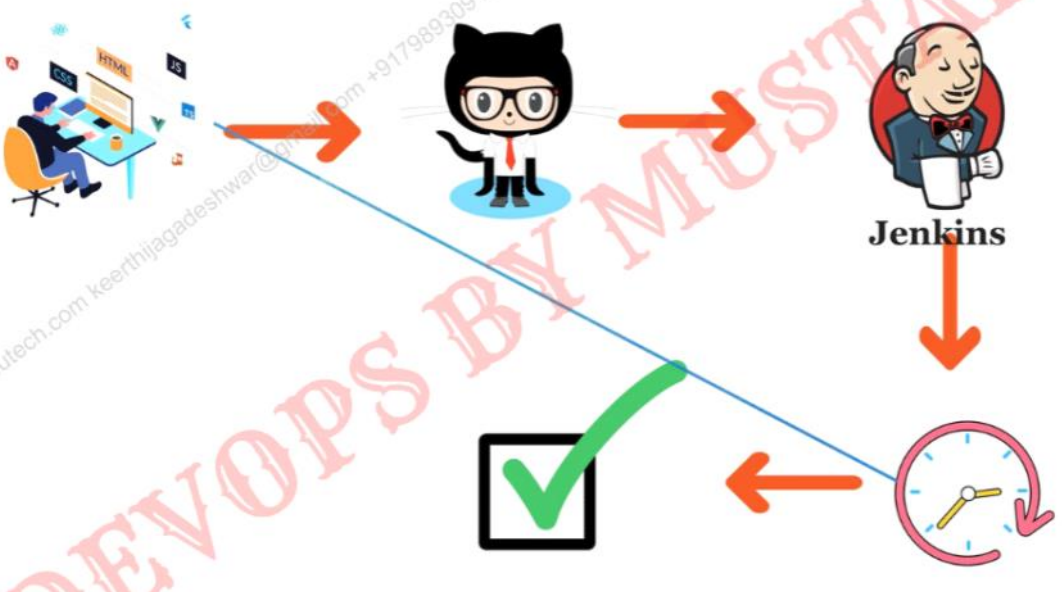
sudo su jenkins
ssh-keygen
ssh-copy-id jenkins@localhost
yes
exit
ssh-copy-id jenkins@public IPV4 of slave
ssh jenkins@public IPV4 of Slave

CRON JOB: We can schedule the jobs that need to be run at a particular intervals.

Build Triggers -- > Build periodically -- > * * * * * -- > save

If you want to make it customize then we can use cron syntax generator.

POLL SCM: This is used when we want to build after developer pushes the code for regular interval



DEVOPS BY MUSTAFA

WEBHOOKS: If developer changes the code then it will build at that point of time.

Go to the github and select the repo
select settings -- > webhooks -- > create a webhook
Payload url : jenkinsurl/github-webhook/
Content type: Application.json
Now change the code in repo you will get a new build.

ENVIRONMENT VARIABLES:

BUILD — > SHELL SCRIPT
name=raham
location=hyd
echo "HAI, MY NAME IS ${name} IAM FROM ${location}"

SELECT TO SEE LIST OF ALL ENVIRONMENT VARIABLES

echo "BUILD ID IS ${BUILD_ID}"
echo "JOB NAME IS ${JOB_NAME}"

echo "${GLOBAL_VAR}" : THIS WONT WORK WE NEED TO SET IT MANUALLY

MANAGE JENKINS — >CONFIGURE SYSTEM — > GLOBAL PROPERTIES — > ENVIRONMENT NAME — >
NAME: GLOBAL_VAR & VALUE: ABCD — > SAVE
NOW IT WILL PRINT THE GLOBAL VAR OUTPUT
APPLY THESE TO PARAMETERISED BUILDS

echo "${GLOBAL_VAR}"
echo "BUILD ID IS ${BUILD_ID}"
echo "JOB NAME IS ${SERVER}"
echo "DEPLOYED TO ${DEPLOY}"
echo "YOUR PASSWORD IS ${yourPasswordPara}"
echo "IM PRINTING ${multiLineStringParam}"
echo "THIS IS FILE ${FILEPATH}"

TIMEOUT:

BUILD ENVIRONMENT — > ABORT THE BUILD IF IT STUCK — > ABSOLUTE: 3 — > EXECUTE SHELL
SLEEP 240 (BUILD WILL TAKE 240 SECONDS) — > SAVE
AFTER 4 MINUTES IT WILL ABORT AUTOMATICALLY
IF YOU WANT TO MAKE IT AS FAIL CHECK TIME-OUT ACTIONS BELOW.

TIME STAMP:

echo "HAI ALL GOOD EVENING"
sleep 10
echo "WELCOME TO MY CLASS"
TO CHECK IT ENABLE TIMESTAMP BUILD ENVIRONMENT
CONCURRENT OR PARALLEL JOBS :

IF YOU WANT TO RUN PARALLEL BUILDS
GENERAL — > RUN PARALLEL BUILDS IF REQUIRED

THROTTLE BUILD:

IF YOU WANT TO RUN A SPECIFIC JOB FOR SPECIFIC TIMES IN A PERIOD OF TIME
NUMBER OF BUILDS: 4 TIME PERIOD: MINUTES
THAT MEANS ON A MINUTE MAXIMUM YOU CAN DO 4 BUILDS ONLY
YOU CAN SEE THE APPROXIMATE TIME BELOW THE NUMBER OF BUILDS

DEVOPS BY MUSTAFA

CUSTOM WORKSPACE:

THIS IS USED WHEN WE WANT TO CREATE OUR OWN CUSTOM WORK SPACE
GENERAL — > ADVANCED — > USE CUSTOM WORKSPACE — > PATH — > /tmp/test
NOW WE CREATED A TEST DIRECTORY UNDER /TMP
GIVE EXECUTE SHELL COMMAND (echo "HAI RAHAM" >> abc.txt)
NOW YOU WILL SEE abc.txt IN TEST DIRECTORY UNDER /tmp

RENAME A JOB:

IF YOU WANT TO RENAME A BUILD THEN
GENERAL — > ADVANCE — >DISPLAY NAME — > SAVE

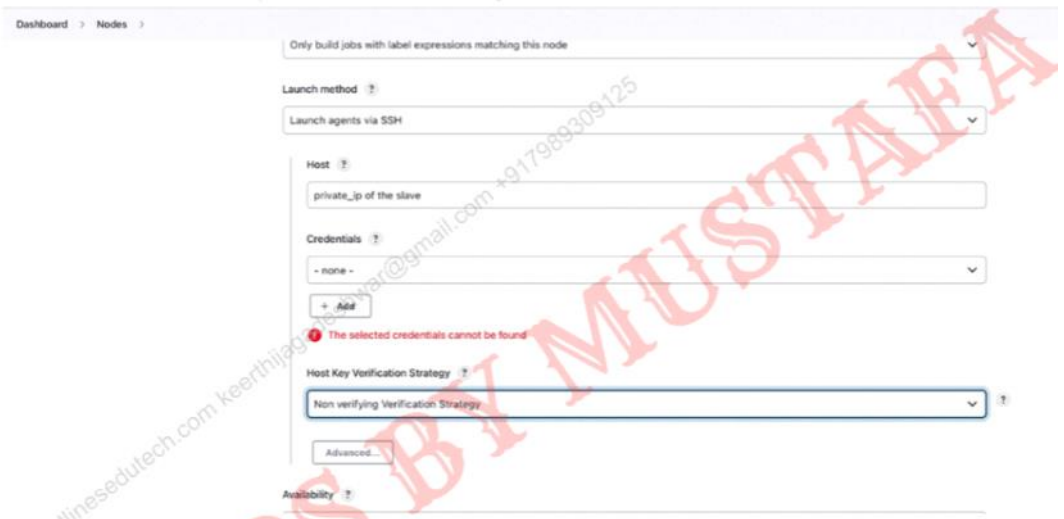
CREATE A PIPELINE THROUGH BUILD PIPELINE:

PLUGINS — > BUILD PIPELINE — > INSTALL
CREATE TWO JOBS AND CONFIGURE EXECUTE SHELL FOR BOTH JOBS
JOB1-BUILD: (sleep 10 & echo "THIS IS BUILDING")
JOB2-TEST: (sleep 10 & echo "THIS IS DEPLOYING TO TEST ENVIRONMENT")
JOB1-BUILD — > CONFIGURE — > POST BUILD ACTION — > BUILD OTHER PROJECT — > JOB2-TEST
CLICK ON + SYMBOL ON DASH BOARD — > BUILD PIPELINE VIEW — > NAME — > CREATE
SELECT INITIAL JOB (JOB-1)— > OK & APPLY

JENKINS AGENT SETUP:

LAUNCH 2 INSTANCES (MASTER & SLAVE) WITH PEM FILE.
JENKINS SETUP IN MASTER
INSTALL JAVA-OPENJDK11 IN SLAVE
GO TO JENKINS DASHBOARD--> MANAGE JENKINS --> SETUP AGENT --> give node name and select permanent agent and create.

DEVOPS BY MUSTAFA



Add jenkins credentials
Kind: SSH username with private key
Username: ec2-user
Private key: pem file copy paste
save & add node

PIPELINES:

Jenkins Pipeline is a combination of plugins that supports integration and implementation of continuous delivery pipelines.
A Pipeline is a group of events interlinked with each other in a sequence.

There are two types of Jenkins pipeline syntax used for defining your JenkinsFile
1. Declarative
2. Scripted

NEW ITEM — > NAME — > PIPELINE — > SAVE

DECLARATIVE

```
pipeline {
  agent any

  stages {
    stage('Hello') {
      steps {
        echo 'Hello World'
      }
    }
  }
}
```

SCRIPTED

```
node {
  stage ("stage1"){
    echo "hai"
  }
  stage ("stage1"){
    echo "hello"
  }
}
```

DEVOPS BY MUSTAFA

MULTISTAGE PIPELINES:

```
pipeline {
  agent any

  stages {
    stage('Hello') {
      steps {
        echo 'Hello World'
      }
    }
    stage('Test') {
      steps {
        echo 'Hello World test'
      }
    }
    stage('Deploy') {
      steps {
        echo 'Hello World deploy'
      }
    }
  }
}
```

PIPELINE AS A CODE:

YOU CAN RUN COMMANDS HERE

```
pipeline {
  agent any

  stages {
    stage('CMD') {
      steps {
        sh 'touch file1'
        sh 'pwd'
      }
    }
  }
}
```

MULTIPLE COMMANDS OVER SINGLE LINE:

```
pipeline {
  agent any

  stages {
    stage('CMD') {
      steps {
        sh ''
        touch file2
        pwd
        date
        whoami
        ''
      }
    }
  }
}
```

ENVIRONMENT VARIABLES:

```
pipeline {
  agent any

  stages {
    stage('ENV') {
      steps {
        sh 'echo "${BUILD_ID}"'
      }
    }
  }
}
```

```
pipeline {
  agent any
  environment {
    name = 'raham'
  }
  stages {
    stage('ENV') {
      steps {
        sh 'echo "${BUILD_ID}"'
        sh 'echo "${name}"'
      }
    }
  }
}
```

```
pipeline {
  agent any
  environment {
    name = 'raham'
  }
  stages {
    stage('ENV1') {
      steps {
        sh 'echo "${BUILD_ID}"'
        sh 'echo "${name}"'
      }
    }
    stage('ENV2') {
      environment {
        name = 'shaik'
      }
      steps {
        sh 'echo "${BUILD_ID}"'
        sh 'echo "${name}"'
      }
    }
  }
}
```

DEVOPS BY MUSTAFA
www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125

PIPELINE AS A CODE PARAMETERS:

STRING

```
pipeline {
  agent any
   environment {
       name = 'raham'
    }
    parameters {
      string(name: 'person', defaultValue: 'raham shaik', description: "how are you")
    }
  stages {
    stage('parameters') {
      steps {
        sh 'echo "${name}"'
        sh 'echo "${person}"'
      }
    }
  }
}
```

BOOLEAN

```
pipeline {
  agent any
   environment {
       name = 'raham'
    }
    parameters {
      booleanParam(name: 'male', defaultValue: true, description: "")
    }
  stages {
    stage('parameters') {
      steps {
        sh 'echo "${name}"'
        sh 'echo "${person}"'
      }
    }
  }
}
```

CHOICE

```
pipeline {
  agent any
   environment {
       name = 'raham'
    }
    parameters {
      choice(name: 'server', choices: ['a','b','c'], description: "")     }
  stages {
    stage('parameters') {
      steps {
        sh 'echo "${name}"'
        sh 'echo "${person}"'
      }
    }
  }
}
```

INPUT

```
stage('Continue ?') {
    input {
      message "Can We Continue ?"
      ok "Yes We Can Continue"
    }
      steps {
        echo 'HAI ALL'
      }
  }
```

www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125
OPS BY MUSTAFA

POST-BUILD:

THIS WILL BE PRINTED EVEN IF THE BUILD GETS FAILED

```
pipeline {
  agent any

  stages {
    stage('Hello') {
      steps {
        echo 'Hello World
      }
    }
  }
  post{
    always {
        echo 'THIS WILL BE PRINTED ANYWAY'
      }
    }
}
```

```
pipeline {
  agent any

  stages {
    stage('Hello') {
      steps {
        echo 'Hello World'
      }
    }
    stage ('Deploy') {
      steps {
        echo12 'This is deployed'
      }
    }
  }
  post{
    always {
        echo 'THIS WILL BE PRINTED ANYWAY'
      }
    }
}
```

```
pipeline {
  agent any

  stages {
    stage('Hello') {
      steps {
        echo 'Hello World'
      }
    }
    stage ('Deploy') {
      steps {
        echo 'This is deployed'
      }
    }
  }
  post{
    always {
        echo 'THIS WILL BE PRINTED ANYWAY'
      }
    failure {
      echo "THIS IS FAILED"
    }
    success {
      echo "THIS IS SUCCESS"
    }
  }
}
```

ALWAYS : WILL PRINT EVEN IF IT FAILS OR SUCCESS
FAILURE : WILL PRINT WHEN BUILD GOT FAILED
SUCCESS: WILL PRINT WHEN BUILD GOT SUCCESS

DO WITH THIS BY REMOVING 12 IN ECHO12

DEVOPS BY MUSTAFA

JENKINS CUSTOM WORK SPACE:

```
pipeline {
 agent {
  node {
   label 'my-defined-label'
   customWorkspace '/some/other/path'
  }
 }
 stages {
  stage ("one") {
   steps {
    echo "hai"
   }
  }
 }
}
```

PARALLEL STAGES ARE BUILD:

```
pipeline {
  agent any
  stages {
   stage('stage-1) {
    parallel {
     stage('stage-2') {
      steps {
       echo "this is stage-1"
      }
     }
     stage('stage-3') {
      steps {
       echo "this is stage-2"
      }
     }
    }
   }
  }
}
```

INPUT:

```
pipeline {
 agent any
 stages {
  stage ('build') {
 input{
  message "Press Ok to continue"
  submitter "user1,user2"
  parameters {
   string(name:'username', defaultValue: 'user',
description: 'Username of the user pressing
Ok')
  }
 }
 steps {
  echo "User: ${username} said Ok."
 }

  }
  }
}
```

TASK USING PARAMETERS:

```
pipeline {
   agent any
   parameters {
      string(name: 'NAME', description: 'Please
tell me your name')
      choice(name: 'GENDER', choices: ['Male',
'Female'], description: 'Choose Gender')
   }
   stages {
      stage('Printing name') {
         steps {
            script {
               def name = "${params.NAME}"
               def gender = "${params.GENDER}"
               if(gender == "Male") {
                  echo "Mr. $name"
               } else {
                  echo "Mrs. $name"
               }
            }
         }
      }
   }
}
```

DEVOPS BY MUSTAFA
www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125

BUILD TRIGGERS:

Create a job -- > job-1 -- > build triggers -- > Authentication Token : raham --> save
open new tab give like this [http://54.184.85.228:8080/job/job-1/build?token=raham]
Click enter your will see a new build
open incognito mode and execute then it will ask for login
Plugin -- > Build Authorization Token Root -- > install
http://54.184.85.228:8080/buildByToken/build?job=job-1&token=raham
(note if it doesnt work give like job=job-1\&

USER BASED ACCESS:

Manage jenkins -- > create users 2 or 3
Plugins -- > Role-Based Authorization Strategy -- > install
Manage Jenkins -- > Configure Global Security -- > Authorization -- > Role-Based Strategy
Manage and Assign Roles -- > Manage role -- > developer -- > add -- > permissions
Note: Give overall read access
item roles -- > roles to add: developer -- > Pattren: dev.*
item roles -- > roles to add: tester -- > Pattren: test.*
Assign roles -- > enter user1 give developer & enter user2 give tester role -- > save

GVOPS BY MUSTAFA

Session : 36

JENKINS-DECLARATIV...

ADVANCE TOPICS (2)

JENKINS INTERVIE...

Session : 39

ANSIBLE CLASS-1

Session : 41

ANSIBLE CLASS-2 (1)

Session : 42

ANSIBLE CLASS-4

ANSIBLE
CLASS-4

Session : 43

ANSIBLE:

- It is a Configuration Management Tool.
- Configuration: Ram, Storage, OS, Software and IP address of device.
- Management: Update, Delete, Add.
- Ansible is simple open-source IT engine which automates application deployment.
- Orchestration, Security and compliance.
- Uses YAML Scripting language which works on KEY-VALUE PAIR
- Ansible GUI is called as Ansible Tower. It was just Drag and Drop.
- Used PYTHON for Back end.

HISTORY:

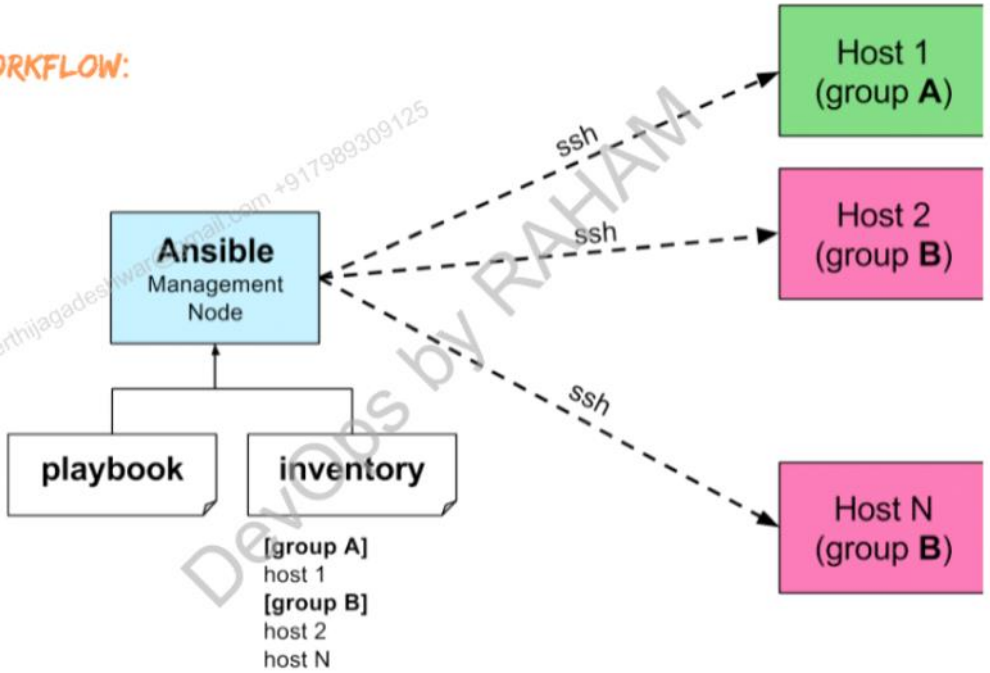
- Michael Dehhan developed Ansible and the Ansible project began in Feb 2012.
- Ansible was taken over by Red-hat.
- Ansible is Available for RHEL, Debian, CentOS, Oracle Linux.
- Can use this tool whether your servers are in On-prem or in the Cloud.
- It turns your code into Infrastructure i.e. Your computing environment has some of the same attributes as your application.

Ansible uses plain SSH. so nothing needs to install on client machines. but other automation tools like Chef/Puppet needs to install agent on client machine when we need to perform a task.

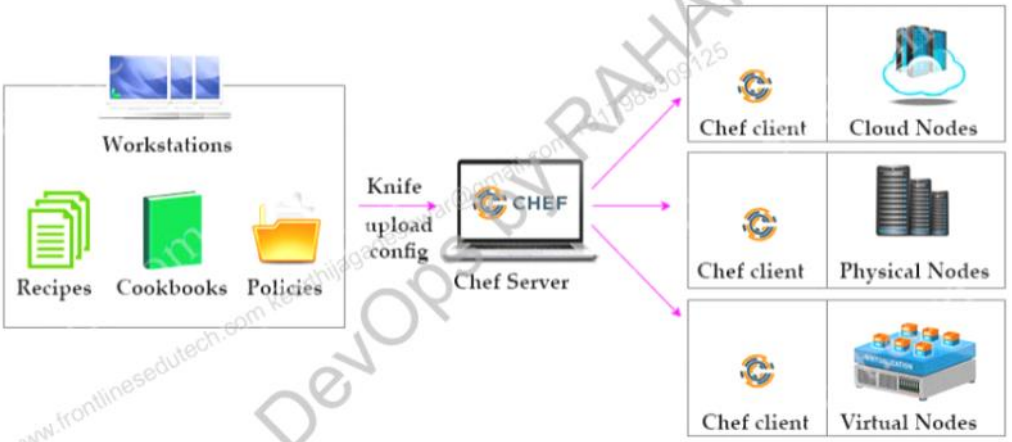
Ansible's is light weight, relative easy to use and speed of deployment compared to other tools. Ansible handle either via standard SSH commands, or the Paramiko module which provides a Python interface to SSH2.

DevOps by RAHAM

ANSIBLE WORKFLOW:



CHEF WORKFLOW:



DevOps by RAHAM

SETUP:

ANSIBLE SERVER:
sudo -i
sudo amazon-linux-extras install ansible2 -y
yum install git python python-pip python-level openssl -y
vi /etc/ansible/hosts
vi /etc/ansible/ansible.cfg
useradd ansible
passwd ansible
visudo
vim /etc/ssh/sshd_config

ALL NODES
useradd ansible
passwd ansible
visudo
vim /etc/ssh/sshd_config

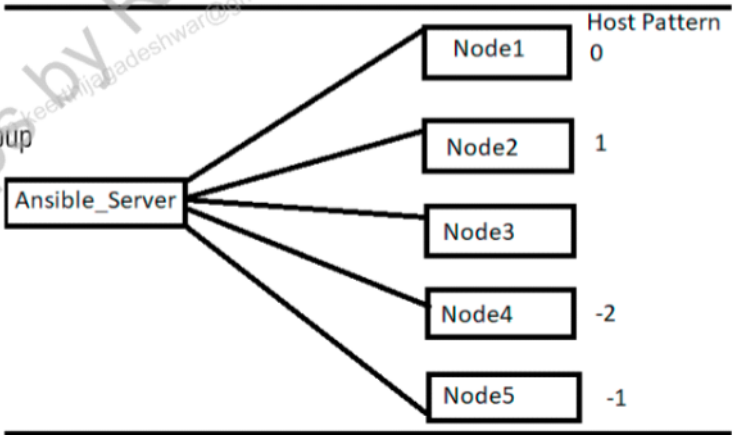
ALL SERVERS
sudo systemctl restart sshd
sudo systemctl status sshd
su - ansible

ANSIBLE SERVER:
ssh-copy-id ansible@localhost
yes & password
exit
ssh-copy-id ansible@privateip
yes & password
exit

www.frontlinesedutech.com
DevOps

HOST PATTREN:

'all' patterns refer to all the machines in an inventory.
ansible all-list-hosts ansible <groupname[remo]> --list-hosts
ansible <groupname> [remo][0] --list-hosts
groupname [0] – picks first machine of group
groupname [1] – picks second machine of group
groupname [-1] – picks last machine of group
groupname [1:4] – picks 2,3,4,5 machines in group
groupname [2:5] – picks 3 to 6 machines in the group
ansible all -m ping -v



If we want to push the code from Ansible server to nodes it can be done in 3 ways.
Ad-hoc Commands (Simple Linux) Ad-hoc means temporary & it will over-ride commands.
Modules – A Single Command.
Playbooks – More than one module is called Playbook.
Both module and Playbook is in YAML.

ADHOC COMMANDS:

These commands can be run individually to perform Quick functions.
Not used for configuration management and deployment, bcz the cmds are one time usage.
The ansible ad-hoc cmds uses /usr/bin/ansible/ command line tool to automate single task.
Go to ansible server and switch to ansible server

ansible remo -a "ls" [remo: Group name, -a: argument, ls: command]
ansible remo [0] -a "touch file1"
ansible all -a "touch file2"
ansible remo -a "sudo yum install httpd -y"
ansible remo -ba "yum install httpd -y" (b: become you will become sudo user)
ansible remo -ba "yum remove httpd -y"

DevOps by RAHAM

MODULES:

Ansible ships with number of modules (called library modules) that can be executed directly to remote hosts or playbooks.
Your library of modules can reside on any machine, and there are no servers, daemons or database required.
The default location for the inventory file is /etc/ansible/hosts
Go to ansible server and switch to ansible server

ansible remo -b -m yum -a "pkg=httpd state=present" (install: present)
ansible remo -b -m yum -a "pkg=httpd state=latest" (update: latest)
ansible remo -b -m yum -a "pkg=httpd state=absent" (uninstall: absent)
ansible remo -b -m service -a "name=httpd state=started" (started: start)
ansible remo -b -m user -a "name=raj" (to check go to that servers and sudo cat /etc/passwd).
ansible remo -b -m copy -a "src=filename dest=/tmp" (to check go to that server and give ls /tmp).

PLAYBOOKS:

Playbooks in ansible are written in YAML language.
It is human readable & serialization language commonly used for configuration files.
You can write codes consists of vars, tasks, handlers, files, templates and roles.
Each playbook is composed of one or more modules in a list.
Module is a collection of configuration files.
Playbooks are mainly divided into sections like

TARGET SECTION: Defines host against which playbooks task has to be executed.
VARIABLE SECTION: Defines variables.
TASK SECTION: List of all modules that we need to run in an order.

DevOps by RAHAM

YAML:

For ansible, nearly every YAML file starts with a list
Each item in the list is a list ok key-value pairs commonly called Dictionary.
All YAML files have to begin with "---" and end with "..."
A dictionary is required in a simple key: value form (note: space before value is must)
For example:

```
--- # Customer details
Customer:
   Name: abc
   Age : 26 y
   Salary: 100,000
   Exp : 4 year
```

Extension for playbook file is .yml

BASIC POINTS:

Go to ansible server and login as ansible and create one playbook
Vi target.yml

```
---# Target Playbook
hosts: remo  -- > remo: Groupname
user: ansible  -- > ansible: You are ansible user now
become: yes  -- > become: become sudo user -- > yes
connection: ssh
gather_facts: yes -- > Gives private IP of the nodes -- > yes
```

now save that file and execute the playbook by giving the command: ansible-playbook target.yml

Now create one more playbook in ansible server with cmd Vi task.yml

DevOps by RAHAM

NORMAL PLAYBOOK:

```
---
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: installing git
      action: yum name=git state=present
```

TO EXECUTE: ansible-playbook playbook.yml
DRY RUN: Check whether the playbook is formatted correctly or not.
ansible-playbook playbook.yml --check

VARIABLES:

Ansible also provides various ways of setting variables. They are used to store values that can be later used in the playbook.
Variable names in Ansible should start with a letter. The variable can have letter, numbers and underscore. Invalid variable declaration comes when we use dot (.), a hyphen (-), a number or variable separated by mutlple words.

```
--- #VARIABLES
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  vars:
    pkgname: httpd
  tasks:
    - name: installing httpd
      action: yum name='{{pkgname}}' state=present
```

DevOps by RAHAM

other way of passing arguments is by passing them to the command line while running using the –extra-vars parameter.

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: install git
      action: yum name='{{abc}}' state=present
~
```

for single var: ansible-playbook one.yml --extra-vars "abc=git"
for multiple vars: ansible-playbook one.yml --extra-vars "abc=git def=maven"

Passing a Varaible file - A Varaible can be defined in a variable file and can be passed to a playbook using the **include**

```
---
  - set_fact: abc=httpd
  - name: install Apache
    yum: name=httpd state=present
~
~
```

one.yml

two.yml

```
---
- hosts: dev
  become: yes
  tasks:
    - include: one.yml
    - name: install git
      service: name='{{abc}}' state=restarted
~
```

DevOps by RAHAM

HANDLERS:

Handler is same as task but it will run when called by another task. (OR)
It will run if the task contains a notify directive and also indicates that it changed something.

```
--- # HANDLER
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
        - name: install httpd server on centos
          action: yum name=httpd state=installed
          notify: restart httpd
  handlers:
          - name: restart httpd
            action: service name=httpd state=restarted
```

LOOPS:

Ansible loop includes changing ownership on several files & directories with file module, creating multiple users with user modules and repeating a polling step until result reached.

```
--- # LOOPS
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: add list of users in my nodes
            user: name='{{item}}' state=present
            with_items:
                    - raham
                    - mustafa
                    - shafi
                    - nazeer
```

DevOps by RAHAM

CONDITIONS:

If we have different scenarios, then we apply conditions according to the scenarios.

WHEN STATEMENT

Sometimes we want to skip a particular command on a particular node.

```
--- # CONDITIONS
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: Install apache server for debian family
            command: apt-get-y install apache2
            when: ansible_os_family== "Debian"
          - name: install apache server for redhat family
            command: yum install httpd -y
            when: ansible_os_family== "RedHat"
```

VAULT:

In ansible we can keep sensitive data like our passwords and keys in encrypted format.

ENCRYPTION TECHNIQUE: AES256 Used by Facebook.

ansible-vault create vault.yml : creating a new encrypted playbook.

ansible-vault edit vault.yml : Edit the encrypted playbook.

ansible-vault rekey vault.yml : To edit the password.

ansible-vault encrypt vault.yml : To encrypt the existing playbook.

ansible-vault decrypt vault.yml : To decrypt the encrypted playbook.

ROLES:

We can use two techniques for resulting a set of tasks they are Includes and Roles.

Roles are good for organizing tasks & encapsulating data needed to accomplish the task.

ANSIBLE ROLES: Default, Files, Handlers, Meta, Templates, Tasks, Vars.

We can organize playbooks into directory structure called Roles.

Adding more functionality to the playbooks will make it difficult to maintain in a single file.

mkdir -p playbook/roles/webserver/tasks --- > To see o/p use tree command.

Cd playbook & touch master.yml & touch roles/webserver/tasks/main.yml

vi roles/webserver/tasks/main.yml

```
- name: install apache on redhat
  yum: pkg=httpd state=latest
```

```
--- #MASTER PLAYBOOK
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  roles:
          - webserver
```

DevOps by RAHAM

TAGS:

If you have a large playbook, it may be useful to run only specific parts of it instead of running the entire playbook. You can do this with Ansible tags. Using tags to execute or skip selected tasks

```
---
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: installing git
      action: yum name=git state=present
      tags: install
    - name: uninstalling git
      action: yum name=git state=absent
      tags: uninstall
```

TO EXECUTE A SINGLE TASK: ansible-playbook abc.yml --tags tagname
TO EXECUTE A MULTIPLE TASK: ansible-playbook abc.yml --tags tagname1,tagname2
TO SKIP A TASK: ansible-playbook abc.yml --skip-tags "uninstall"

GALAXY:

Ansible Galaxy is a galaxy website where users can share roles and to a command-line tool for installing, creating, and managing roles.
Ansible Galaxy gives greater visibility to one of Ansible's most exciting features, such as application installation or reusable roles for server configuration. Lots of people share roles in the Ansible Galaxy.
Ansible roles consist of many playbooks, which is a way to group multiple tasks into one container to do the automation in a very effective manner with clean, directory structures.

ansible-galaxy init raham
ansible-galaxy search elasticsearch
ansible-galaxy search elasticsearch --author alikins
ansible-galaxy install alikns.elasticsearch
cd /home/ansible/.ansible/roles/
USER INFO:
Go to the ansible galaxy website and select a username
ansible-galaxy info bonddim.linux
ansible-galaxy collection install bonddim.linux

DevOps by RAHAM

ADVANTAGES:

Very simple to set up and use.
No special coding skills are necessary to use Ansible's playbooks.
Ansible lets you model even highly complex IT workflows.
You can orchestrate the entire application environment no matter where it's deployed.

DIS ADVANTAGES:

Ansible does not have any notion of state like other automation tools such as Puppet
Ansible does not track dependencies and simply executes sequential tasks and stops when tasks finish, fail, or any error comes.
Ansible has external dependencies to Python modules
Windows interaction requires some scheming

JENKINS SETUP USING PLAYBOOK:

```
---
- hosts: dev[0]
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: getting links from jenkins.io
      get_url:
        url: https://pkg.jenkins.io/redhat-stable/jenkins.repo
        dest: /etc/yum.repos.d/jenkins.repo

    - name: getting 2nd link from jenkins.io
      ansible.builtin.rpm_key:
        state: present
        key: https://pkg.jenkins.io/redhat-stable/jenkins.io.key

    - name: update links
      action: yum name="*" state=latest

    - name: install java
      command: sudo amazon-linux-extras install java-openjdk11 -y

    - name: install jenkins
      action: yum name=jenkins state=present


     - name: import systemd
       ansible.builtin.systemd:
         daemon_reload: yes

     - name: restart jenkins
       ansible.builtin.systemd:
         name: jenkins
         state: restarted
```

DevOps by RAHAM

PLAYBOOK TO CREATE A FILE/FOLDER:

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: creating a file
      file:
        path: "jenkins.txt"
        state: touch
~
```

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: creating a file
      file:
        path: "folder"
        state: directory
~
```

TO ENTER A DATA IN A FILE:

```yaml
---
- hosts: dev
  tasks:
    - name: inserting a data in a file
      copy:
        dest: "devops.txt"
        content: |
          hi this is devops file
          we are inserting the data ij a file
          using ansible playbook
~
```

PLAYBOOK TO CHANGE PERMISSIONS TO A FILE:

```yaml
---
- hosts: dev
  tasks:
    - name: change permissions to a file
      file:
        path: "devops.txt"
        state: touch
        mode: 777
```

DevOps by RAHAM

PLAYBOOK TO DEPLOY A WEBAPP:

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: install httpd
      action: yum name=httpd state=present

    - name: restart httpd
      service: name=httpd state=restarted

    - name: create a file
      file:
        path: "/var/www/html/index.html"
        state: touch

    - name: enter data in a file
      copy:
        dest: "/var/www/html/index.html"
        content: |
          <h1>this is my webapplication, i have deployed using ansible </h1>
~
```

PIP MODULE:

Ansible pip module is used when you need to manage python libraries on the remote servers. There are two prerequisites if you need to use all the features in the pip module.

- The pip package should already be installed on the remote server.
- Virtualenv package should be installed on the remote server already if you need to manage the packages in the python virtual environment.

```
- hosts: all
  tasks:
  - name: Installing NumPy python library using pip module
    pip:
      name: NumPy
```

RAW MODULE:

RAW module is used when there is more need than Command module or if the command module does not support the operation. This module makes a SSH to the remote machine and run the command. For the Ansible to work we need to have Python available but for this module we don't need a Python to be available

```
---
- hosts: dev
  become: yes
  tasks:
   - name: Install VIM
     raw: yum -y install vim-common
~
~
```

DevOps by RAHAM

PLAYBOOK TO GET CODE FROM GIT (PUBLIC REPO):

```
---
- hosts: localhost
  become: yes
  tasks:
    - name: getting code from git
      git:
        repo: "https://github.com/devops0014/pubg.git"
        dest: "/home/mycode"
~
~
```

PLAYBOOK TO GET CODE FROM GIT (PRIVATE REPO):

```
---
- hosts: localhost
  become: yes

  tasks:
    - name: link
      git:
        repo: 'https://ghp_6IplSHNjPFSkW3wBz02jHipPUozmmO4doQ0@github.com/devops0014/ansible.git'
        dest: "/home/mygitcode"
~
~
```

SYNTAX: TOKEN@GITHUB.COM/USERNAME/REPO.GIT

ANSIBLE SETUP MODULES:

ansible_os_family

os name like RedHat, Debian, Ubuntu etc..

ansible_processor_cores

No of CPU cores

ansible_kernel

Based on the kernel version

ansible_devices

connected devices information

ansible_default_ipv4

IP Mac address, Gateway

ansible_architecture

64 Bit or 32 Bit

DevOps by RAHAM

After executing a playbook, if you want to see the output in json format
ansible -m setup private_ip

if you want to apply a see particular output, you can apply filter.
ansible -m setup -a "filter=ansible_os_family" private_ip
ansible -m setup -a "filter=ansible_devices" private_ip
ansible -m setup -a "filter=ansible_kernel" private_ip

ANSIBLE DEBUG:

it can fix errors during execution instead of editing your playbook.

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - debug:
        msg: "os family for {{ansible_fqdn}} is {{ansible_os_family}}"
```

You can see that the task is performing on which OS.

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - debug:
        msg: "ansible_memory_mb  memory is  {{ansible_memory_mb.real}}"

    - debug:
        msg: "ansible_memory_mb  free memory is  {{ansible_memory_mb.real.free}}"

    - debug:
        msg: "ansible_memory_mb used memory is  {{ansible_memory_mb.real.used}}"
~
~
~
```

Depends upon the memory, we can debug.

Depends upon the ip, we can debug.

```
- debug:
      msg: "ip info of all the devices is  {{ansible_all_ipv4_addresses}}"
```

If you run a command to check the files along with its content in a server, it will not shows us the output. But we can debug the output.

```
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: get users
      command: cat /etc/passwd
      register: output

    - debug:
        msg: "users list in the ansible is {{output.stdout}}"
~
```

DevOps by RAHAM

ANSIBLE IN-OUT OPERATOR:

To check weather GIT is installed or not.

```
---
- hosts: dev
  become: yes
  tasks:
    - name: Check for the GIT
      shell: rpm -qa  | grep git
      register: rpm_status
      ignore_errors: yes

    - name: check if the GIT is installed or not
      debug: msg="git is installed on the client server"
      when: " 'git' in rpm_status.stdout"

    - name: Check if GIT is installed
      debug: msg="GIT is not installed on the client server"
      when: not 'git' in rpm_status.stdout
```

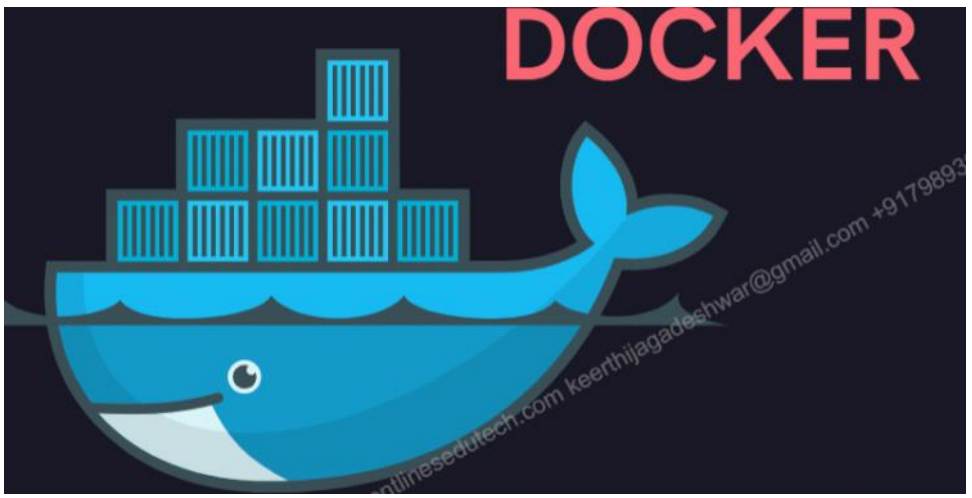
Demos by RAHAM
www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309

ANSIBLE INTERVIE...

Session : 53

compose installation

Session : 54

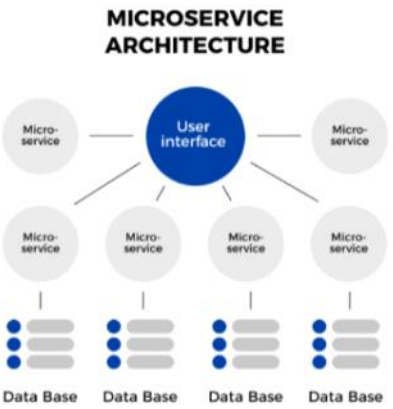


MONOLITHIC

If an application contains N number of services (Let's take Paytm has Money Transactions, Movie Tickets, Train tickets, etc..) If all these services are included in one server then it will be called Monolithic Architecture. Every monolithic Architecture has only one database for all the services.

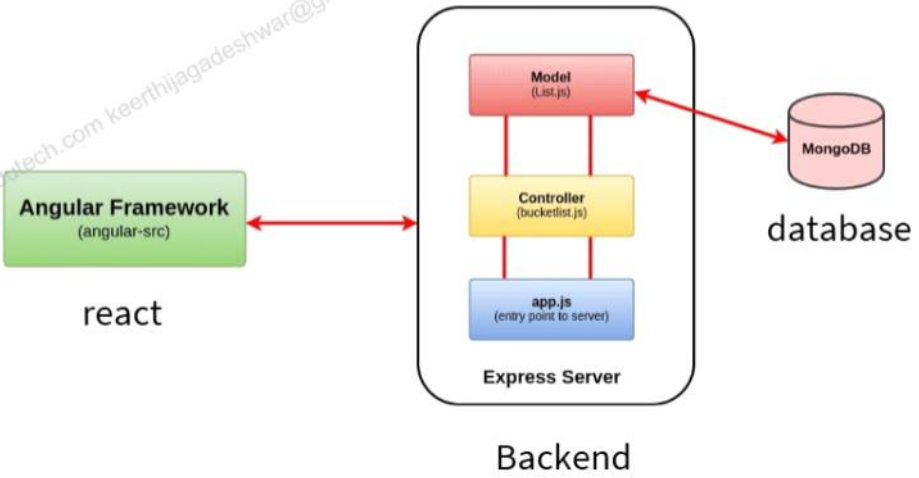
MICRO SERVICE

If an application contains N number of services (Let's take Paytm has Money Transactions, Movie Tickets, Train tickets, etc..) if every service has its own individual servers then it is called microservices. Every microservice architecture has its own database for each service.



WHY DOCKER

let us assume that we are developing an application, and every application has front end, backend and Database.



To develop the application we need install those dependencies to run to the code.

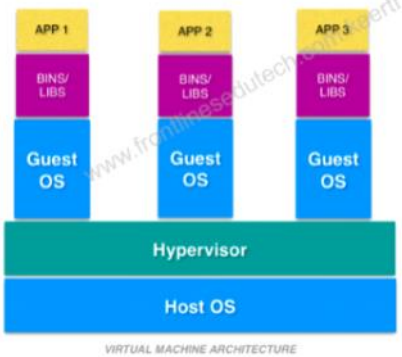
So i installed Java11, ReactJS and MongoDB to run the code.
After some time, i need another versions of java, react and mongo DB for my application to run the code.

So its really a hectic situation to maintain multiple versions of same tool in our system.

To overcome this problem we will use virtualization.

VIRTUALIZATION:

It is used to create a virtual machines inside on our machine. in that virtual machines we can hots guest OS in our machine.
by using this Guest OS we can run multiple application on same machine.
Hypervisor is used to create the virtualization.



VIRTUAL MACHINE ARCHITECTURE

DRAWBACKS:

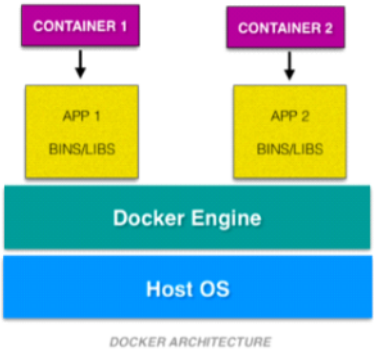
- It is old method.
- If we use multiple guest OS then the performance of the system is low.

CONTAINERIZATION:

- It is used to pack the application along with its dependencies to run the application.

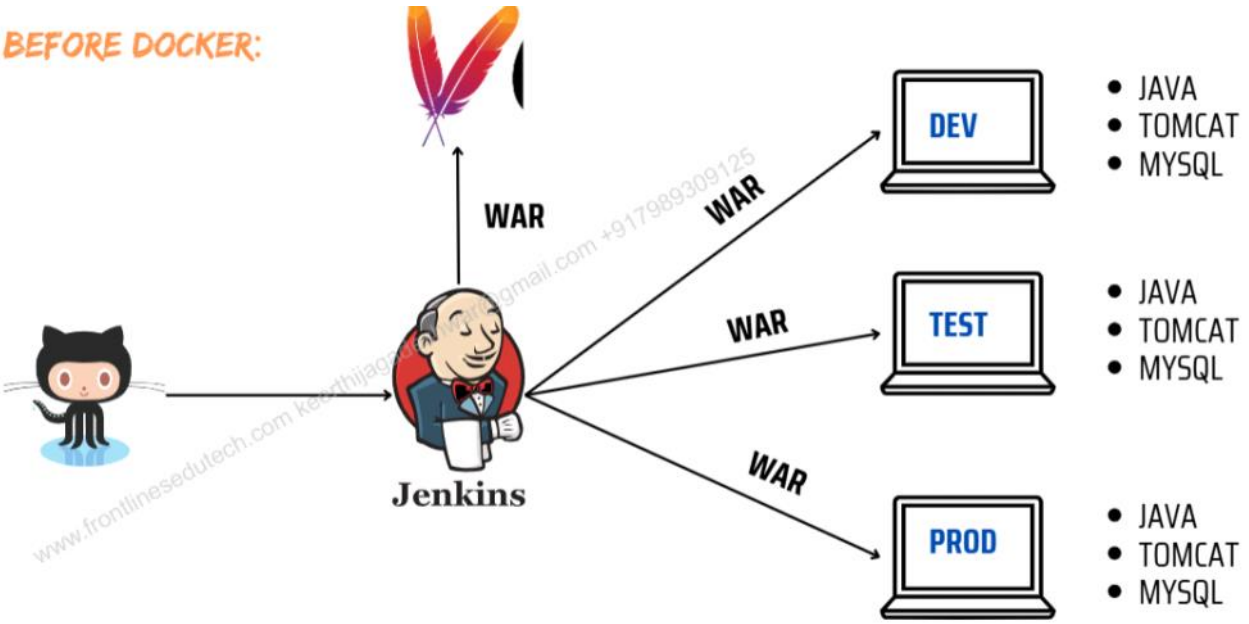
CONTAINER:

- Container is nothing but, it is a virtual machine which does not have any OS.
- Docker is used to create these containers.



DOCKER ARCHITECTURE

BEFORE DOCKER:



www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125

AFTER DOCKER:

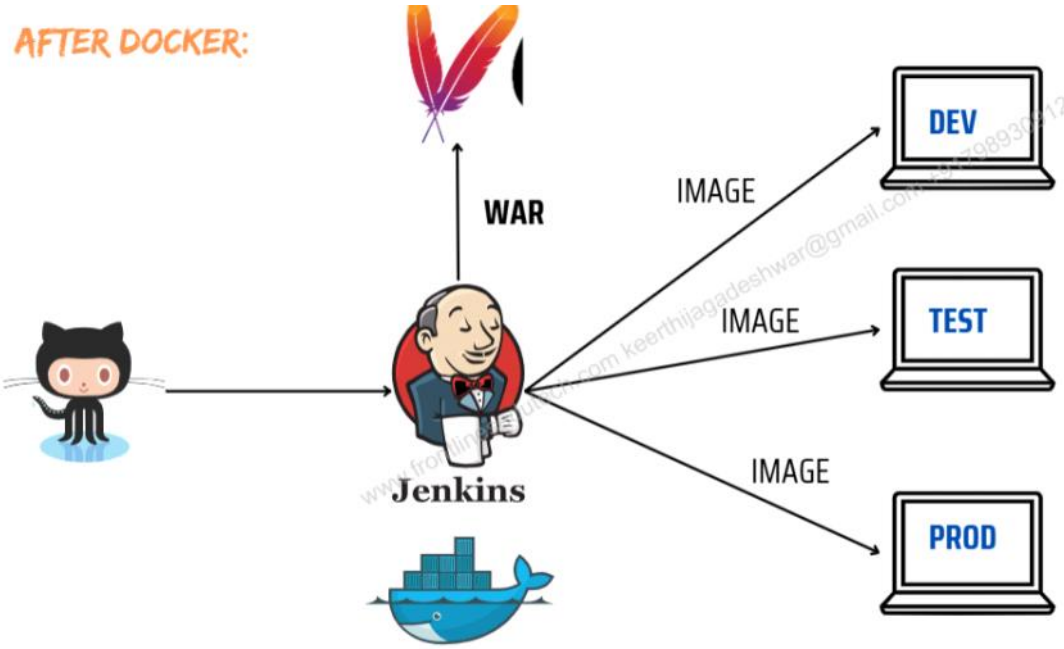
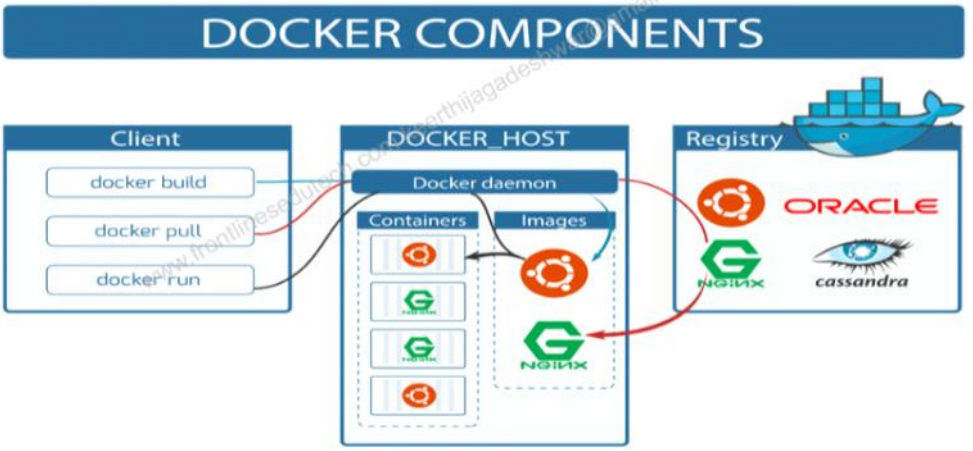


IMAGE = WAR + JAVA + TOMCAT + MYSQL

DOCKER

- It is an open source centralized platform designed to create, deploy and run applications.
- Docker is written in the Go language.
- Docker uses containers on host O.S to run applications. It allows applications to use the same Linux kernel as a system on the host computer, rather than creating a whole virtual O.S.
- We can install Docker on any O.S but the docker engine runs natively on Linux distribution.
- Docker performs O.S level Virtualization also known as Containerization.
- Before Docker many users face problems that a particular code is running in the developer's system but not in the user system.
- It was initially released in March 2013, and developed by Solomon Hykes and Sebastian Pahl.
- Docker is a set of platform-as-a-service that use O.S level Virtualization, where as VM ware uses Hardware level Virtualization.
- Container have O.S files but its negligible in size compared to original files of that O.S.

ARCHITECTURE:



DOCKER CLIENT:

It is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to docker daemon, which carries them out. The docker command uses the Docker API.

DOCKER HOST:

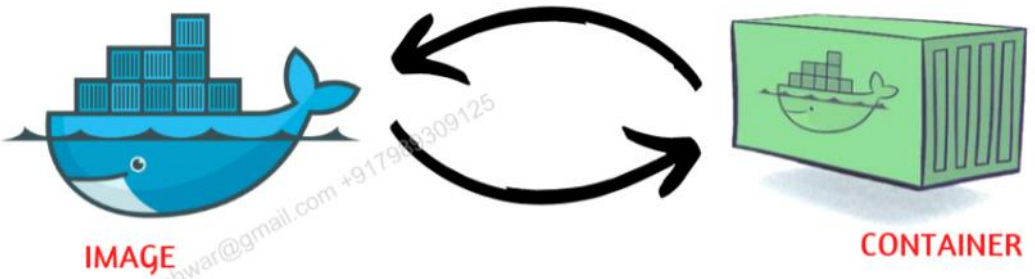
Docker host is the machine where you installed the docker engine.

DOCKER DAEMON:

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage.

DOCKER REGISTRY:

A Docker registry is a scalable open-source storage and distribution system for docker images.



POINTS TO BE FOLLOWED:

- You cant use docker directly, you need to start/restart first (observe the docker version before and after restart)
- You need a base image for creating a Container.
- You cant enter directly to Container, you need to start first.
- If you run an image, By default one container will create.

BASIC DOCKER COMMANDS:

To install docker in Linux : yum install docker -y
To see the docker version : docker --version
To start the docker service : service docker start
To check service is start or not : service docker status
To check the docker information : docker info
To see all images in local machine : docker images
To find images in docker hub : docker search image name
To download image from docker hub to local : docker pull image name
To download and run image at a time : docker run -it image name /bin/bash
To give names of a container : docker run -it --name raham img-name /bin/bash
To start container : docker start container name
To go inside the container : docker attach container name
To see all the details inside container : cat /etc/os-release
To get outside of the container : exit
To see all containers : docker ps -a
To see only running containers : docker ps (ps: process status)
To see only exited containers: docker ps -q -f "state=exited"
To stop the container : docker stop container name
To delete container : docker rm container name
To stop all the containers : docker stop $(docker ps -a -q)
To delete all the stopped containers : docker rm $(docker ps -a -q)
To delete all images : docker rmi -f $(docker images -q)

DOCKER RENAME:

To rename docker container: Docker rename old_container new_container
To rename docker port:
stop the container
go to path (var/lib/docker/container/container_id)
open hostconfig.json
edit port number
restart docker and start container

DOCKER EXPORT:

It is used to save the docker container to a tar file
Create a file which contains will gets stored: touch docker/password/secrets/file1.txt
TO EXPORT: docker export -o docker/password/secrets/file1.txt container_name
SYNTAX: docker export -o path container

BASIC DOCKER COMMANDS:

To see list of containers : docker container ls
To see all running containers: docker container ls -a
To see latest 2 containers : docker container ls -n 2
To see latest container : docker container ls --latest
To see all container id's : docker ls -a -q
To remove all containers : docker container rm -f $(docker container ls -aq)
To see containers with sizes : docker container ls -a -s
To stop container after some time: docker stop -t 60 cont_id

KILL VS STOP:

KILL: It passes SIGKILL signal to the container and container must be in running state.
STOP: It passes SIGTERM signal to the container and container m

RUNNING A CONTAINER:

- docker run --name cont1 -d nginx
- docker inspect cont1
- curl container_private_ip:80
- docker run --name cont2 -d -p 8081(hostport):80(container port) nginx

docker exec:

- syntax - docker exec cont_name command
- ex-1: docker exec cont1 ls
- ex-2: docker exec cont mkdir devops
- to enter into container: docker exec -it cont_name /bin/bash **or** docker exec -it cont_name bash

LIMITS TO CONTAINER

It is used to set a memory limits to containers

docker run -dit --name cont_name --memory=250m --cpu="0.25" image_name

to check: docker inspect cont_name | grep -i memory

to check: docker inspect cont_name | grep -i nanocpu

CREATE IMAGE FROM CONTAINER:

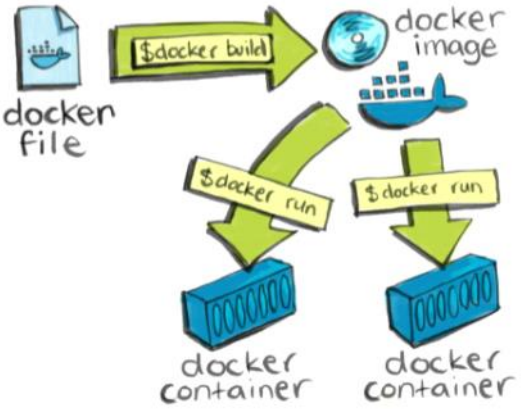
- First it should have a base image - docker run nginx
- Now create a container from that image - docker run -it --name container_name image_name /bin/bash
- Now start and attach the container
  - go to tmp folder and create some files (if you want to see the what changes has made in that image - docker diff container_name)
- exit from the container
- now create a new image from the container - docker commit container_name new_image_name
- Now see the images list - docker images
- Now create a container using the new image
- start and attach that new container
- see the files in tmp folder that you created in first container.

DOCKER FILE:

- It is basically a text file which contains some set of instructions.
- Automation of Docker image creation.
- Always D is capital letters on Docker file.
- And Start Components also be Capital letter.

HOW IT WORKS:

- First you need to create a Docker file
- Build it
- Create a container using the image



DOCKER FILE COMPONENTS:

FROM: For base image this command must be on top of the file. Ex: ubuntu, Redis, Jenkins
LABEL: Labeling like EMAIL, AUTHOR, etc.
RUN: To execute commands during image creation.
COPY: Copy files/folders from local system (docker VM) where need to provide Source and Destination.
ADD: It can download files from the internet and also, we can extract files at docker image side.
EXPOSE: To expose ports such as 8080 for tomcat and port 80 nginx etc.
WORKDIR: To set working directory for the Container.
CMD: Executes commands but during Container creation.
ENTRYPOINT: The command that executes inside of a container. like running the services in a container.
ENV: Environment Variables.

ARG argument is not available inside the Docker containers and
ENV argument is accessible inside the container

RUN: it is used to execute the commands while we build the images and add a new layer into the image.
CMD: it is used to execute the commands when we run the container.
if we have multiple CMD's only last one will gets executed.
ENTRYPOINT: it overwrites the CMD when you pass additional parameters while running the container.

COPY: Used to copy local files to containers
ADD: Used to copy files form internet and extract them

STOP: attempts to gracefully shutdown container, issues a SIGTERM signal to the main process.
KILL: immediately stops/terminates them, while docker kill (by default) issues a SIGKILL signal.

DOCKER FILE TO CREATE AN IMAGE:

FROM: ubuntu
RUN: touch aws devops linux

FROM: ubuntu
RUN: touch aws devops linux
RUN echo "hello world">/tmp/file1

TO BUILD: docker build -t image_name . (. represents current directory)

Now see the image and create a new container using this image.
go to container and see the files that you created.

```
FROM ubuntu
WORKDIR /tmp
RUN echo "hello world!" > /tmp/testfile
ENV myname raham
COPY testfile1 /tmp
ADD test.tar.gz /tmp
```

```
[root@ip-172-31-83-27 ~]# touch testfile1
[root@ip-172-31-83-27 ~]# touch test
[root@ip-172-31-83-27 ~]# ls
Dockerfile  test  testfile1
[root@ip-172-31-83-27 ~]# tar -cvf test.tar test
test
[root@ip-172-31-83-27 ~]# ls
Dockerfile  test  testfile1  test.tar
[root@ip-172-31-83-27 ~]# gzip test.tar
[root@ip-172-31-83-27 ~]# ls
Dockerfile  test  testfile1  test.tar.gz
[root@ip-172-31-83-27 ~]# rm -rf test
[root@ip-172-31-83-27 ~]# ls
Dockerfile  testfile1  test.tar.gz
[root@ip-172-31-83-27 ~]# docker build -t raham .
```

docker run -it - -name container name image-name /bin/bash

```
FROM ubuntu:14.04
MAINTAINER abc "abc@abc.com"
RUN apt-get update
RUN apt-get install -y nginx
RUN echo 'Our first Docker image for Nginx' > /usr/share/nginx/html/index.html
EXPOSE 80
~
```

TO BUILD:

docker build -t image1 .

TO RUN:

docker run -dit --name mustafa -p 8081:80 image1 nginx -g "daemon off;"

DOCKER VOLUMES:

- When we create a Container then Volume will be created.
- Volume is simply a directory inside our container.
- First, we have to declare the directory Volume and then share Volume.
- Even if we stop/delete the container still, we can access the volume.
- You can declare directory as a volume only while creating container.
- We can't create volume from existing container.
- You can share one volume across many number of Containers.
- Volume will not be included when you update an image.
- If Container-1 volume is shared to Container-2 the changes made by Container-2 will be also available in the Container-1.

- You can map Volume in two ways:
  1. Container < ------ > Container
  2. Host < ------- > Container

USES OF VOLUMES:

- Decoupling Container from storage.
- Share Volume among different Containers.
- Attach Volume to Containers.
- On deleting Container Volume will not be deleted.

CREATING VOLUMES FROM DOCKER FILE:

- Create a Docker file and write
  FROM ubuntu
  VOLUME["/myvolume"]

```
FROM ubuntu
ADD file1 /ubuntu1/file
VOLUME /ubuntu1
~
```

- build it - docker build -t image_name .
- Run it - docker run -it - -name container1 ubuntu /bin/bash
- Now do ls and you will see myvolume-1 add some files there
- Now share volume with another Container - docker run -it - -name container2(new) - -privileged=true - -volumes-from container1 ubuntu
- Now after creating container2, my volume1 is visible
- Whatever you do in volume1 in container1 can see in another container
- touch /myvolume1/samplefile1 and exit from container2.
- docker start container1
- docker attach container1
- ls/volume1 and you will see your samplefile1

CREATING VOLUMES FROM COMMAND:

- docker run -it - -name container3 -v /volume2 ubuntu /bin/bash
- now do ls and cd volume2.
- Now create one file and exit.
- Now create one more container, and share Volume2 - docker run-it - -name container4 - - -privileged=true - -volumes-from container3 ubuntu
- Now you are inside container and do ls, you can see the Volume2
- Now create one file inside this volume and check in container3, you can see that file

VOLUMES (HOST TO CONTAINER):

- Verify files in /home/ec2-use
- docker run -it - -name hostcont -v /home/ec2-user:/raham - -privileged=true ubuntu
- cd raham [raham is (container-name)]
- Do ls now you can see all files of host machine.
- Touch file1 and exit. Check in ec2-machine you can see that file.

SOME OTHER COMMANDS:

- docker volume ls
- docker volume create <volume-name>
- docker volume rm <volume-name>
- docker volume prune (it will remove all unused docker volumes).
- docker volume inspect <volume-name>
- docker container inspect <container-name>
- docker system df -v :

MOUNT VOLUMES:

- To attach a volume to a container: docker run -it --name=example1 --mount source=vol1,destination=/vol1 ubuntu
- To send some files from local to container:
  - create some files
  - docker run -it --name cont_name -v "$(pwd)":/my-volume ubuntu
- To remove the volume: docker volume rm volume_name
- To remove all unused volumes: docker volume prune

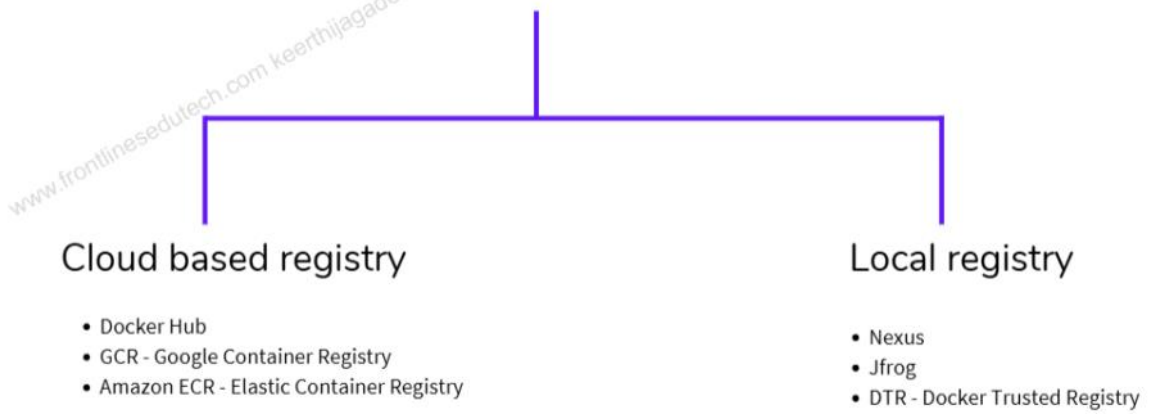
BASE VOLUMES:

STEPS

- create a volume : docker volume create volume99(volume-name)
- mount it: docker run -it -v volume99:/my-volume --name container1 ubuntu
- now go to my-volume and create some files over there and exit from container
- mount it: docker run -it -v volume99:/my-volume-01 --name container2 ubuntu

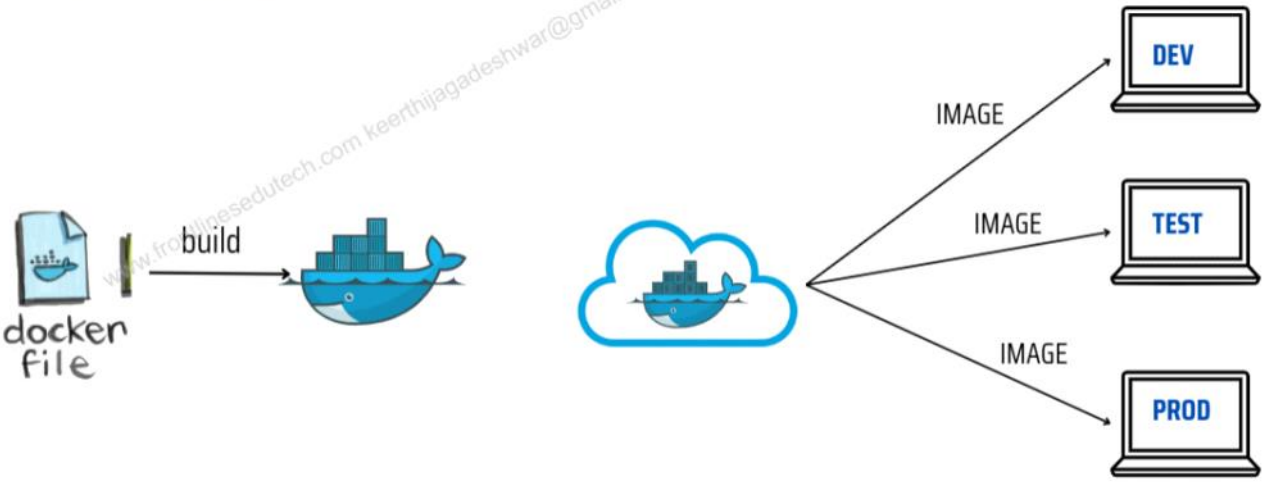
DOCKER REGISTRY:

It is used to store the images. Docker hub is the default registry



www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125

WHY DOCKER HUB



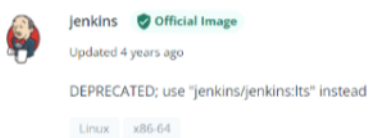
DOCKER PUSH:

- Select an image that includes docker and S.G SSH and HTTP enable anywhere on it.
- docker run -it ubuntu /bin/bash
- Create some files inside the container and create an image from that container by using - docker commit container-name image1
- now create a docker hub account
- Go to ec2-user and log in by using docker login.
- Enter username and password.
- Now give the tag to your image, without tagging we can't push our image to docker.
- docker tag image1 rahamshaik/new-image-name (ex: project1)
- docker push rahamshaik/project1
- Now you can see this image in the docker hub account.
- Now create one instance in another region and pull the image from the hub.
- docker pull rahamshaik/project1
- docker run -it - -name mycontainer rahamshaik/project1 /bin/bash
- Now give ls and cd tmp and ls you can see the files you created.
- Now go to docker hub and select your image -- > settings -- > make it private.
- Now run docker pull rahamshaik/project1
- If it is denied then login again and run it.
- If you want to delete image settings -- > project1 -- > delete

JENKINS SETUP USING DOCKER IMAGE:

DESCRIPTION: By using the docker file, we can set up the Jenkins dashboard without installing any dependencies.

- Login to docker hub
- search for Jenkins then you will get Jenkins official image.



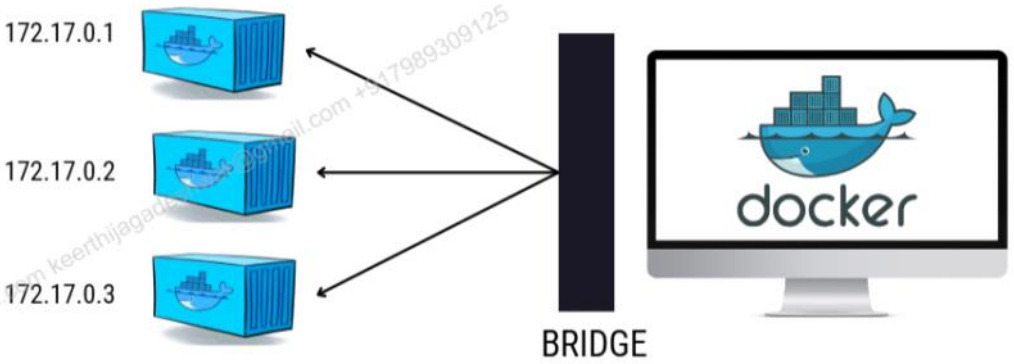
- Click on the image

www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125

- copy the code : docker pull jenkinsci/jenkins:lts
- run this code in docker engine
- now see docker images then you will get jenkins image
- create container using that image
- Now exit from the container and start the container again
- Inspect the container : docker inspect jenkins
- now go to jenkins image in docker hub and scroll down you will get command

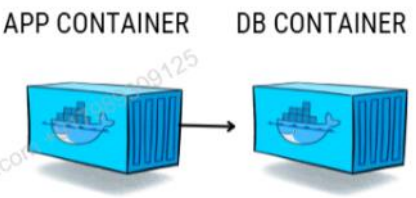
- run this command on docker engine and connect with Jenkins dashboard (public ip:8080)

DOCKER NETWORK:



WHY DOCKER NETWORK:

Lets assume we are having 2 containers like APP container and DB container. This App container has to communicate with DB container. So the developer will write a code to connect the application to the DB container.



The IP address of a container is not permanent. If a container is removed due to hardware failure, a new container will be created with a new IP, which can cause connection issues.

To resolve this issue we are using docker networks to create our custom network.

Docker networks are used to make a communication between the multiple containers that are running on same or different docker hosts. We have different types of docker networks.

- Bridge Network
- Host Network
- None network
- Overlay network

BRIDGE NETWORK: It is a default network that container will communicate with each other within the same host.

HOST NETWORK: When you Want your container IP and ec2 instance IP same then you use host network

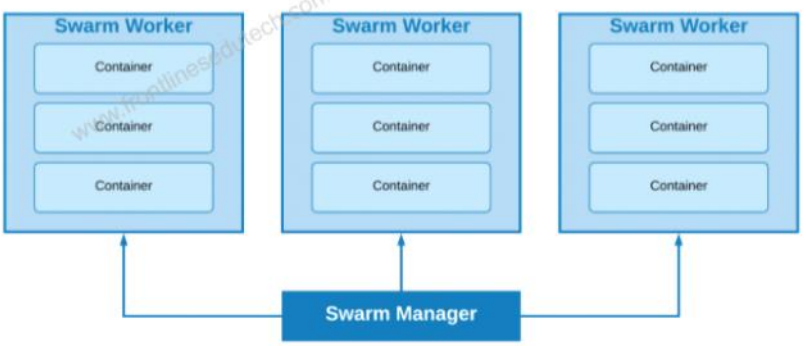
NONE NETWORK: When you don't Want The container to get exposed to the world, we use none network. It will not provide any network to our container.

OVERLAY NETWORK: Used to communicate containers with each other across the multiple docker hosts.

To create a network: docker network create network_name
To see the list: docker network ls
To delete a network: docker network rm network_name
To inspect: docker network inspect network_name
To connect a container to the network: docker network connect network_name container_id/name
apt install iputils-ping -y : command to install ping checks
To disconnect from the container: docker network disconnect network_name container_name
To prune: docker network prune

DOCKER SWARM:

- Docker swarm is an orchestration service within docker that allows us to manage and handle multiple containers at the same time.
- It is a group of servers that runs the docker application.
- It is used to manage the containers on multiple servers.
- This can be implemented by the cluster.
- The activities of the cluster are controlled by a swarm manager, and machines that have joined the cluster is called swarm worker.



- Docker Engine helps to create Docker Swarm.
- There are mainly worker nodes and manager nodes.
- The worker nodes are connected to the manager nodes.
- So any scaling or update needs to be done first it will go to the manager node.
- From the manager node, all the things will go to the worker node.
- Manager nodes are used to divide the work among the worker nodes.
- Each worker node will work on an individual service for better performance.

DOCKER SWARM COMPONENTS:

SERVICE: Represents a part of the feature of an application.
TASK: A single part of work.
MANAGER: This manages the work among the different nodes.
WORKER: Which works for a specific purpose of the service.

SETUP:

Create 3 node one is manager and another two are workers
Manager node: docker swarm init --advertise-addr (private ip)
Run the below command to join the worker nodes
To check nodes on docker swarm: docker node ls
Here * Indicates the current node like master branch on git
Now we created the docker swarm cluster
docker swarm leave : To down the docker node (need to wait few sec)
docker node rm node-id : To remove the node permenantly
docker swarm leave : To delete the swarm but will get error
docker swarm leave –force : To delete the manager forcefully
docker swarm join-token worker. : To get the token of the worker
docker swarm join-token manager : To get the token of the worker

SWARM SERVICE:

Now we want to run a service on the swarm
So we want to run a specific container on all these nodes
To do that we will use a docker service command which will create a service for us
That service is nothing but a container.
We have a replicas here when one replica goes down another will work for us.
At least one of the replica needs to be up among them.

docker service create --name raham --replicas 3 --publish 80:80 httpd
raham : service name replicas : nodes publish : port reference image: apache

docker service ls : To list the services
docker service ps service-name : To see where the services are running
docker ps : To see the containers (Check all nodes once)
docker service rm service_name : To remove the service (it will come again later)
public ip on browser : To check its up and running or not
docker service rm service-name : To remove the service

To create a service: docker service create —name devops —replicas 2 image_name
Note: image should be present on all the servers
To update the image service: docker service update —image image_name service_name
Note: we can change image,
To rollback the service: docker service rollback service_name
To scale: docker service scale service_name=3
To check the history: docker service logs
To check the containers: docker service ps service_name
To inspect: docker service inspect service_name
To remove: docker service rm service_name

DOCKER COMPOSE:

- It is a tool used to build, run and ship the multiple containers for application.
- It is used to create multiple containers in a single host.
- It used YAML file to manage multi containers as a single service.
- The Compose file provides a way to document and configure all of the application's service dependencies (databases, queues, caches, web service APIs, etc).

COMMANDS:

- Start all services: Docker Compose up.
- Stop all services: Docker Compose down.
- Run Docker Compose file: Docker-compose up -d.
- List the entire process: Docker ps.

COMPOSE FILE:

The Docker Compose file includes Services, Networks and Volumes.
The Default Path is ./docker-compose.yml, compose.yml (yaml=yml)
It contains a service definition which configures each container started for that service.

COMPOSE INSTALLATION:

```
sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose version
```

COMPOSE FILE:

- version - specifies the version of the Compose file.
- services - it the services in your application.
- networks - you can define the networking set-up of your application.
- volumes - you can define the volumes used by your application.
- configs - configs lets you add external configuration to your containers. Keeping configurations external will make your containers more generic.

CREATING DOCKER-COMPOSE.YML:

```
version: '3'
services:
  webapp1:
    image: nginx
    ports:
      - "8000:80"
```

vim docker-compose.yml
Version: It is the compose file format which supports the relavent docker engine
Services: The services that we are going to use by this file (Webapp1 is service name)
Image: Here we are taking the Ngnix image for the webserver
Ports: 8000 port is mapping to container port 80
Docker-compose up -d
Public-ip:8000 -- > You can see the Nginx image
Docker network ls -- > you can see root_default
Docker-compose down -- > It will delete all the Created containers

```
version: '3'
services:
  webapp1:
    image: nginx
    ports:
      - "8000:80"
  webapp2:
    image: nginx
    ports:
      - "8001:80"
```

Docker-compose up -d
Public-ip:8000 & public-ip:8001-- > You can see the Nginx image on both ports
Docker container ls
Docker network ls

CHANGING DEFAULT FILE:

mv docker-compose.yml docker-compose1.yml
docker-compose up -d
You will get some error because you are changing by default docker-compose.yml
Use the below command to overcome this error
docker-compose -f docker-compose1.yml up -d
docker-compose -f docker-compose1.yml down

```yaml
version: "3.1"
services:
  mobile_recharge:
    image: nginx
    ports:
      - "9999:80"
    volumes:
      - "mobile-recharge-volume1"
    networks:
      - "mobile-recharge-network"

  mobile_recharge2:
    image: nginx
    ports:
      - "9998:80"
networks:
  mobile-recharge-network:
   driver: bridge
~
```

docker-compose up -d - used to run the docker file
docker-compose build - used to build the images
docker-compose down - remove the containers
docker-compose config - used to show the configurations of the compose file
docker-compose images - used to show the images of the file
docker-compose stop - stop the containers
docker-compose logs - used to show the log details of the file
docker-compose pause - to pause the containers
docker-compose unpause - to unpause the containers
docker-compose ps - to see the containers of the compose file

DOCKER STACK:

- Docker stack is used to create multiple services on multiple hosts. i.e it will create multiple containers on multile servers with the help of compose file.
- To use the docker stack we have initialized docker swarm, if we are not using docker swarm, docker stack will not work.
- once we remove the stack automatically all the containers will gets deleted.
- We can share the containers from manager to worker according to the replicas
- Ex: Lets assume if we have 2 servers which is manager and worker, if we deployed a stack with 4 replicas. 2 are present in manager and 2 are present in worker.
- Here manager will divide the work based on the load on a server

COMMAND:
TO DEPLOY : docker stack deploy --compose-file docker-compose.yml stack_name
TO LIST : docker stack ls
TO GET CONTAINERS OF A STACK : docker stack ps stack_name
TO GET SERVICES OF A STACK: docker stack services stack_name
TO DELETE A STACK: docker stack rm stack_name

DOCKER INTEGRATION WITH JENKINS

- Install docker and Jenkins in a server.
- vim /lib/systemd/system/docker.service

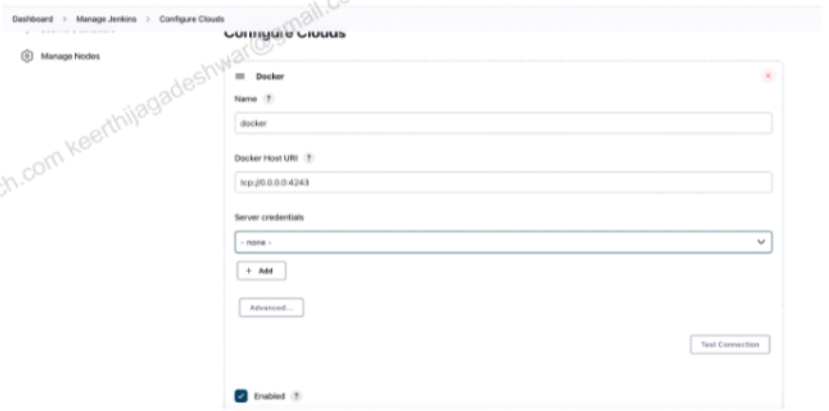
```
[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:4243 -H unix:///var/run/docker.sock
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always
```

- Replace the above line with

ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:4243 -H unix:///var/run/docker.sock

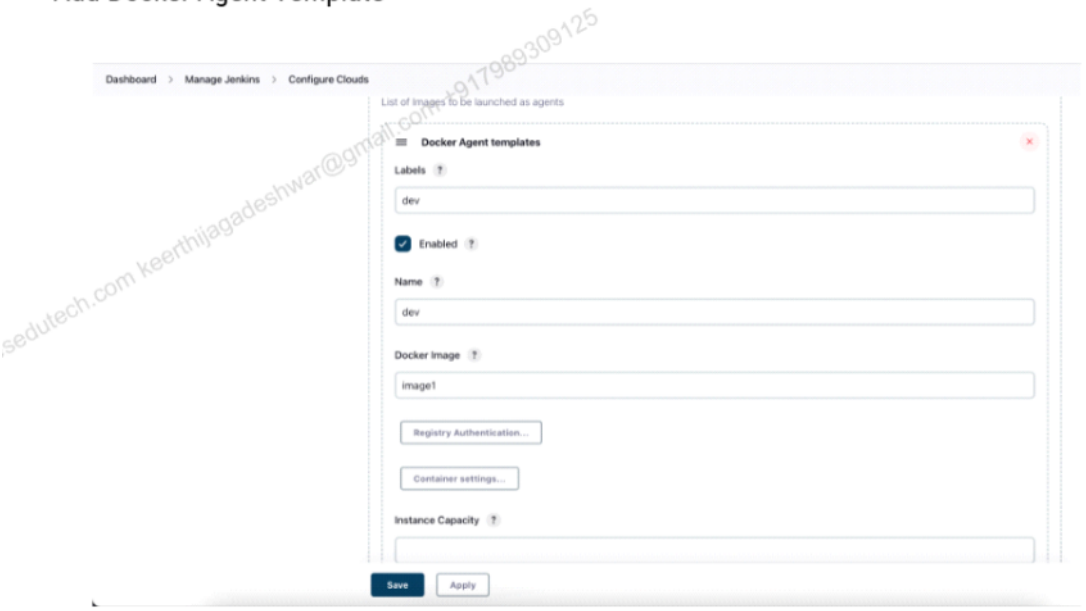
- systemctl daemon-reload
- service docker restart
- curl http://localhost:4243/version

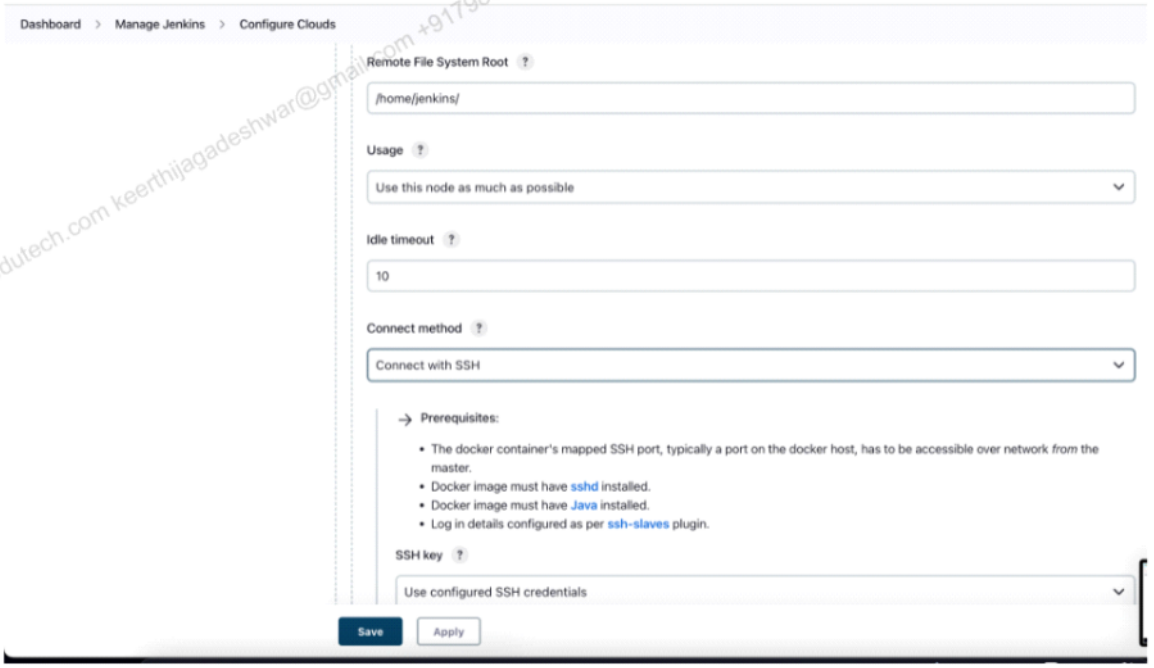
- Install Docker plugin in Jenkins Dashboard.
- Go to manage jenkins>Manage Nodes & Clouds>>Configure Cloud.
- Add a new cloud >> Docker
- Name: Docker
- add Docker cloud details.

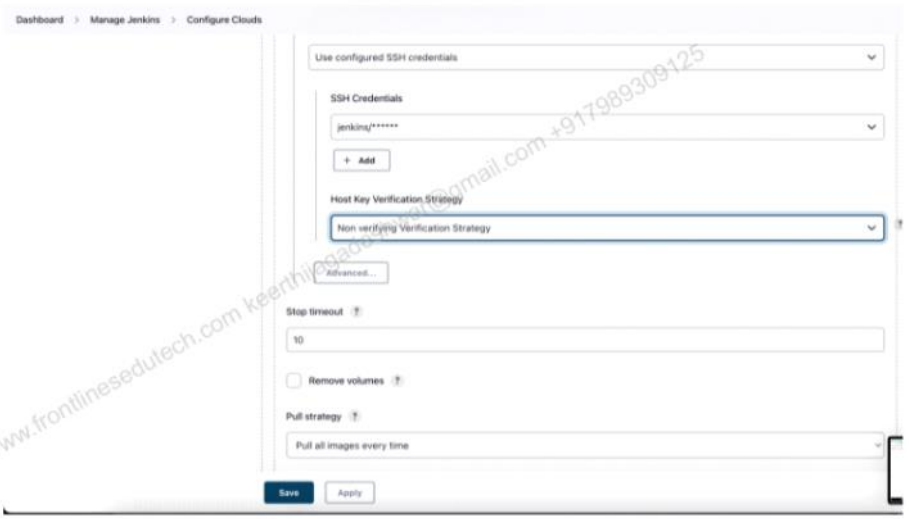


Jutech.com keerthijagadeshwar@gmail.com +917989309125

- Add Docker Agent Template







- Save it and do and watch the container in Jenkins dashboard.
- Manage Jenkins>>Docker (last option)

DEPLOYMENT DOCKER FILE:

Create 2 files:

1. Dockerfile
2. index.html file

Dockerfile consists of

```
FROM UBUNTU
RUN APT-GET UPDATE -Y
RUN APT-GET INSTALL APACHE2 -Y
COPY INDEX.HTML /VAR/WWW/HTML/
CMD ["/USR/SBIN/APACHECTL", "-D", "FOREGROUND"]
```

Index.html file consists of

```
<H1>HI THIS IS MY WEB APP</H1>
```

www.frontlinesedutech.com keerthijagadeshwar@gmail.com +917989309125

Add these files into GitHub and Integrate with Jenkins by declarative code pipeline.

```
pipeline {
  agent any
  stages {
    stage ("git") {
      steps {
        git branch: 'main', url: 'https://github.com/devops0014/dockabnc.git'
      }
    }
    stage ("build") {
      steps {
        sh 'docker build -t image77 .'
      }
    }
    stage ("container") {
      steps {
        sh 'docker run -dit -p 8077:80 image77'
      }
    }
  }
}
```

You will get Permission Denied error while building the code.
To resolve that error you need to follow these steps:

- usermod -aG docker jenkins
- usermod -aG root jenkins
- chmod 777 /var/run/docker.sock
- systemctl daemon-reload

Now you can build the code and it will gets deployed.

DOCKER DIRECTORY DATA:

We use docker to run the images and create the containers. but what if the memory is full in instance. we have a add a another volume to the instance and mount it to the docker engine. Lets see how we do this.

- Uninstall the docker - yum remove docker -y
- remove all the files - rm -rf /var/lib/docker/*
- create a volume in same AZ & attach it to the instance
- to check it is attached or not - fdisk -l
- to format it - fdisk /dev/xvdf --> n p 1 enter enter w
- set a path - vi /etc/fstab (/dev/xvdf1 /var/lib/docker/ ext4 defaults 0 0)
- mkfs.ext4 /dev/xvdf1
- mount -a
- install docker - yum install docker -y && systemctl restart docker
- now you can see - ls /var/lib/docker
- df -h

PORTAINER:

- it is a container organizer, designed to make tasks easier, whether they are clustered or not.
- abel to connect multiple clusters, access the containers, migrate stacks between clusters
- it is not a testing environment mainly used for production routines in large companies.
- Portainer consists of two elements, the Portainer Server and the Portainer Agent.
- Both elements run as lightweight Docker containers on a Docker engine

PORTAINER:

- Must have swarm mode and all ports enable with docker engine
- curl -L https://downloads.portainer.io/ce2-16/portainer-agent-stack.yml -o portainer-agent-stack.yml
- docker stack deploy -c portainer-agent-stack.yml portainer
- docker ps
- public-ip of swamr master:9000

DOCKER RUN VS CMD VS ENTRYPOINT:

RUN: it is used to execute the commands while we build the images and add a new layer into the image.

FROM centos:centos7
RUN yum install git -y
or
RUN ["yum", "install", "git" "-y"]

CMD: it is used to execute the commands when we run the container.
It is used to set the default command.
if we have multiple CMD's only last one will gets executed.

FROM centos:centos7
CMD yum install maven -y
or
CMD ["yum", "install", "maven", "-y"]

If you want to overwrite the parameters:
docker run image_name httpd (FAILED)
docker run image_name yum install httpd -y (only httpd will gets installed)

ENTRYPOINT: it overwrites the CMD when you pass additional parameters while running the container.

FROM centos:centos7
ENTRYPOINT ["yum", "install", "maven", "-y"]

If you want to overwrite the parameters:
docker run image_name httpd (both maven and httpd will gets installed)
docker run image_name yum install httpd -y (both maven and httpd will gets installed)

FROM centos:centos7
ENTRYPOINT ["yum", "install", "-y"]
CMD ["httpd"]

Bydefault it will executes httpd command, if you specify the command while running the container it will gets executed.
docker run image_name (httpd will install)
docker run image_name git (only git will install)
docker run image_name git tree(both git & tree will install)

DOCKER FILE TO DEPLOY NODE JS FILE:

```
FROM node:16
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 8081
CMD ["node", "index.js"]
```

REFERENCE: https://github.com/devops0014/nodejs-docker.git

DOCKER FILE TO DEPLOY STATIC WEBSITE USING NGINX:

```
FROM ubuntu
RUN apt-get update
RUN apt-get install nginx -y
COPY index.html /var/www/html/
EXPOSE 80
CMD ["nginx","-g","daemon off;"]
```

OR

```
FROM nginx
COPY . /usr/share/nginx/html
```

REFERENCE: https://github.com/devops0014/staticsite-docker.git

TIC-TAC-TOE GAME

REFERENCE: https://github.com/devops0014/tic-tac-toe-docker.git

CONTAINER LOAD BALANCER:

```
.
├── docker-compose.yml
├── nginx
│   ├── Dockerfile
│   └── nginx.conf
├── swiggy
│   ├── Dockerfile
│   └── index.html
└── zomato
    ├── Dockerfile
    └── index.html
```

DOCKER FILE FOR SWIGGY AND ZOMATO SERVICES

```
FROM nginx
COPY . /usr/share/nginx/html
```

DOCKER FILE FOR NGINX

```
FROM nginx
RUN rm /etc/nginx/conf.d/default.conf
COPY nginx.conf /etc/nginx/conf.d/default.conf
```

NGINX.CONF FILE

```
upstream loadbalancer {
server 172.17.0.1:5001 weight=6;
server 172.17.0.1:5002 weight=4;
}
server {
location / {
proxy_pass http://loadbalancer;
}}
```

COMPOSE FILE

```
version: '3'
services:
  swiggy:
    image: image1
    ports:
      - "5001:80"
  zomato:
    image: image2
    ports:
      - "5002:80"
  nginx:
    build: ./nginx
    ports:
      - "8080:80"
    depends_on:
      - swiggy
      - zomato
```

After writing all the file, we need to build the Dockerfiles for both the zomato and swiggy services.
command: **docker build -t image_name .**

write the docker-compose file and build it.
command: **docker-compose up -d**

access the application: **publicip:8080**

DOCKER SAVE:
docker image save swiggy:v1 > swiggy:v1.tar
docker image history swiggy:v1
docker rmi swiggy:v1
docker images
docker image load < swiggy\:v1.tar

save vs export

PROJECT: USE DOCKER STACK FOR PROJECT
Take 2 nodes 1 manger & 1 worker & setup jenkins in manager
setup a cluster of docker swarm
create images and upload to docker hub
pull images to both nodes (opt)
write a compose file with replicas
write a pipeline and build it

MY PROJECT LINK : https://github.com/RAHAMSHAIK007/evedockerproject.git

memory management:

docker run -it --cpus=".5"--name cont-name ubuntu /bin/bash
docker run -d --name cont3 --memory 50M nginx : It can use now maxx 50 MB
docker stats

DOCKER FILE TO DEPLOY WAR FILE

```
FROM tomcat:8.0.20-jre8
COPY tomcat-users.xml /usr/local/tomcat/conf/
COPY target/*.war /usr/local/tomcat/webapps/myweb.war
```

Session : 55

DOCKER INTERVIE...

DOCKER-PROJECT-PIP...

DOKCER PROJECT

DOKCER PROJECT

Session : 57

minikube-script (1)

Session : 59

kops (1) (2)

Session : 64

DEPLOYMENT STRAT...

Free-Ethical-Hacking-bootcamp

Notes Day1

free_ethical_hacking_...

Session : 69

KUBERNETES NOTES (1)

Session : 70

NAGIOS (1)

nagios client ser...

nagios client ser...

Session : 76

STORAGE CLASSES I...

Session : 78 (Resume_Building)

EXP RESUME (...

FRESHER RESUME (...

Session : 83

CLOUD WATCH - ...

Session : 86

bash shell (1)

ADV BASH

Session : 87

TERRAFORM FULL N...

Session : 89

MONITORING IN K8S

MONITORI
NG IN K8S

Session : 90

AWS
INTERVIE...

TERRAFOR
M INTERV...

KUBERNETE
S INTERVI...

DOCKER
INTERVIE...

ANSIBLE
INTERVIE...

JENKINS
INTERVIE...

GIT
INTERVIE...

DOCKER
CHEAT SH...