Devops Notes

What is Software Development Life Cycle (SDLC)

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to **develop, maintain, replace and alter or enhance specific software**.

The life cycle defines a methodology for improving the quality of software and the overall development process. **SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.**

- SDLC is a framework defining tasks performed at each step in the software development process.

- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

  { ISO/IEC 12207 was published on 1 August 1995 and was the first International Standard to provide a comprehensive set of life cycle processes, activities and tasks for software that is part of a larger system, and for standalone software products and services. }

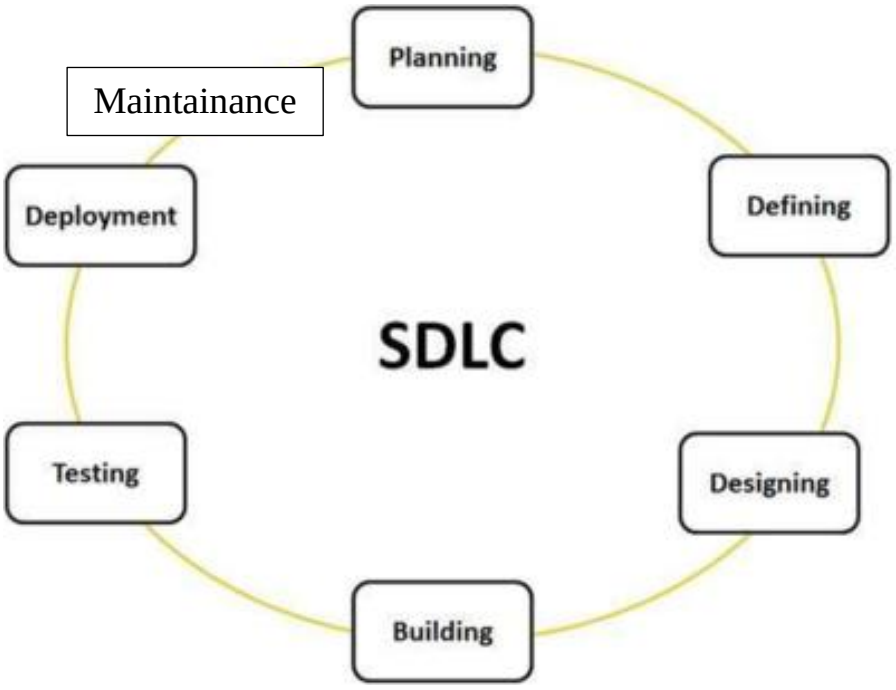
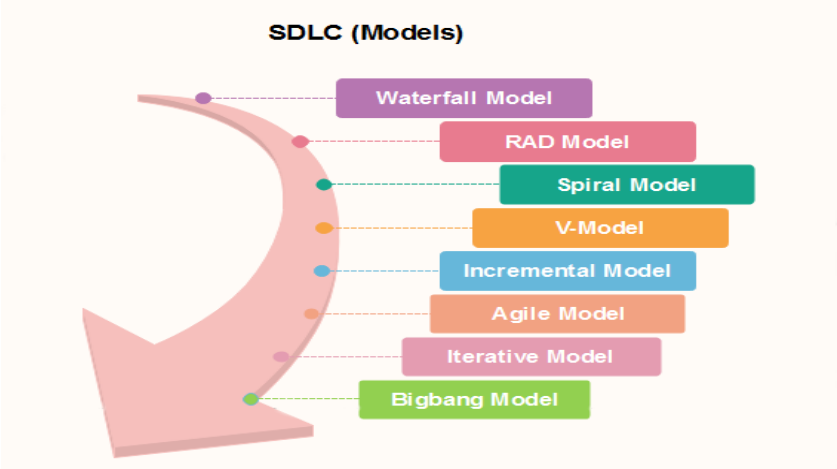


Fig. Stages of SDLC

Types of SDLC models:



Imp **1. Waterfall model:**

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use**. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.**

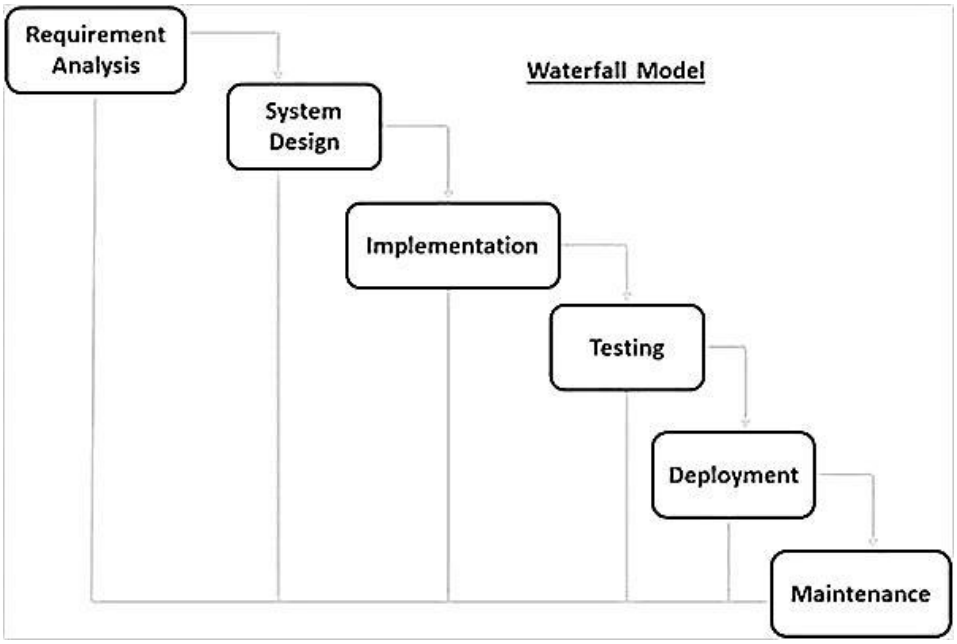


Fig. A representation of the different phases of the Waterfall Model.

2. Agile model: Agile model is a combination of iterative and incremental process models which focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Model break the product into small incremental builds. These builds are provided in iterations **(iteration means repetition of process).** Each iteration typically lasts from about one to three

weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

The advantages of the Agile Model are as follows −

- It is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

What is Devops

Defn.: "Devops is the combination of practices and tools designed to increase an organizations ability to deliver application and services faster than traditional software development processes." A process which enables the developed code to get into environment (Dev, QA, production) via CI/CD operation.

- DevOps promotes collaboration between Development and Operations team to deploy code to production faster in an automated & repeatable way.
- DevOps helps to increase organization speed to deliver applications and services. It also allows organizations to serve their customers better and compete more strongly in the market

GITHUB

It is a platform and cloud-based service for software development and version control using Git, allowing developers to store and manage their code.

What is Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency, Control & tracking of code changes over time. Git is the most commonly used Version Control System; that helps in tracking changes made to the files. Hence, we may have a record of all the changes that have been made in a file and we can also revert back to a specific version.

What is Version Control

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

What is pull request

A pull request is used to push local changes into a shared repository. It let you tell others about changes you have pushed to a branch in a repository on github.

- Pull request is always in between two branches.
- There should be difference between two branches.
- Pushing sends the recent commit history from your local repository up to github.

What is push cmd used for?

The push cmd is used to upload local repository content to a remote repository. Pushing is how you transfer commits from local repository to remote repository.

What is the difference between fetch vs push

- **Fetching imports commits to local branches.**
- **Pushing exports commits to remote branches.**

What is commit ?

Committing is the process which records changes in the repository. (Think of it as a snapshot of the current status of the project.), commits are done locally.

What is the difference between git pull and git fetch

Git Fetch is the command that tells the local repository (local machine) that there are changes available in the remote repo. without bringing the changes into the local repo(local machine).

Git pull brings the copy of the remote directory changes into the local repository (local machine).

What is Git Rebase

Rebase is designed to integrate changes from one branch onto another. Rebasing is the process of combining or moving a sequence of commits on top of a new commit. **It is the linear process of merging.**

What is the difference between git merge and rebase

Git merge and rebase are two different ways which can be used to merge changes from one branch into another.

- **Git merge** can develop a new commit that can integrate the changes from two branches. This new commit has two parent commits, one from each branch.
- **Git rebase** execute the changes from one branch onto another and explore it as though the changes were made directly on that branch.

There are two different Git rebase modes, **standard and interactive.**

A **standard mode** Git rebase automatically grabs the commits present in your current working branch and immediately applies them to the head of the passed branch.

Interactive mode lets you change different commits in the process instead of just scooping up everything and tossing it into the passed branch. If you use interactive mode, you can remove, split, or alter existing commits and clean up the history.

Note:

- Release branch is used for TEST, DEV, UAT environment.
- It is created from Master branch only

What is git branching Strategy

The main idea behind the git branching strategy is to isolate your work into different types of branches. There are Five types of branches :

1. Main 2. Develop 3. Feature 4. Release 5. Hotfix

Two primary branches in Git are: main and develop. There are three types of supporting branches : feature, release, and hotfix.

- The various types of branches make it easy to organize your work.
- The systematic development process allows for efficient testing.
- The use of release branche allows you to easily and continuously support multiple versions of production code.

What is GitLab?

Gitlab is a web-based git repository that allows users to work on free repositories. One can choose whether to keep their repository to be public or private. Gitlab is a complete DevOps platform that enables the team to build better software along with working with each other. Gitlab reduces the time and cost of development cycle.

What is git stash?

Git stash stores the changes you made to the working directory locally (inside your project's .git directory; /.git/refs/stash, to be precise) and allows you to retrieve the changes when you need them. It's handy when you need to switch between contexts. It allows you to save changes that you might need at a later stage and is the fastest way to get your working directory clean while keeping changes intact.

1. git stash : to move file to stash.
2. git stash -u or git stash --include-untracked : stash untracked files.
3. git stash -a or git stash --all : stash untracked files and ignored files.
4. git stash --patch : To stash specific files.
5. git stash list : Stashes are saved in a last-in-first-out.
6. git stash save < description > : to add a description to the stash.
7. git stash pop : used for unstash the file.
8. git stash clear : empties the stash list by removing all the stashes.
9. git stash drop < stash_id > : deletes a particular stash from the stash list.

Few cmds :

git clone < repo. link(ssh or https) > : to clone a repo. on your local machine.

git add . : adds the changes in the working dir. to the staging area. It tells git that you want to include updates to a particular file in the next commit.

git commit -m " commit message" : to take all changes that have been made locally and push them up to a remote repo.

git push origin < branch name> : to push files, dir. to remote github from local.

git status : to check the untracked files and dir.

git branch –delete < branch name >: to delete branch from local machine.

git branch < branch name > : it creates a branch or

git checkout -b < branch name > : it creates a branch as well as switch in to that branch.

git push origin -u <branch name> : to push branch to github

git push : to push specific file to the specific branch.(goto the branch in which you want to push the file > then hit the cmd git push after git add and git commit).

git pull --all : to attach remote and local both .(it downloads all the updated dir. and files which are present on remote.)

git rm -r : to delete a dir from local

git rm : to delete a file from local

git push origin --delete < branch name > : to delete branch from remote

git switch < branch name > : to switch between the branches (it switch from current branch to another branch)

Interview ques. topics:

What is Continuous integration? What is Continuous delivery?

What is Continuous deployment? What is Continuous monitoring?

What is Rebase? What is Branching Strategy?

What is environment in Devops?

Continuous Integration (CI):

It is the practice of automating the integration of code changes from multiple contributors into a single software project.

Continuous Delivery (CD): Continuous delivery is a software development practice where code changes are automatically prepared for a release to production. continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.

Continuous Deployment: It is a strategy in software development where code changes to an application are released automatically into the production environment. Once new updates pass those tests, the system pushes the updates directly to the software's users.

What is the difference between CI and CD

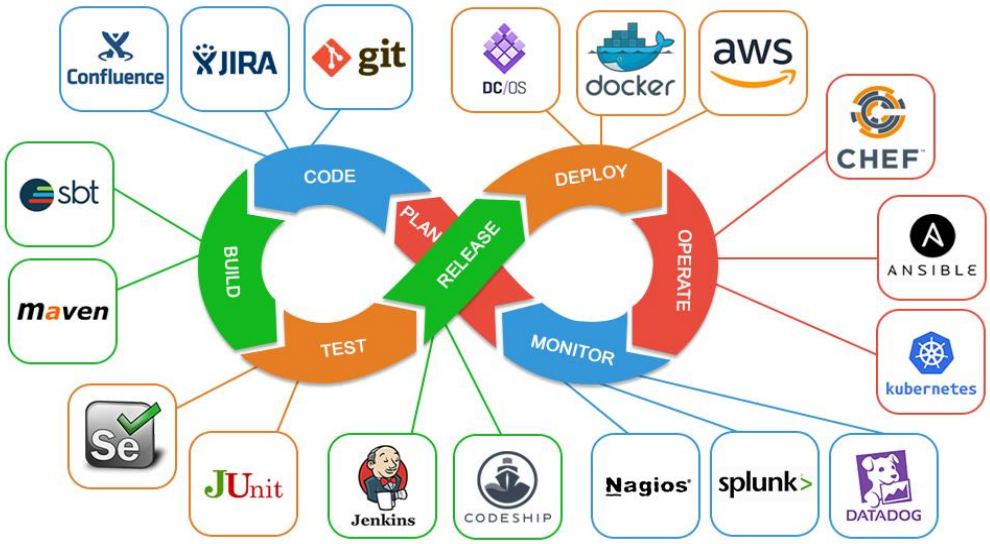
CI focuses on preparing code for release (build / test)

Whereas, CD involves the actual release of code (release / deploy).

Version control tools: (git areas)

Local Central Distributed

(working → staging → local → remote)

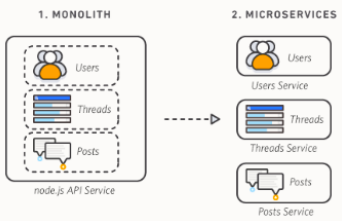


What are microservices?

Monolithic vs. Microservices Architecture

With monolithic architectures, all processes are tightly coupled and run as a single service. This means that if one process of the application experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features becomes more complex as the code base grows. This complexity limits experimentation and makes it difficult to implement new ideas. Monolithic architectures add risk for application availability because many dependent and tightly coupled processes increase the impact of a single process failure.

With a microservices architecture, an application is built as independent components that run each application process as a service. These services communicate via a well-defined interface using lightweight APIs. Services are built for business capabilities and each service performs a single function. Because they are independently run, each service can be updated, deployed, and scaled to meet demand for specific functions of an application.



Breaking a monolithic application into microservices

----------XXXXXXXXXX----------

Docker

What is Docker

"Docker is a containerization tool that is used to deliver software quickly by using the concept of containerized code which helps for easy management and maintenance of applications."

or

"A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application."

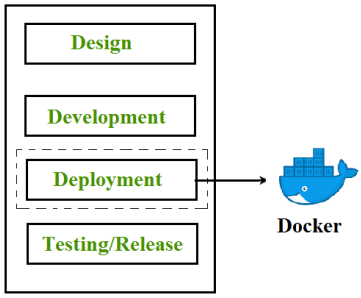


Fig: Container cycle

What is Container

Container is a unit of software that package the code & all its dependencies together. A Container is a runnable instance of an image.

- You can run one or more applications in 1 server using docker container.
- Server is replaced by container.
- Application is deployed as a container.

Docker file → Docker image → Container

Docker container run on stored image → which run on registry (repository)

What is the difference between docker and virtualization?

- Virtual machine uses a hypervisor to run a complete copy of an operating system.
- Container is processes in Linux host.
- In docker, you can deploy and scale applications on any machine and ensure your code runs consistently.
- virtual machine is a digital copy of physical machine.

(Virtual machines are considered a suitable choice in a production environment, rather than Docker containers since they run on their own OS without being a threat to the host computer.)

What is dockerfile

A Dockerfile is a text document that contains all the commands you would normally execute manually in order to build a Docker image. But Docker can build images automatically by reading the instructions from a Dockerfile.

How many commands are there in Docker?

Docker has 13 Management commands and 41 general commands. One can run the docker command using the docker CLI tool.

What is an image in Docker?

An image is a read-only template with instructions for creating a Docker container. The container is a runnable instance, while image is a read only template of it. Using the command docker pull one can get the images from the docker registry.

WORKDIR: In docker 'WORKDIR' is used to set the working directory for any subsequent instruction in docker file. It helps to define location where cmds will be executed inside the container.

CMD: It is used to specify the default command or execution that should be run when a container is started from the image. It can include argument.
It defines the default executable of Docker image.

COPY: It support only the basic copying of local files into the container.

ADD: It has some features like (local-only tar extraction and remote URL support).

ENTRYPOINT: configures the command to run when the container starts.
FROM: The FROM instruction initializes a new build stage and sets the Base Image open in new for subsequent instructions.

Alpine : This linux distribution known for being exceptionally lightweight and secure also reliable.

What is multi-stage dockerfile?

A multi-stage build is a process that allows you to break the steps in building a Docker image into multiple stages. This will enable you to create images that include only the dependencies that are necessary for the desired functionality of the final application, cutting down on both time and space.

[code pull → code build → then deploy]

{ **Pom.xml file :** it contains all the dependencies and libraries which are required for particular image. POM stands for Project Object Model. It is fundamental unit of work in Maven. It is an XML file that resides in the base directory of the project as **pom.xml**. The POM contains information about the project and various configuration detail used by Maven to build the project(s). }

What is docker swarm

It is a group of physical or virtual machines that are running the docker application and that have been configured to join together in a cluster. The activities of cluster are controlled by a swarm manager, and machines that have joined the cluster are referred to as nodes.

Docker Swarm converts multiple Docker instances into a single virtual host. A Docker Swarm cluster is generally composed of three items:

1. Nodes
2. Services and tasks
3. Load balancers

Nodes are instances of the Docker engine that control your cluster alongside managing the containers used to run your services and tasks.

Load balancing is also a part of Docker Swarm clusters and it is used to route requests across node.

Docker volume: In docker, volume is a persistent storage location that exists outside of the container.

[volumes are useful for storing data that needs to persist even if the container is stopped or removed.]

Or

Docker volume achives data persistency while working with data container

Docker volume Types:

- Named Volume : It is a way to persist data in docker containers. They allow you to create a named volume and mount it to a container so that the data stored in the volume is preserved even if the container is stopped or removed

- Bind mount :

Docker Network: Docker networking enables a user to link a Docker container to as many networks as requires. Docker networks are user to provide complete isolation for docker containers.

Extra :

Docker typically runs on a daemon process that's a single process or single source of truth for docker and this daemon process is bydefault is not accessible to other users..

So if you install docker as a root then, only root user has the access to docker daemon. To grant access to Jenkin user and Ubuntu user permission for docker daemon we use some cmds which is given below:

$ Sudo su –

usermod –aG docker jenkins

usermod –aG docker Ubuntu

systemctl restart docker

K8s

What is Kubernetes in simple words ?

"Kubernetes is a container orchestration system that automates the management of security, network, and network components. It allows users to deploy and manage containerized applications in a clustered environment."

Another definition:

"Kubernetes is an open-source Container Management tool that automates container deployment, container scaling, descaling, and container load balancing (also called a container orchestration tool)."

(**What is Orchestration:** Kubernetes orchestration allows you to build application services that span multiple containers, schedule containers across a cluster, scale those containers, and manage their health over time.)

- Kubernetes can group 'n' number of containers into one logical unit for managing and deploying them easily. It works brilliantly with all cloud vendors i.e. public, hybrid, and on-premises.
- Kubernetes is an open-source platform that manages Docker containers in the form of a cluster. Along with the automated deployment and scaling of containers, it provides healing by automatically restarting failed containers and rescheduling them when their hosts die. This capability improves the application's availability.

Features of Kubernetes :

1. **Automated Scheduling**– Kubernetes provides an advanced scheduler to launch containers on cluster nodes. It performs resource optimization.

2. **Self-Healing Capabilities**– It provides rescheduling, replacing, and restarting the containers which are dead.

3. **Automated Rollouts and Rollbacks**– It supports rollouts and rollbacks for the desired state of the containerized application.

4. **Horizontal Scaling and Load Balancing**– Kubernetes can scale up and scale down the application as per the requirements.

5. **Resource Utilization**– Kubernetes provides resource utilization monitoring and optimization, ensuring containers are using their resources efficiently.

6. **Support for multiple clouds and hybrid clouds**– Kubernetes can be deployed on different cloud platforms and run containerized applications across multiple clouds.

7. **Extensibility**– Kubernetes is very extensible and can be extended with custom plugins and controllers.

What is Cluster

A cluster is a set of machines, or nodes, that work together to manage and run containerized applications. It consists of **control plane components and worker nodes**, providing a unified platform for deploying, scaling, and managing containers.

Nodes in a Kubernetes cluster may be physical, or virtual. There are two types of nodes: A Control-plane-node type, which makes up the Control Plane, acts as the "brains" of the cluster **Also called as Master node**. A Worker-node also called as Data Plane, runs the actual container images (via pods).

Additional points:

- **Control plane:** The K8s **control plane manages clusters and resources such as worker nodes and pods.** The control plane receives information such as cluster activity, internal and external requests, and more. Based on these factors, the control plane moves the cluster resources from their current state to the desired state.

- **Worker nodes:** It is used to run containerized applications and handle networking to ensure that traffic between applications across the cluster and from outside of the cluster can be properly facilitated.

- **Pods: A pod is a collection of one or more Linux® containers, and is the smallest unit of a Kubernetes application.** Any given pod can be composed of multiple, tightly coupled containers or just a single container.

- **K8s objects:** Kubernetes objects are persistent entities in the Kubernetes system. **Kubernetes uses these entities to represent the state of your cluster.** Specifically, they can describe: What containerized applications are running (and on which nodes) and what are the resources available to those applications.

- **Namespaces in k8s:** Namespaces are a way to organize clusters into virtual sub-clusters — they can be helpful when different teams or projects share a Kubernetes cluster. Any number of namespaces is supported within a cluster, each logically separated from others but with the ability to communicate with each other.

What are the services and types of services in K8s?

A Kubernetes service is a logical abstraction for a deployed group of pods in a cluster (which all perform the same function). Since pods are ephemeral, a service enables a group of pods, which provide specific functions (web services, image processing, etc.) to be assigned a name and unique IP address (clusterIP).

[what is ephemeral: A special type of container that runs temporarily in an existing pod to accomplish user initiated action such as troubleshooting. **" you use ephemeral containers to inspect services rather than to build applications."]**

Types of services:

There are five types of Services:

ClusterIP (default): Internal clients send requests to a stable internal IP address.

NodePort: Clients send requests to the IP address of a node on one or more nodePort values that are specified by the Service.

LoadBalancer: Clients send requests to the IP address of a network load balancer.

ExternalName: Internal clients use the DNS name of a Service as an alias for an external DNS name.

Headless: You can use a headless service when you want a Pod grouping, but don't need a stable IP address.

Why Kubernetes?

Kubernetes helps you run your containerized applications reliably. It automatically places and balances containerized workloads and scales clusters appropriately to accommodate increasing demand and keep the system live.

Why should we move to Kubernetes?

There are various reasons to migrate a container or application to Kubernetes, but the primary reasons are to enable seamless automated app deployment, easy scalability, and efficient operability.

Why is Kubernetes better than cloud?

Kubernetes is designed to be utilized everywhere, allowing applications to run on-premises, in public clouds, or in a hybrid environment. Kubernetes provides the flexibility to alter the cluster size required to execute a service, allowing you to scale up and run applications effectively on demand.

What is the difference between minikube and Kubernetes?

A Kubernetes cluster can be deployed on either physical or virtual machines. To get started with Kubernetes development, you can use Minikube.

Minikube is a lightweight Kubernetes implementation that creates a VM on your local machine and deploys a simple cluster containing only one node.

What is the difference between resource and object in Kubernetes?

Objects represent the state of your cluster,

while Resources represent the computational power that these Objects consume.

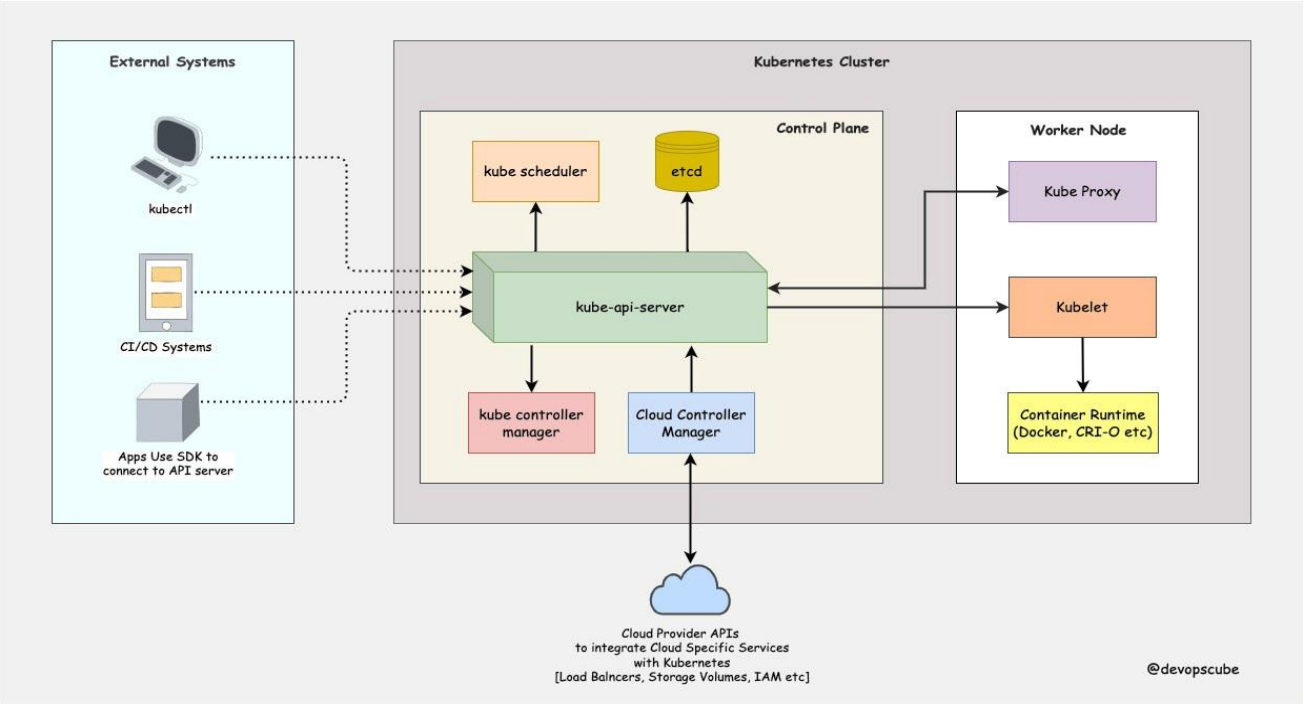
What is the purpose of K8s

" The purpose of k8s is to host your application in the form of containers in an automated fashion so that, you can easily deploy as many instances of your application as required and easily enable communication between different services within your application so there are many things involved to make this possible. "

The first and foremost thing you should understand about Kubernetes is, it is a **distributed system**. Meaning, it has multiple components spread across different servers over a network. These servers could be Virtual machines or any servers. We call it a Kubernetes cluster.

K8s consists of Cluster node (also known as Master node) and worker node.

K8s Architecture



Explanation:

Control Plane (Master node):

The control plane (Master node) is responsible for container orchestration and maintaining the desired state of the cluster. It has few components in it:

→ The **Kubernetes Master node** manages clusters and resources such as worker nodes and pods. The control plane receives information such as cluster activity, internal and external requests, and more. Based on these factors, the control plane moves the cluster resources from their current state to the desired state.

1. **kube –api--server:**

(The kube-api-server decides which container should run on which node, It basically exposes the k8s. kubernetes has to be exposed to the external world. It takes all the requests from the external world)

The API server validates and configures data for the api objects which include pods, services, replication controllers, and others. The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact (The REST API is the fundamental fabric of Kubernetes. All operations and communications between components, and external user commands are REST API calls that the API Server handles. Consequently, everything in the Kubernetes platform is treated as an API object and has a corresponding entry in the API.)

2. **etcd:**

Etcd is the database that stores the information in a key value format.

(It acts as a backing service of entire cluster. It is a key value store and store the entire k8s cluster information as objects or key value pairs inside etcd)

3. **kube-scheduler:**

It is responsible for scheduling your pods or scheduling your resources on K8s. (api server decides the information and the scheduler acts on it.**)**

A scheduler identifies the right node to place a container based on the containers resource requirements ,the worker node capacity or any other policies or constraints such as node affinity rules that are on them.

4. **kube-controller-manager:**

It ensures that the two or more pods (or all the pods) are actually running and this is taken care by replica set controller.

Types of controller manager:

1. Node Controller : It is responsible for on boarding new nodes to the cluster handling situations where nodes become unavailable or get destroyed to keep our application running.

2. Replica Controller : A Replication Controller ensures that a specified number of pod replicas are running at any one time. In other words, a Replication Controller makes sure that a pod or a homogeneous set of pods is always up and available.

3. Job Controller : The Job controller is an example of a Kubernetes built-in controller. Built-in controllers manage state by interacting with the cluster API server. Job is a Kubernetes resource that runs a Pod, or perhaps several Pods, to carry out a task and then stop.

4. Deployment Controller : A K8s deployment controller object is utilized for monitoring, management of upgrade, downgrade, and scaling of services (e.g., pods) without any disruption or downtime.

5. cloud-controller-manager : The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out the components that interact with that cloud platform from components that only interact with your cluster.

Worker node:

Kubelet:

kubelet is responsible for running your pod. It is responsible for maintaining the pod. It always look if the pod or container is running or not.

Kube proxy: It is responsible for providing the networking. Every pod we are creating it has to be some IPs and it has to be provided by default load balancing capabilities. **Kube-proxy** uses IP tables for networking related configuration.

(networking is mandatory to run pod , generating ip addresses and load balancers)

Container runtime: It is providing the execution environment for the container.

In k8s you need to have **container runtime (dockershim, containerd, cri-o)which implements k8s interface. (**It is responsible for running the container.**)**

What is Replica Set in K8s?

To avoid the accidental deletion of pods we use replicaSet.

list of cmds: **{** kubectl get rs, kubectl get pods, Kubectl get nodes, kubectl delete pod <pod name>, kubeclt delete rs <name of rs> **}**

What is Replication Controller ?

A *Replication Controller* ensures that a specified number of pod replicas are running at any one time. In other words, a Replication Controller makes sure that a pod or a homogeneous set of pods is always up and available.

What is Namespace in K8s?

Namespaces are way to organize clusters into virtual sub-cluster. They can be helpful when different teams or projects share a k8s cluster.

What are Objects?

Objects are persistent entities in the k8s systems. K8s uses these entities to represent the state of your cluster. They can describe which containerized applications are running on which node.

What are the Services and types of services of K8s ?

Services is a logical abstraction for a deployed group of pods in a cluster. Since pods are empheral, service enables a group of pods, which provide specific function to be assigned on name or unique. Ip addr.(cluster ip)

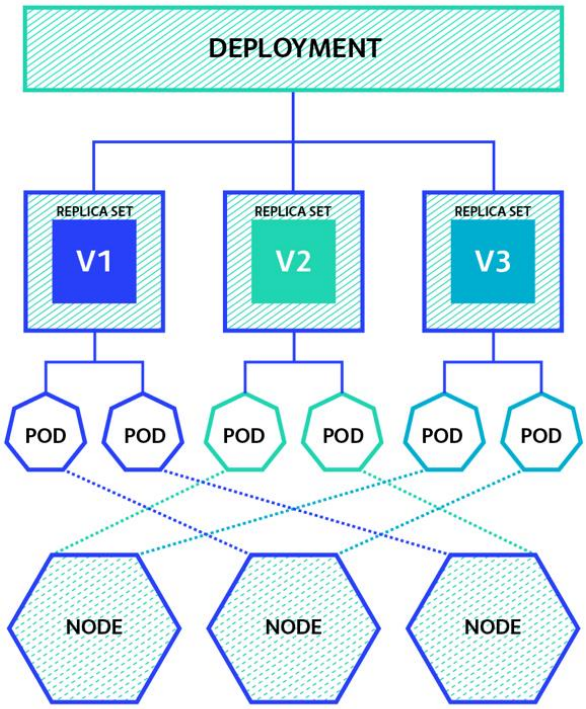


Fig: K8s Tools

What is CRD(Custome Resource Definition)?

It is a way you can add resources and enhance the capabilities of your k8s cluster.

Ex : suppose your application is deployed on k8s and if you want to add some advance load balancer capabilities using nginx load balancer but the problem is that by default k8s does not support this application because they are different kind of resources and k8s supports only deployments, services, pods, and the native k8s services so, to solve this problem and to not support this 100s of services k8s has concept called CRD. These custom resource application can do is they can write custom resource definition by there own and they can create a custom controller.

As a user once I create a custom resource this controller watches the resource and implements it what you have written in custom resource.

What is the difference between Grafana vs Prometheus?

Grafana : It is data visualization tool (It will get the data and process it then present to us in a very simple visualized format.)

Grafana will use the data to present to us in the form of charts or graphs

Prometheus : It is a data source tool (It will collect data)

Ex : If we have a website that we have deployed on a server , so we use Prometheus to collect data and logs from that website. Once the Prometheus collected the data and logs from that website then that data will be passed to Grafana and Grafana will process the data and present to us in the very proper format.(format is very easily readable and easily understandable).

What is Node Affinity ?

Node affinity is to assign particular pod on particular node.

OR

Node affinity is a set of rules used by the scheduler to determine where a pod can be placed. The rules are defined using custom labels on nodes and label selectors specified in pods. Node affinity allows a pod to specify an affinity (or anti-affinity) towards a group of nodes it can be placed on.

There are two types of node affinity rules (also known as Label types) :

Required : In required, rules must be met before a pod can be scheduled on a node.

Preferred : In preferred, rules specify that if the rule is met, the scheduler tries to enforce the rules, but does not guarantee enforcement.

What is resource allocation ?

Resource allocation is used to allocate the memory cpu utilization to the pods.

What is daemon set ?

It ensures that all (or some) nodes run a copy of a pod. It also ensures every pod should have a copy on every node.

Use case: suppose we have some library files then it should have an every pod so daemon set make sure pod has copy on all the nodes.

What is Ingress in K8s

Kubernetes Ingress is an API object that helps developers expose their applications and manage external access by providing http/s routing rules to the services within a Kubernetes cluster. **Ingress consists of an Ingress API object and the Ingress Controller.**

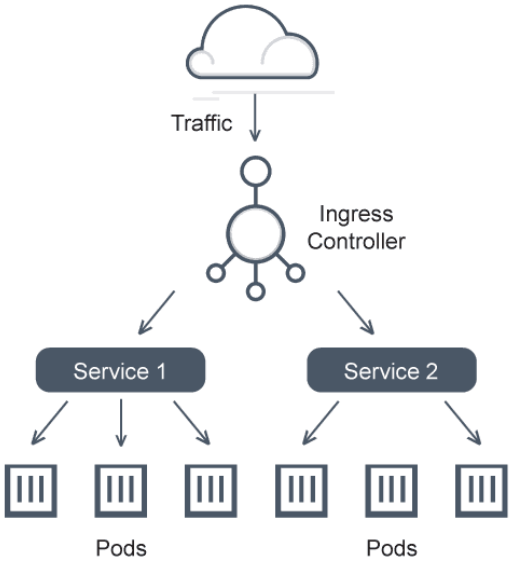


Fig : ingress service diagram

It can simplify production environments because it facilitates a simple method of establishing rules to route traffic rather than creating specialized load balancers or manually exposing each service within a node.

Ingress Resources depend on Ingress Controllers to function.

{ **What is ingress controller :** the Ingress Controller is the actual implementation of the Kubernetes Ingress API. It covers all layer four to layer seven network services and typically acts as a load balancer by distributing traffic across pods. It's also responsible for processing all Ingress resource information. }

What is the difference between ingress and egress in K8s?

Ingress : is incoming traffic to the pod

egress : is outgoing traffic from the pod.

Kubectx : is a power tool to switch between kubernetes clusters with ease as well as a utility to manage and switch between kubectl contexts.

(It manages the cluster**)**

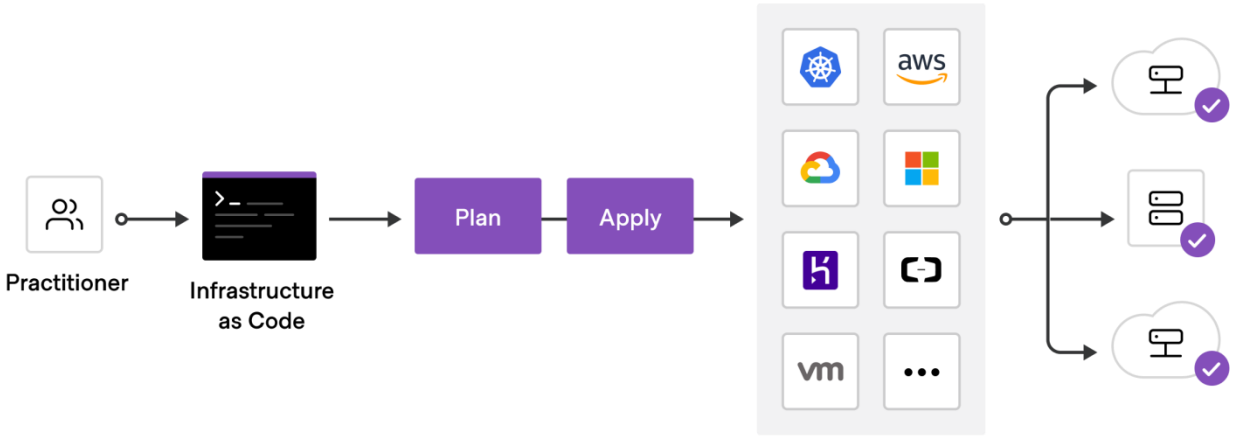
Kubens : is a power tool that allows for ease of switching between namespaces. These power tools come in handy when navigating your clusters and namespaces.

(It switch to the particular namespace**)**

Terraform

What is Terraform?

Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.



Init cmd:

"It will initialize the configuration." This cmd will read the files and whatever the cloud provider you mentioned under provide it will initialize configure to us.(It initializes provider plugins.)

Terraform plan cmd:

This cmd is used before applying or creating the infrastructure. (It will show you what exactly it is going to create). **This cmd is also called as Dry run.**

Terraform .tfstate:

Terraform uses this file to record whatever it is creating.

Terraform life cycle has 4 phases:-

1. terraform init ----- initialize
2. terraform plan ------ shows what resources are creating
3. terraform apply ------ shows which resources are applying
4. terraform destroy. ---- shows what is destroying

Interview question :

How will you create resources in multiple regions

How will you create infra in multi cloud (hybrid cloud)

Providers:

Provider is a plugin that helps Terraform to understand where it has to create the infrastructure. (It acts as a medium for terraform or entire project has to be created).

Variables:

Variables is used for parameterize. They can be used as params to pass values to the projects.

There are two types of variables in Terraform:

- Input variable
- Output variable.

Input variable: If you want to pass information to terraform then it is called input variables.

Output variable: If you want terraform to print particular value in the output.

Terraform.tfvars:

This file is used to set specific values for input variables defined in your terraform configuration i.e. **main.tf**

Purpose of .tfvars file :

- Separation of configuration from code
- Sensitive information
- Reusability
- Collaboration

Suppose if you change the name of the file from terraform.tfvars to dev.tfvars then you have to specify the name while applying the changes as :

Ex: terraform apply -var-file=dev.tfvars

Conditional_expressions (true or false)

Built in functions: length, map, lookup, etc.

Terraform.tfstate file:

The terraform state file, by default, is named terraform. tfstate and is held in the same directory where Terraform is run. It is created after running terraform apply .

If there is no state file then terraform will terraform will never come to know that it has to update infrastructure rather than creating it.

If you delete your state file, Terraform will no longer have a record of your infrastructure.

Cmd: terraform show : this cmd will give you the statefile on your terminal you don't have to go to the statefile from folder everytime.

Remote Backend:

To overcome the problem of secret credential to be get compromised we gave remote backend.

Remote backend is a type of terraform backend that stores the state file in a remote location such as cloud object storage service or data base.

[you have to create the backend.tf file in the folder of you project.]

tf.lock.hcl:

lock.hcl this name is intended to signify that it is a lock file for various items that Terraform caches in the **.terraform** subdirectory of your working directory. Terraform automatically creates or updates the dependency lock file each time you run the terraform init command.

[what is monolithic: The code with thousand of lines written as a 1 single application or file or single project could be reffered as Monolithic. **]**

Modules:

To avoid the monolithic architecture i.e where, all the resource code has been written in one single tf file and when you want to test one particular service or resource it is very difficult to avoid this thus tf introduced modules.

Advantages of using modules :

1. **Modularity**: Terraform modules allow you to break down your infrastructure configuration into smaller, self-contained components. This modularity makes it easier to manage and reason about your infrastructure because each

module handles a specific piece of functionality, such as an EC2 instance, a database, or a network configuration.

2. **Reusability**: With modules, you can create reusable templates for common infrastructure components. Instead of rewriting similar configurations for multiple projects, you can reuse modules across different Terraform projects. This reduces duplication and promotes consistency in your infrastructure.

3. **Simplified Collaboration**: Modules make it easier for teams to collaborate on infrastructure projects. Different team members can work on separate modules independently, and then these modules can be combined to build complex infrastructure deployments. This division of labor can streamline development and reduce conflicts in the codebase.

4. **Versioning and Maintenance**: Modules can have their own versioning, making it easier to manage updates and changes. When you update a module, you can increment its version, and other projects using that module can choose when to adopt the new version, helping to prevent unexpected changes in existing deployments.

5. **Abstraction**: Modules can abstract away the complexity of underlying resources. For example, an EC2 instance module can hide the details of security groups, subnets, and other configurations, allowing users to focus on high-level parameters like instance type and image ID.

6. **Testing and Validation**: Modules can be individually tested and validated, ensuring that they work correctly before being used in multiple projects. This reduces the risk of errors propagating across your infrastructure.

7. **Documentation**: Modules promote self-documentation. When you define variables, outputs, and resource dependencies within a module, it becomes clear how the module should be used, making it easier for others (or your future self) to understand and work with.

8. **Scalability**: As your infrastructure grows, modules provide a scalable approach to managing complexity. You can continue to create new modules for different components of your architecture, maintaining a clean and organized codebase.

9. **Security and Compliance**: Modules can encapsulate security and compliance best practices. For instance, you can create a module for launching EC2 instances with predefined security groups, IAM roles, and other security-related configurations, ensuring consistency and compliance across your deployments.

Provisioners: provisioners is a concept in terraform that will let you copy some particular things or execute some particular actions at the time of creation also we can use it at the time of destruction(destroy).

Types of provisioners:

- Remote exec
- Local exec
- File provisioner.

If we want to use complex project we use provisioners.

Remote exec: At the time of creating of any resource you can connect to that resource and you may be able to execute any kind of command.

Local exec: If you want to print anything on the console you use print cmd. But instead of printing everything on the console along with printing you can use local-exec and you can ask terraform copy all the output to a particular file.

File provisioner: It is used to copy the file.

Terraform workspace:

It will maintain or solve the problem of single statefile. And if you created or updated the file it will delete or recreate the resource.

{with the help of terraform workspace you can create same resource for dev env, stage or in production env.}

(terraform apply-var-file=stage.tfvars → with this cmd it will not create the new resource instead it will edit the existing resource.)

To create workspace: terraform workspace new dev ------ this cmd will create terraform.tfstate.d folder and in that folder there is again folder called dev; which we have create by using the cmd.

Similarly, for stage: terraform workspace new stage.

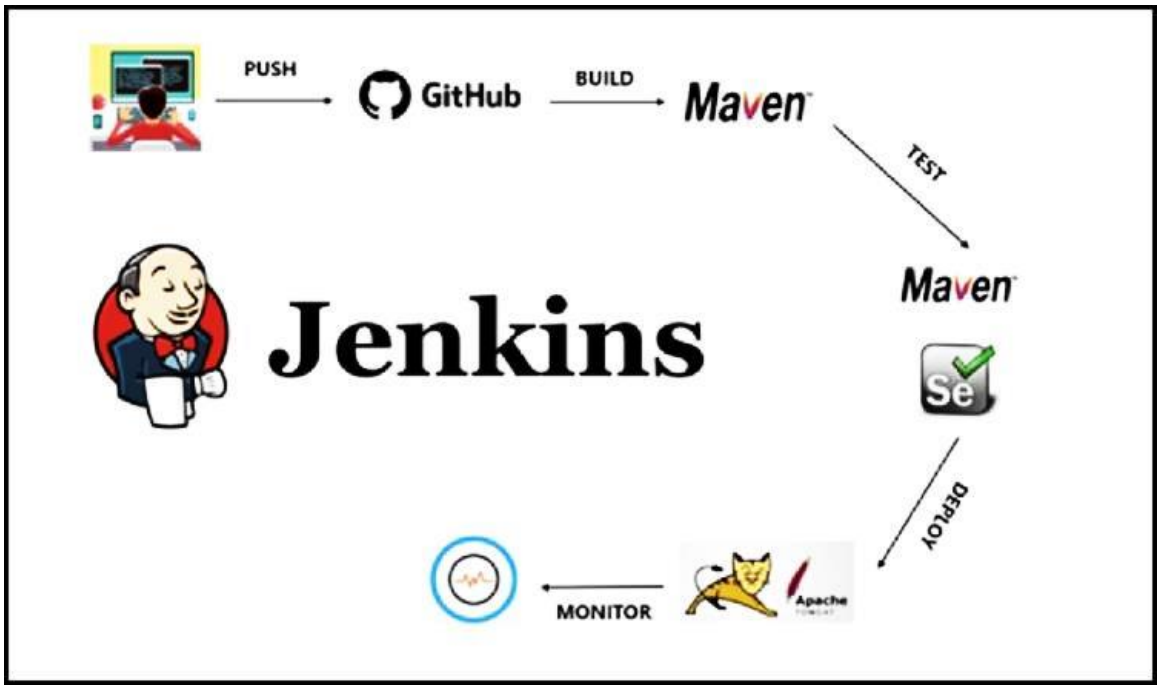
To switch between workspaces: terraform workspace select dev.

To see your current workspace : terraform workspace show.

Jenkins

Jenkins is an open-source solution comprising an automation server to enable continuous integration and continuous delivery (CI/CD), automating the various stages of software development such as build, test, and deployment.

It is a Java-based open-source automation platform with plugins designed for continuous integration.



What is Jenkins pipeline?

Pipelines are needed to run Jenkins. A pipeline is a set of steps the Jenkins server will execute to complete the CI/CD process's necessary tasks. In the context of Jenkins, a pipeline refers to a collection of jobs (or events) connected in a specific order. It is a collection of plugins that allow the creation and integration of Continuous Delivery pipelines in Jenkins.

Features:

1. A variety of plugins
2. Simpl e installation and configuration
3. Open-source availability
4. Significant extensibility
5. Server based security

Types of pipeline:

1. Scripted pipeline
2. Declarative pipeline

Scripted pipeline: Scripted Pipeline provides a lot of flexibility and control over the workflow, but it can be more complex and verbose than Declarative Pipeline.

Ex: node {
```
    stage('Build') {
        // Build the application
        sh 'mvn clean install'
    }
    stage('Test') {
        // Run the tests
        sh 'mvn test'
    }
    stage('Deploy') {
        // Deploy the application
        sh 'deploy.sh'
    }
}
```

[Here, Each stage is executed on a Jenkins node **]**

Declarative pipeline: It provides a more structured and simpler syntax for defining pipelines. Declarative Pipeline is based on the Groovy programming language, but it uses a Groovy-based DSL (Domain-Specific Language) for pipeline configuration. The main benefit of Declarative Pipeline is its readability and ease of use, as it is designed to be more intuitive and less verbose than Scripted Pipeline.

Ex: pipeline {
```
    agent any
    stages {
        stage('Build') {
            steps {
                // Build the application
                sh 'mvn clean install'
            }
        }
        stage('Test') {
            steps {
                // Run the tests
                sh 'mvn test'
            }
        }
        stage('Deploy') {
            steps {
                // Deploy the application
                sh 'deploy.sh'
            }
        }
```

```
    }
}
```

[here, we define a pipeline using the pipeline block, and we specify the agent to run the pipeline on any available node **]**

Differences between Scripted and Declarative Pipeline:

1. Syntax: The syntax for Scripted Pipeline is based on Groovy scripting language, while the syntax for Declarative Pipeline is also based on Groovy, but it uses a more structured and predefined format.
2. Flexibility: Scripted Pipeline provides more flexibility and control over the pipeline workflow, while Declarative Pipeline provides a simpler and more structured syntax.
3. Error handling: Scripted Pipeline allows for more granular error handling and recovery mechanisms, while Declarative Pipeline provides a simpler error handling mechanism that is more intuitive and easier to understand.
4. Code reuse: Scripted Pipeline allows for more code reuse and modularity, while Declarative Pipeline is designed to be more self-contained and less reliant on external scripts and libraries.
5. Readability: Declarative Pipeline is designed to be more readable and easier to understand, while Scripted Pipeline can be more complex and verbose.

Different types of Jobs in jenkins:

Job Type	Description
Freestyle Project	This is the central and the most widely used feature in Jenkins. It is an available Jenkins build job offering multiple operations. Using this option, you can build and run pipelines or scripts seamlessly.
Maven Project	If your work involves managing and building projects containing POM files, you prefer using Maven Project to build jobs in Jenkins. On choosing this option, Jenkins, by default, will pick the POM files, make configurations, and run builds.
Pipeline	Freestyle Project is often not a good option to create Jenkins Jobs. Therefore, Pipeline is the best option. Use the option Pipeline for creating Jenkins Jobs, especially when working on long-running activities.
Multi-configuration Project	If you are working on a project requiring multiple configurations, you prefer to use the Multi-configuration Project option. This option allows for making multiple configurations for testing in multiple environments.
GitHub Organization	If you click on this option, it scans the User's GitHub account for all repositories. And then, it matches markers a