Linux---  
  
unix based operating system.

Architecture is – hardware -> kernal->shell->application program

Kernal :-

It is the heart of the os. It is a program that acts as a abstraction of the hardware.

So if any thing is required from the hardware the kernal communicates to the hardware and gets it. (memory management , task shcedulng)

Terminal:-

It is nothing but the workarea where we type the command

Shell:-

When a terminal is open it runs a shell.

It is nothing but a utility program which allows the user to interact with the os.

It takes the command types as input , process it and gives back the output.

Linux safe:-

Scenario based example

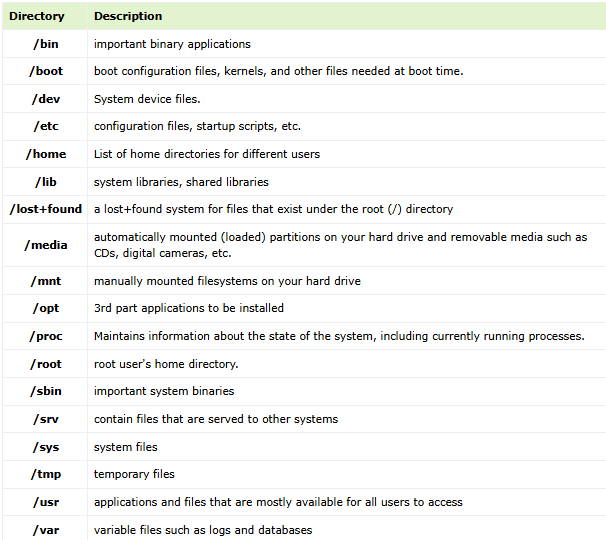
If we want to install something in linux it always asks for the password. Root user can be allowed to do that but normal user has no rights.

Same in windows admin . by default every user has admin rights.

So if an attacker tries to install a malware in windows It can be done and if at all admin pass required it can be bypassed / exploited.

But whereas in linux it only asks for the password in terminal which cant.  
by this way linux uses more cli than normal ui , so it is safe .

Linux directories:-

In linux file is a directory.  


File permissions:-

1. Filetype

2,3,4 - permission to the users

5,6,7- permission to the groups

8,9,10- permission to the others

Change ownership:-

Chown user:group /dir

Adding permission:-

Chmod g+r /dir ( g- group r-read + add)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated  
  
Services:-

Are the background process that contribute the functionality to the system. Such as networking , authentication etc.

Systemctl status <service name> ------ check the status of the service

Systemctl start<service name> ---------- to start a service

Systemctl stop<service name>.-------- to stop a service.

LINUX PROCESS:-

Process - nothing but a program which is in execution

Zombie process – a process that completes its execution but still have a entry inprocess table. Because its parents doesn’t record a exit.

Orphan process – is a process who s parent process has completed its execution but the child doesn’t . this is orphan.

BASH SCRIPTNG:-

**Bash** (Bourne Again SHell) is a command-line shell and scripting language.

Default shell in unix based systems.

BASH scripting – refers to the series of commands that automates a task(ex regular backups )

#!/bin/bash – shebang character. Tells the system that script should be executed in bash shell.

Variables – temp storage on process   
 created by $VARNAME

Commandline aurguments – the values that are passed to the script.  
 ex: in script file I need to print the value which the user gives in the terminal, then

In script file – echo $1

While executing - ./fiile.sh Krithik ##### Krithik is the argument passed to the script file

$0 – file name

$1-$9 – are the variables.

GETTING USER INPUTS:-

Read -p varname

-p is the prompt like(enter the number );

Conditional statements:

If syntax ------- if [ $n -gt 100 ] ----space is important

VIRTUALIZATION:-

Allows us to create the virtual versions of physical resources such as hardware / os etc

--virtual machines—

Virtual machines are the emulation of the physical machines. So we can run multiple virtual machines in the single system . it uses the system’s resources such as cpu etc.

--hypervisor------

It is the layer of software that enables us to create virtual machines.

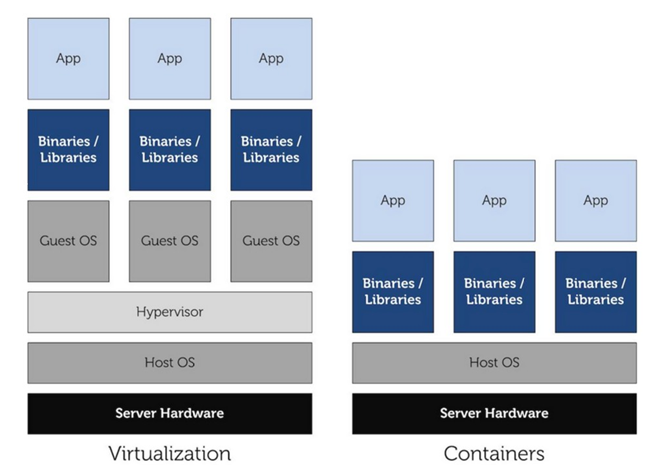
CONTAINERS:-

virtual machines are the hardware virtualization

containers are the os virtualization

--applications inside the virtual machines are uses the vm’ os where applications in the container uses the host os.

Example:- if I host a java application inside the container then the application uses the libraries packages of the container but the computing resources are relied on the host.

* Containers are isolated. Every container has their own process tree , networking , ip port etc
* 

DOCKER :-

Docker is a Paas tool which allows us to create and run containers.

Docker engine is responsible for creating and running isolated containers.

A close-up of a computer screen

Description automatically generated

DOCKER IMAGES:-

It is the stopped state of the container. It is used to run a container. When you run an image it becomes a container.

Docker images are stored in the docker registries called docker hub.

DOCKER FILE:-

Docker file contains the commands in order to create an docker image.

NAMESPACE:-

Namespace helps in isolation between the each containers.

CGROUPS:-

Cgroups is the Linux kernal feature and it allocates and manages the resources(cpu etc) among the processes.

DOCKER VOLUMES:-

Docker Volumes are a mechanism for persisting data that is generated or used by containers. Volumes allow data to persist beyond the lifecycle of a container and can be shared between containers.

DOCKER NETWORKING:-

So docker networking allows 2 containers on the host machine to communicate with each other .

* **Bridge Network** is the default. Containers connected to the same bridge network can communicate with each other by their container names.
* **Host Network** directly binds containers to the host's networking stack, bypassing Docker's network isolation. Useful for high-performance applications.
* **None Network** disables networking for a container. It has no network interfaces.
* **Custom Networks** allow containers to have private communication among themselves.

DOCKER COMPOSE:-

Docker Compose is a tool for defining and running multi-container Docker applications. With Compose, you can use a YAML file to configure your application's services, networks, and volumes. Then, with a single command (docker-compose up), you can create and start all the services from the configuration.

DOCKER MULTI-STAGE BUILDS: -

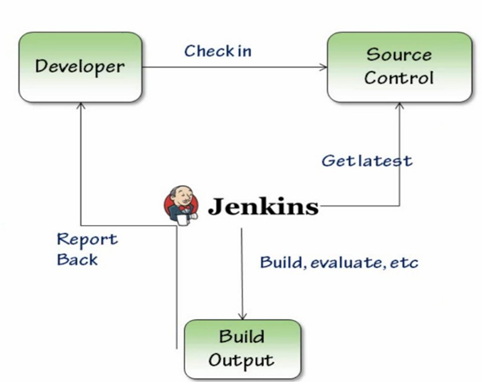
In a multi-stage build, you can separate the build and runtime environments into different stages. This allows for a cleaner, smaller final image since you can discard build tools and unnecessary dependencies after compiling your application.

FOR EXAMPLE:-

In this multi stage builds first we will specify the base image and app code and the dependencies and the libraries that are needed to run the app code. Using this docker builds the app and generates the artifacts. After that in run time we just use the app that was build by the build stage and make that app run. So this eliminates the installation of libraries and dependencies in the run time and makes the final image less size.

CONTINUOUS INTEGERATION:-

It is the process of automating the build and testing of code when a change is committed to a vcs.



A diagram of a computer network

Description automatically generated

Jenkins master :-

The main server of Jenkins is the Jenkins Master.

It is a web dashboard which is nothing but powered from a war file. By default it runs on 8080 port.

With the help of Dashboard, we can configure the jobs/projects but the build takes place in Nodes/Slave.

By default one node (slave) is configured and running in Jenkins server. We can add more nodes using IP address, user name and password using the ssh, jnlp or webstart methods.

Jenkins slave:-

Jenkins slave is used to execute the build jobs dispatched by the master.

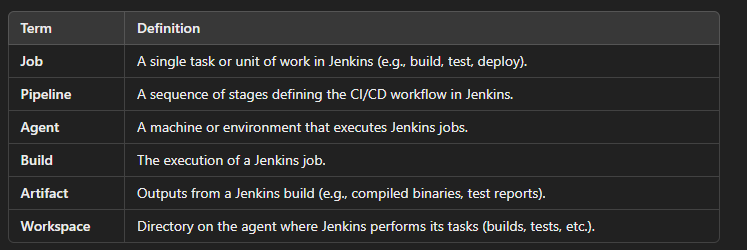
We can configure a project to always run on a particular slave machine, or particular type of slave machine, or simple let the Jenkins to pick the next available slave/node.

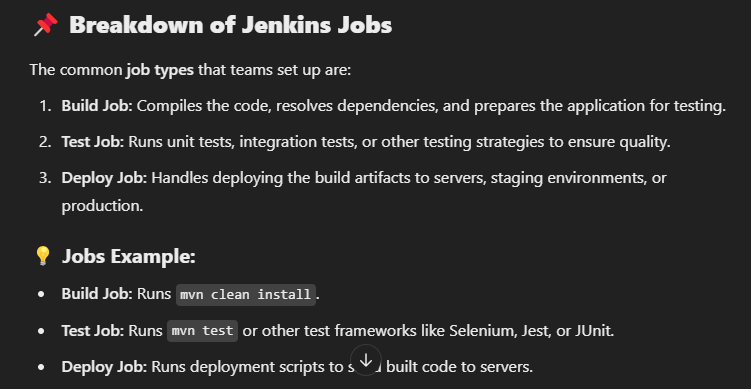
A diagram of a workload

Description automatically generated

TERMINOLOGIES:-

1. Job:- A **Job** in Jenkins refers to a single unit of work. It's the basic task that Jenkins performs. Jobs can be defined to build, test, deploy, or perform any task related to a project





A screenshot of a computer program

Description automatically generated

Jenkins ci:-

Setting up 3 instances for Jenkins,sonarqube , nexus

Jenkins->

Tools - > install necessary ( jdk , maven)

New job -> freestyle-> specify git url and branch -> select build steps (execute shell)

Pipeliejob -> same but apply Jenkins script file

Sonarqube integration->

In Jenkins->

Tools -> install sonarqube scanner

System-> select sonarqube and give the url (private ip of sonar) -> in sonar generate token and paste .

**Codeanalysis ->**

While writing the Jenkins file we specify the sonar qube scanning command in it . and create pipeline job and paste the Jenkins file so after building this job, output of code analysis will be in sonarqube server.

**Qualitygates:-**

These are like threshold . ( ex: if code has above 30 quality gate then scan should result in fail).

To add quality gate:-

In sonarqube -> quality gates -> create -> give name and add contion-> (like bug limit 30)

And in project modify the quality gate to this created one.

Add webhook:-

Webhook is after scanning the quality gate sonar should contact Jenkins

Webhook->create -> url ( Jenkins private ip : port 8080)

In Jenkins file:-

Add the qualitygate code in the stages.

Docker Integeration :-

Normally this pipeline builds the application and generate artifacts.

So we should write docker file as running the image application from the artifacts.

So as a pipeline it will runs the application in the docker container.

Build triggers :-

Automatically triggering the build .

1-web hooks – where if there is a new commit in github then the github sends a json payload to Jenkins to trigger the build

1. Poll scm – Jenkins monitors the github for a new commit and triggers the build.

**ANSIBLE:-**

**Ansible is an open-source IT engine that automates application deployment, cloud provisioning, intra service orchestration, and other IT tools.**

**Does not need any agent to install and run**

**Scripting – just running the scripts does not check the states.**

**Configuration management – compare the states and apply the changes.**

**--used for:**

**1- any system management**

**2-change management – any production server changes**

**3-provisioning- setup servers**

**4-orchestration-large scale automation.**

**--workflow:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A close up of text

Description automatically generated**

Inventory: where we write the hosts what we want to connect.

Example inventory file :

A screenshot of a computer

Description automatically generated

We have specified the hosts such as ec2 instances .

Execute command:

Ansible myweb01 -m ping -I inventory. ( -m is the module,ping is the module here)

(-I is the inventory )

Ping is the adhoc commands.

ANOTHER EXAMPle :

* Created a index file
* Should install httpd service in targets and run this site.

To install httpd and start service:

Ansible webservers -m ansible.builtin.yum -a “name=httpd state=present” -I inventory –become

-m module -builtin.yum to install service in redhat linux

-a arguments- what to install name,state

-I inventory - targets are specified

--become – giving root user access to the controller.

Ansible webservers -m ansible.builtin.service -a “ name=httpd state=started enabled=yes” -I inventory –become

* We are starting the service here.

Now we are having index.html in controller and we should host it it target(webservers)

* Ansible webservers -m builtin.copy -a “ src=index.html dest=/var/www/html/index.html” -I inventory –become

After this the site will be running in all of the targets ( webservers).

(note: if we donot do any changes and run this command means , it will just return success as there is no change . but if u change the html and run this command then it will compare the state and apply the changes. This is confihuration management).

A black screen with white text

Description automatically generated

PLAYBOOK:

A black and white text on a black background

Description automatically generated

A screenshot of a computer

Description automatically generated

Playbooks – written in yaml. It is like specifying the tasks that we need to automate or configure it on the targets ..

Example :

A screenshot of a computer program

Description automatically generated

The same task where we need to install httpd on webservers and maria db on dbserver

EXECUTING THE PLAYBOOK:

Ansible-playbook -I inventory yamlfilename.yaml

VARIABLES IN PLAYBOOK:

A screen shot of a computer program

Description automatically generated

Will bbe difeined under vars:

Ans will be substituted in {{ varname }}

OUTPUT / PRINTING:-

Debug; debug is used to print the message or variables to the console.  
from the above variable example to print the package name

Debug:

Var: {{ package name}} /// it outputs the package name httpd

TERRAFORM:-

Terraform is an open source Iaac tool for creating,managing and automating the it infrastructure through human readable configuration files written in hashicorp(HCL).

We can eliminate the need for manual provisioning by automating the it infra.

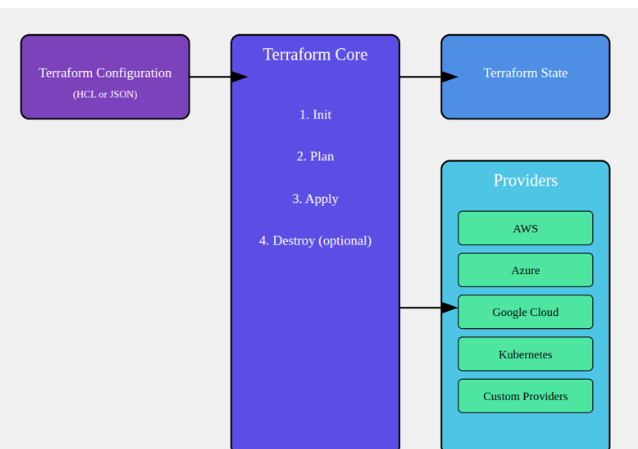
DIFFERENCE BETWEEN ANSIBLE AND TERRAFORM:-

Ansible is used for automating the tasks such as software installation , it infra provisioning , configuration management etc

Where as terraform is used for provisioning infra on the cloud platforms are premises.

Steps :-

1. Write configuration in hcl – ( main.tf)
2. ( terraform init) – initializes terraform and download plugins needed for written code.
3. (plan terraform) – this gives the preview of what it will provision or create
4. ( terraform apply) – once after preview apply the changes
5. (terraform destroy) - destroy those created .



Terraform concepts:-

1. Providers

**A screenshot of a black screen

Description automatically generated**

**Code of the provider:-**

Provider “aws” {

Region = “us-east-1”

}

1. RESOURCES:-
   1. Resources are the components of the infrastructure. ( eg. Ec2, vm , bucket)

Code:

Resource “aws\_instance” “demo”{

Ami =” ami id”

Instance\_type = “t2.micro”

}

1. VARIABLES:-
   1. Variables allows us to parameterize the components .
   2. We can add variable for ec2 region instanc type etc

Code:-

In vars.tf

Variable “variablename” {

Description =” region”

Default = “ us-east-1”

}

In main.tf

Provider “aws” {

Region = var.variablename

}

5. outputs:-

It helps us to extract the information from the resource created.

Output instance\_ip {

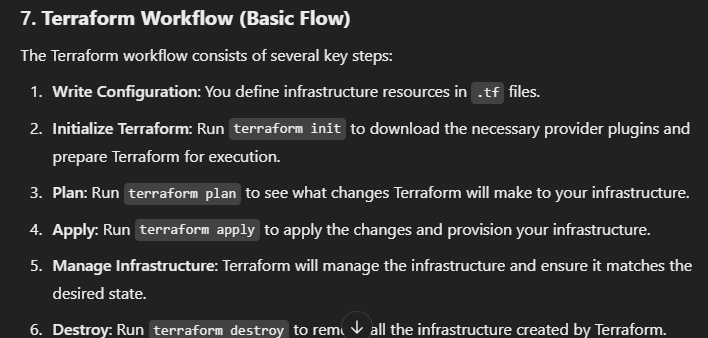
Value = aws\_instance.instancename.public\_ip

}

6. state :-

It stores the information of the current state that the infra manages it.

The states are stored in terraform.tfstate



A black and white screen with white text

Description automatically generated

CONTAINER ORCHESTRATION:-

Process of managing and automating the deployment , scaling the containarised applications.

Kubernetes is the famous orchestration tool.  
architecture of k8s  
A diagram of a computer network

Description automatically generated

MASTER NODE:

It Is called as control plane.

It manages cluster’s overall scheduling, workload management. It controls the worker nodes and ensures it is in correct state.

Components of master node:

1. Api server
2. Controller
3. Scheduler
4. Etdc server
5. Api server:

* It is the front end for master node.
* It exposes rest api (kubectl) where users can communicate through
* It receives the commands about the pods and deployment and validates before sending it to appropriate component.

1. Controller manager

* Responsible for monitoring the clusters state and making decision to maintain the state
* It runs several controller such as node , replication , deployment contrilelr

1. Scheduler:-

* It assigns pods to the worker nodes in the cluster based on available resources
* Based on the requirements and the availability it assigns pods

1. Etdc:-

* Key value store to store the cluster data
* Data such as secrets, state, configuration, metadata etc

WORKER NODES;-

Worker nodes are the actuall machines where container actually runs

Components of worker nodes :

1. Kubelet
2. Kube proxy
3. Container runtime
4. Pods
5. Kubelet

* It is an agent that runs on the worker nodes in the cluster
* Makes sures that containers are running in the desired state.
* It watches the api servers for change and manages the containers based on those changes
* It monitors the status of pods and ensures that cintaier are healthy.

1. Kubeproxy

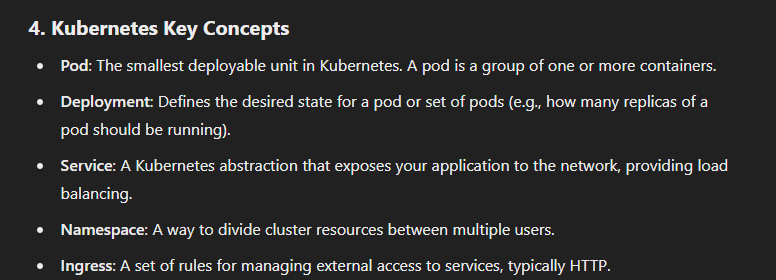
* It is responsible for maintaining networking and ensures that containers can communicate
* It implements load balancing within the clusters
* Routes traffic to appropriate pods.

1. Container runtime

* Running the containers.
* It is docker

1. Pods

* Smallest unit
* It is group of one or more containers that shares the same network , volumes , namespace etc.
* Pods give the resources for the container to run



A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated