* **CICD**

CI is process where developer frequently merge their code change into shared repository then automated build and testing.

Continuous delivery is extension of continuous integration, after everything working fine then there is manual approval to deploy code in production & continuous deployment where code is deploy without any manual approval.

* **Git Merge & Git Rebase**

Git Merge & Git Rebase are design for same purpose to integrate changes from one branch to another branch. But they do it in different ways.

For example, when I am working on new feature in my project and one of my team member update their code with new commit and that new commit is relevant to feature that I am working on then I would integrate those new commit in my feature branch. So to do this I have two options merge and rebase. When I do git merge it will create new merge commit in my feature branch including history of both branches. Then when I do git rebase then entire feature branch come on tip of another branch with clean project history.

* **Git Squash**

It is basically merging several commits into a single commit using interactive git rebase command. It keeps our commit history clean and easy to understand.

* **What is git cherry-pick ?**

Cherry-picking in Git stands for applying some commit from one branch into another branch. In case you made a mistake and committed a change into the wrong branch, but do not want to merge the whole branch. You can revert the commit and apply it on another branch.

* Build lifecycle of maven

Clean, Default, Site :- In default we have 7 lifecycle

Validate, Compile, Test, Package, Verify, Install, Deploy

* Maven Commands
* **mvn clean:** Cleans the project and removes all files generated by the previous build.
* **mvn compile:** Compiles source code of the project.
* **mvn test-compile:** Compiles the test source code.
* **mvn test:** Runs tests for the project.
* **mvn package:** Creates JAR or WAR file for the project to convert it into a distributable format.
* **mvn install:** Deploys the packaged JAR/ WAR file to the local repository.
* **mvn deploy:** Copies the packaged JAR/ WAR file to the remote repository after compiling, running tests and building the project.
* **mvn site:** Generate site documentation for project
* **What is GAV in maven ?**

GroupID, ArtifactID, Version used to uniquely identified your project

* JAR is default value of packing tag
* **Post Block in Jenkins**

It run on end of pipeline execution. We can perform several tasks like cleaning, finalisation, notification. We have some test condition used in post block of Jenkins like always, unstable, success.

* Backup in Jenkins

All the settings, build logs are stored under Jenkins’s home directory. So, we can just take backup of Jenkins home directory. It will be under **.jenkins** path. Also, we have Jenkins backup plugin.

* Docker Compose

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application's services. Then, using a single command, you create and start all the services from your configuration.

* Docker Volume

When use docker volume it will create new folder in host system. It is used to preserved the while working with container. Types : 1) Host Volume 2) Anonymous Volume 3) Named Volume

* Bind Mounts

A Bind Mount is a**storage area (file/directory) on your local machine available inside your container**. So any changes you make to this storage space (file/directory) from the outside container will be reflected inside the docker container and vice-versa.

* Taints and tolerations

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more **taints are applied to a node**; this marks that the node should not accept any pods that do not tolerate the taints.

* ConfigMaps

A ConfigMap is an API object used to store non-confidential data in key-value pairs

* Deployment strategy in K8s

Rolling update is default strategy in k8s. We have two strategies recreate and rolling update. In rolling update application never goes down all the instances at once. But in recreate strategy all instances destroy once and again create new one.

* Helm in K8s

[**Helm**](https://helm.sh/) is a package manager for Kubernetes applications**.**

[Helm](https://helm.sh/) is a package manager for Kubernetes applications that includes templating and lifecycle management functionality

* Git Stash
* Password authentication

# vim /etc/ssh/sshd\_config --🡪 Permitrootlogin yes

-🡪 Passwordauthentication yes

After this # systemctl restart sshd

* Passwordless authentication

ssh-keygen

ssh-copy-id root@ipadrr

* **How do you scale k8s ?**

1. Horizontal Pod Autoscaler (HPA)
2. Cluster Autoscaler (Number of node increases)( This 2 will create infra but it take some time)
3. Cluster Overprovisioning (It will create dummy ec2 and pods and when load get increases that time within no time that other dummy are come in picture.)

* **What is Ingress in K8s?**

Kubernetes Ingress, an API object that defines the routing rules explaining how the external users can access the services running within the Kubernetes cluster via HTTPS/HTTP.

Ingress will enable you to set up the routing rules for incoming traffic without the need to create several Load Balancers or exposing the running services on the node.

That is why it is the best option for production environments.

Jenkins Pipeline

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building..'

}

}

stage('Test') {

steps {

echo 'Testing..'

}

}

stage('Deploy') {

steps {

echo 'Deploying....'

}

}

}

}

* **Two different account having having VPC, How we connect that 2 vpc of different account ?**

By using transit gateway we can establish connection between two account vpc.

* **What is VPC peering ?**

VPC Peering is a**networking connection** that allows you to connect one VPC with another VPC through a direct network route using private IP addresses.

* **What is the difference between an IAM user and role in AWS?**

IAM users can sign in to the AWS Management Console for interactive tasks and to make programmatic requests to AWS services using the API or CLI. A user in AWS consists of a name, a password to sign into the AWS Management Console, and up to two access keys that can be used with the API or CLI.

IAM roles are identities you can create with specific permissions for short durations. You can assign IAM roles to entities you trust so that those entities can assume the role when needed. IAM Roles are primarily meant for internal use.

1. **User can sign in**
2. **Console for Interactive tasks**
3. **User has name, password for sign**
4. **Roles are identities with specific permission for short duration**
5. **Roles use for internal use.**

### **How will you secure Jenkins?**

To configure Security in Jenkins, follow the steps given below.  
  
Step 1 − Click on Manage Jenkins and choose the 'Configure Global Security' option.  
  
Step2 − Click on Enable Security option.  
  
Step 3 − You will be prompted to add your first user.  
  
Step 4 − It's now time to setup your users in the system.

* **what is matrix based authentication in Jenkins ?**

Matrix-based security is **one of the authorization strategies available for securing Jenkins**. It allows you to grant specific permissions to users and groups.

### **What is the concept of sudo in Linux?**

Sudo stands for either "substitute user do" or "super user do" and it allows you to elevate your current user account to have root privileges temporarily. This is different from the“su” which is not temporary.

* **What is VPC ?**

A virtual private cloud (VPC) is a secure, isolated private cloud hosted within a public cloud.

* **What is Subnet?**

A *subnet* is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that won't be connected to the internet.

* **Git Stash**

Sometimes you want to switch the branches, but you are working on an incomplete part of your current project. You don't want to make a commit of half-done work. Git stashing allows you to do so. The **git stash command** enables you to switch branches without committing the current branch.

* **Jenkins Backup**

[Jenkins Thin Backup](https://wiki.jenkins.io/display/JENKINS/thinBackup) is a popular plugin for backing up Jenkins. It backs up all the data based on your schedule and it handles the backup retention as well.

### **What could be the steps to move or copy Jenkins from one server to another?**

1.Copy .jenkins folder from one server to another server

2.You may move a job from one Jenkins installation to another just by copying the corresponding job directory.

3.You may make a copy of an already existing job by making a clone of the job directory with an uncommon name.

4.You may also just rename a current job by renaming a directory.

### **Assume that you have a pipeline. The first job that you performed was successful, but the second one failed.  What would you do now?**

You don't have to worry, and you just have to restart the pipeline from the point where it failed by doing 'restart from stage.'

### **How can the parameters be defined in Jenkins?**

In Jenkins, a build can take many input parameters to execute.

* To define parameters for the job, select the “this project is parameterized” box.
* The drop down “Add Parameter” is enabled with the parameter types list. Any number of parameters may be added in the list.

There are several parameter types provided in the list

* **Difference between CMD & Entrypoint**

CMD is an instruction that is best to use if you need a default command which users can easily override. If a Dockerfile has multiple CMDs, it only applies the instructions from the last one.

On the other hand, **ENTRYPOINT** is preferred when you want to define a container with a specific executable.

## What are Docker Volumes?

Docker volumes are file systems mounted on [Docker containers](https://phoenixnap.com/kb/docker-image-vs-container) to preserve data generated by the running container.

The volumes are stored on the host, independent of the container life cycle. This allows users to back up data and share file systems between containers easily.

Path: /var/lib/docker/volume/

### **What do you know about the Docker system prune?**

### It’s a command used to remove all stopped containers, unused networks, build caches, and dangling images. Prune is one of the most useful commands in Docker.

The syntax is:  $ docker system prune

* **What is Multi-stage Dockerfile ?**

Basically Multi-stage dockerfile is**use to minimize the size of the final container**, improve run time performance, allow for better organization of Docker commands and files.

## What is Docker Compose?

An application can consist of multiple containers running different services. It can be tedious to [start and manage containers](https://phoenixnap.com/kb/how-to-list-start-stop-docker-containers) manually, so Docker created a useful tool that helps speed up the process - Docker Compose.

### **What are the drawbacks of Docker?**

* No storage options
* Poor monitoring
* Unable to automatically reschedule inactive nodes
* Has a complicated automatic horizontal scaling setup

### **What is Docker daemon?**

Docker daemon is a service that manages Docker containers, images, storage volumes, and the network. It constantly listens to Docker API requests and processes them. A daemon can communicate with other daemons as well for the management of Docker services.

* **Difference between snapshot and AMI**
* Most basic difference between AMIs and snapshots are the type of services they are associated with. Snapshots are associated with EBS while AMIs are associated with EC2 instances.
* Snapshots are the backup of the data on EBS volumes, whereas AMIs are bootable copy of the whole EC2 instances.
* **What is version control system ?**

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modification done in the code.

* **Difference between Git And GitHub**

Git : Git is a tool for tracking changes in source code during software development.

GitHub: It is web-based git repository hosting service.

* **What is git checkout?**

The git checkout command is used to**switch between branches in a repository**

**> Jenkins Installation steps**

* two minimum requirements for Jenkins setup.

1. Java
2. Git (Not mandatory)

Installation Steps:

* Please launch an Amazon Linux instance using Amazon Linux AMI.
* Login to your Amazon Linux instance.
* Become root using “**sudo su -**” command.
* Update your repositories

**yum update**

* Install Java
* Get Jenkins repository using below command

**wget -O /etc/yum.repos.d/jenkins.repo**[**http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo**](http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo)

* Get Jenkins repository key

**rpm --import**[**http://pkg.jenkins-ci.org/redhat-stable/jenkins-ci.org.key**](http://pkg.jenkins-ci.org/redhat-stable/jenkins-ci.org.key)

* Install jenkins package

**yum install jenkins**

* Start jenkins and make sure it starts automatically at system startup

**service jenkins start**

**chkconfig jenkins on**

* Open your browser and navigate to http://<Elastic-IP>:8080. You will see jenkins dashboard.
* **What is Jenkins Pipeline?** Build", "Test" and "Deploy" **stages**

Jenkins Pipeline is a collection of jobs or events that brings the software from version control into the hands of the end users by using automation tools. It is used to incorporate continuous delivery in our software development workflow.

#### **Declarative Pipeline**

The Declarative pipeline is a new feature that is added to create the pipeline. This is basically written in a Jenkinsfile which can be stored into a [source code management system](https://digitalvarys.com/introduction-to-version-control-systems/) such as Git. Declarative pipelines is an ideal solution for the simple continuous delivery pipeline as it has very limited and pre-defined structure.

#### **Scripted Pipeline**

The scripted pipeline is a traditional way of writing the Jenkins pipeline as code. Ideally, Scripted pipeline is written in Jenkins file on web UI of Jenkins. Unlike Declarative pipeline, the scripted pipeline strictly uses groovy based syntax. Since this, The scripted pipeline provides huge control over the script and can manipulate the flow of script extensively. This helps developers to develop advance and complex pipeline as code.

* **To create and manage users in Jenkins**

Manage users > Create users

## Download Role-based Authorization Strategy plugin

## Go to - Manage and assign roles

* **A broken build for your project in Jenkins what can be done for it?**
* Open the output console of the broken build and check if any changes are lost within the file.
* With the on top of step still, you’re unable to try and do something, then you must clean and update the native geographic point, that may replicate the matter on native geographic point and take a look at to resolve it.

## Jenkins File : Groovy Domain specific

## Docker File : YAML ( Set of instruction )

## Ansible Playbook : YAML

## Kubernetes : YAML

## Jenkins Logs stores under var/log directory.

## What is of Executors in Jenkins?

## It decides, how many jobs can run at a time parallelly. By default, in Jenkins there are 2 executors.

**A Jenkins executor is one of the basic building blocks, which allow a build to run on a node/agent (e.g. build server).** Think of an executor as a single “process ID”, or as the basic unit of resource that Jenkins executes on your machine to run a build.

## How do I add an executor to a Jenkins build?

On the left side of the Jenkins dashboard screen, we can see our nodes and add new ones. Click on **Build Executor Status** -> New Node. Define a name for your node (something simple but descriptive). Select “Permanent Agent”.

* [**How many builds can an executor have in Jenkins?**](https://www.quora.com/How-many-builds-can-an-executor-have-in-Jenkins)

Jobs are built using executors. An executor is basically a process and both the master and the slaves can have any number of executors. If we decide to give our master two executors, that means Jenkins will be able to create two different processes at any given time in order to build two different tasks.

* **While creating pipeline what are pipeline parameter are considered?**

1. **booleanParam** type lets you define boolean parameters. You can set the default value for this type. Note that all parameters have the description argument that you can use to describe the purpose of that parameter goal.
2. **string** type allows you to define single-line strings. In addition to default-value and description, it also supports an additional argument trim to remove white spaces on both sides of the entered value.
3. **text** lets you define multi-line string texts
4. **password** parameter lets you define password input on the build page. The value of this type is not shown — concealed — on both the build page and the pipeline console.
5. **choice** is for defining a multi-choice drop-down menu with a set of pre-defined values. You can define this type with a list of values.

| [pipeline {](https://www.quora.com/profile/Joost-van-der-Griendt) |
| --- |
|  | [agent any](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [parameters {](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [**booleanParam**(name: "TEST\_BOOLEAN", defaultValue: true, description: "Sample boolean parameter")](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [**string**(name: "TEST\_STRING", defaultValue: "ssbostan", trim: true, description: "Sample string parameter")](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [**text**(name: "TEST\_TEXT", defaultValue: "Jenkins Pipeline Tutorial", description: "Sample multi-line text parameter")](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [**password**(name: "TEST\_PASSWORD", defaultValue: "SECRET", description: "Sample password parameter")](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [**choice**(name: "TEST\_CHOICE", choices: ["production", "staging", "development"], description: "Sample multi-choice parameter")](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [}](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [stages {](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [stage("Build") {](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [steps {](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [echo "Build stage."](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [echo "Hello $params.TEST\_STRING"](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [}](https://www.quora.com/profile/Joost-van-der-Griendt) |
|  | [}](https://www.quora.com/profile/Joost-van-der-Griendt) |

* **How do I run a Jenkins job on master node?**

Steps to Configure Jenkins Master and Slave Nodes

1. Click on Manage Jenkins in the left corner on the Jenkins dashboard.
2. Click on Manage Nodes.
3. Select New Node and enter the name of the node in the Node Name field.
4. Select Permanent Agent and click the OK button.
5. Enter the required information.

TCP is a connection-oriented protocol, whereas UDP is a connectionless protocol.

## HTTP : 80

## HTTPS : 443

## What Are StatefulSets?

A StatefulSet is the Kubernetes controller used to run the stateful application as containers (Pods) in the Kubernetes cluster. StatefulSets assign a sticky identity—an ordinal number starting from zero—to each Pod instead of assigning random IDs for each replica Pod. A new Pod is created by cloning the previous Pod’s data. If the previous Pod is in the pending state, then the new Pod will not be created. If you delete a Pod, it will delete the Pod in reverse order, not in random order. For example, if you had four replicas and you scaled down to three, it will delete the Pod numbered 3.

* **When to Use StatefulSets**

There are several reasons to consider using StatefulSets. Here are two examples:

1. Assume you deployed a MySQL database in the Kubernetes cluster and scaled this to three replicas, and a frontend application wants to access the MySQL cluster to read and write data. The read request will be forwarded to three Pods. However, the write request will only be forwarded to the first (primary) Pod, and the data will be synced with the other Pods. You can achieve this by using StatefulSets.
2. Deleting or scaling down a StatefulSet will not delete the volumes associated with the stateful application. This gives you your data safety. If you delete the MySQL Pod or if the MySQL Pod restarts, you can have access to the data in the same volume.

## Use multi-stage builds[🔗](https://docs.docker.com/build/building/multi-stage/#use-multi-stage-builds)

With multi-stage builds, you use multiple FROM statements in your Dockerfile. Each FROM instruction can use a different base, and each of them begins a new stage of the build. You can selectively copy artifacts from one stage to another, leaving behind everything you don’t want in the final image.

* **When we are not use multi-stage dockerfile**
* *Suppose we want to keep our Dockerfile simple and easy to read. In that case, we may not use multi-stage builds, especially in a development environment wherein developers are not used to such complexities. It is because multi-stage builds in Docker increase the physical size and logical organization of Dockerfile.*
* *The advantages of multi-stage builds are minimal when the number of containers is few. The multi-stage builds make a difference only when there are many containers like in CI/CD.*
* **What is Handlers and notifier in ansible?**

Ansible provides feature named handlers, which is like a task but will only run when called by a notifier in another task. This feature is important because your requirements for running a task may depend on the state of a service, existence of a file or a follow up tasks when state changed. We can make use of variables in notify and handlers which makes it more flexible. Ansible Handlers are used extensively in production and live environments as you always depend on something else to decide whether a task should run or not.

### **What’s the difference between virtualization and containerization?**

Virtualization is an abstract version of a physical machine, while containerization is the abstract version of an application.

The primary difference between VM and Container is that a container does not have an OS, multiple containers can run on a single OS instance. While a VM includes an OS instance, allowing multiple OS instances to run on one physical hardware. Running software in containerized environments generally uses less space and memory than running software within different VMs, since the latter requires a separate copy of the OS to run on each VM. Containers can be run within VMs.

**Docker Inspect: Gives details of container**

* What is config-map in Kubernetes ?

A ConfigMap is an API object used to store non-confidential data in key-value pairs. [Pods](https://kubernetes.io/docs/concepts/workloads/pods/) can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a [volume](https://kubernetes.io/docs/concepts/storage/volumes/).

# **Git Cherry-pick**

Cherry-picking in Git stands for applying some commit from one branch into another branch. In case you made a mistake and committed a change into the wrong branch, but do not want to merge the whole branch. You can revert the commit and apply it on another branch.

### **How to do maintenance activity on the K8 node?**

Whenever there are security patches available the Kubernetes administrator has to perform the maintenance task to apply the security patch to the running container in order to prevent it from vulnerability, which is often an unavoidable part of the administration. The following two commands are useful to safely drain the K8s node.

* kubectl cordon
* kubectl drain –ignore-daemon set

The first command moves the node to maintenance mode or makes the node unavailable, followed by kubectl drain which will finally discard the pod from the node. After the drain command is a success you can perform maintenance.

Note: If you wish to perform maintenance on a single pod following two commands can be issued in order:

* kubectl get nodes: to list all the nodes
* kubectl drain <node name>: drain a particular node
* **What is Sidecar container ?**

Sidecar containers are containers that are needed to run alongside the main container. The two containers share resources like pod storage and network interfaces. The sidecar containers can also share [storage volumes](https://kubernetes.io/docs/concepts/storage/volumes/) with the main containers, allowing the main containers to access the data in the sidecars.

* Difference bet replication controller and replica set?
* If we create pod with mismatch of labels and selector ( Both are not same) then pod get create or not/ create then where ?
* If we create pod and our scheduler get failed then how we schedule our pod ?
* If our API gateway get failed then how we find that ?
* Difference bet Kube-controller and Kube-scheduller ?
* Can we create docker image from stop docker container ?
* What is **maxSurge and** maxUnavailable in Kubernetes ?
* **Difference bet labels and selectors ?**

Labels are properties that we can attach to each item for example for their type, kind, and so on. Selectors help us in finding these items. You can think of a selector as a filter. We could label pods based on some attributes i.e. app name, front-end, back-end.

**maxSurge:** The number of pods that can be created above the desired amount of pods during an update

**maxUnavailable:** The number of pods that can be unavailable during the update process

**maxSurge is the maximum number of new pods that will be created at a time, and maxUnavailable is the maximum number of old pods that will be deleted at a time.**

Both maxSurge and maxUnavailable can be specified as either an integer (e.g. 2) or a percentage (e.g. 50%), and they cannot both be zero. When specified as an integer, it represents the actual number of pods; when specifying a percentage, that percentage of the desired number of pods is used, rounded down. For example, If you were using the default values of 25% for both maxSurge and maxUnavailable, and applied an update to a Deployment with 8 pods, then maxSurge would be 2 pods, and maxUnavailable would also be 2 pods. That means that during the update process, the following conditions will be met:

* **What is Orchestration ?**

**Orchestration** refers to the automation configuration and management of systems, applications, or services.

* **Docker Inspect**
* **Backing up a Docker Container**

First of all, in order to backup the docker container, we need the container ID of that particular container. We will use the ps command to get the container IDs of all the running containers and copy the one which we need to backup.

sudo docker ps −a

After that, copy the container ID of the docker container that you want to create a backup for. To create a snapshot of the docker container, we use the docker commit command. The format of the Docker commit command is −

sudo docker commit −p <CONTAINER\_ID> <BACKUP\_NAME>

Difference between an IAM role and an IAM user is that a role is assumable by anyone who needs it. A role does not have standard long-term credentials (like passwords) associated with it. AWS generates temporary security credentials when an IAM role is assumed.

## What is an IAM Policy?

An IAM policy is a document with a set of rules. Each IAM policy grants a specific set of permissions.

Policies are attached to IAM identities like Users, Groups, and Roles. Each IAM policy has a unique name.

## What is an IAM Role?

An IAM role is an IAM identity that you can create in your AWS account and assign specific permissions.

An IAM role is similar to an IAM because it is an IAM identity that has specific permissions associated with it. These permissions determine what the identity can and cannot do.

## What is an IAM Policy?

An IAM policy is a **document with a set of rules**. Each IAM policy grants a specific set of permissions.

Policies are attached to IAM identities like Users, Groups, and Roles. Each IAM policy has a unique name.

There are two types of policies in your AWS account:

* **Managed policies**: These policies can be reused and attached to multiple entities. AWS provides a lot of managed policies by default. Customers can also create their own managed policies.
* **Inline policies**: These policies are applied directly to IAM entities. However, these policies are not reusable and cannot be attached to multiple entities.
* **what is aws organizations unit?**

An organizational unit (OU) is a logical grouping of accounts in your organization, created using AWS Organizations. OUs enable you to organize your accounts into a hierarchy and make it easier for you to apply management controls.

* How will you secure Jenkins
* Ensure global security is on.

∙ Ensure that Jenkins is integrated with my company’s user directory with appropriate plugin.

∙ Ensure that matrix/Project matrix is enabled to fine tune access.

∙ Automate the process of setting rights/privileges in Jenkins with custom version-controlled script.

∙ Limit physical access to Jenkins data/folders.

∙ Periodically run security audits on same

* **What is docker image? Template**

A **Docker image** is an immutable (unchangeable) file that contains the source code, libraries, dependencies, tools, and other files needed for an application to run.

* **What is docker container?**

A **Docker container** is a virtualized run-time environment where users can isolate applications from the underlying system. These containers are compact, portable units in which you can start up an application quickly and easily.

1. **CMD** - The CMD describes the default container parameters or commands. The user can easily override the default command when you use this.
2. **ENTRYPOINT** - A container with an ENTRYPOINT is preferred when you want to define an executable. You can only override it if you use the --entrypoint flag.

* **Docker Volume**
* **Volumes** are stored in a part of the host filesystem which is *managed by Docker* (/var/lib/docker/volumes/ on Linux). Non-Docker processes should not modify this part of the filesystem. Volumes are the best way to persist data in Docker.
* **Bind mounts** may be stored *anywhere* on the host system. A Bind Mount is **a storage area (file/directory) on your local machine available inside your container**. So any changes you make to this storage space (file/directory) from the outside container will be reflected inside the docker container and vice-versa.
* **tmpfs mounts** are stored in the host system’s memory only and are never written to the host system’s filesystem.
* **Pod status in Kubernetes**

1. **Pending:**

The Pod has been accepted by the Kubernetes cluster, but one or more of the containers has not been set up and made ready to run. This includes time a Pod spends waiting to be scheduled as well as the time spent downloading container images over the network.

1. **Running:**

The Pod has been bound to a node, and all the containers have been created. At least one container is still running or is in the process of starting or restarting.

1. **Succussed**

All containers in the Pod have terminated in success and will not be restarted.

1. **Failed:**

All containers in the Pod have terminated, and at least one container has terminated in failure. That is, the container either exited with non-zero status or was terminated by the system.

* **Clone Specific branch in Git**

# git clone –b <branch-name> --single-branch <URL>

* **How to save only last 5 builds of jenkins job?**

There is one section under configure “Discard old builds”. By using plugin we can configure in our job.

How many days and how much job we want?

* **How do you implement security in Jenkins pipeline?**

By using Jenkins LDAP group we can implement security in Jenkins.2

* **Best practices of Docker**

1. Always keep dockerfile in empty directory
2. Use official image when it possible
3. User multi-stage builds

* **Difference between Docker stop and Docker kill**

KILL and STOP are two commands in docker used to stop a container from execution. A docker STOP command issues a SIGTERM signal to the main process running within the container, while KILL command issues a SIGKILL signal to the process.

* docker STOP leads to a safe exit
* docker KILL leads to an unsafe exit
* if a process doesn't exit for a STOP command within a specified timeout, docker issues a KILL command implicitly.
* **Can we have similar container in POD?**

No, we don’t have similar container in one pod.

* **I have 4 container and out off 1 container get failed then how can I check which container is failed?**

By using command **#kubectl describe pod <pod name>**

### **How do I pass Git credentials in Jenkins pipeline?**

In Jenkins job go to configure-> under that in build environment -> User secrete text or file ->