1. What is ITIL?

    - ITIL (Information Technology Infrastructure Library) is a framework designed to standardize the selection, planning, delivery, maintenance and overall lifecycle of IT services within a business

    - ITIL is recognized worldwide as a best-practice approach for delivering IT services and IT service management

    - CCTA released ITIL v1 in 1989

    -
      ```
                    v2 in 2001
      ```

    -
      ```
                    v3 in 2007 - 2011 (Feedback)
      ```

    - Axelos released v4 in 2013 - 2017 - 2019 - 2020

2. Explain Software Development Life Cycle.

    - planning: requirement gathering and analysis -> SRS (Software Requirements Specification)

    - designing: designing architecture, program flow, ui/ux, designing db schema -> SDP (Software Development Plan)

    - coding: developers develop the application using required language

    - testing: testers test the application using various strategies

      ```
            - types: manual and automated
            - methods: black-box, white-box, gray-box
            - level: functional(Unit Testing) and non-
      functional(Performance)
      ```

    - deployment: deployment of application to public or production env

    - maintainance: enhancing or bug-fixing of application

    - implememted using difference models

        - waterfall, iterative, Agile

3. Explore Agile Development. What is scrum? (Day06)

    - Agile Software Development is an iterative and incremental approach to software development that emphasizes the importance of delivering a working product quickly and frequently. It involves close collaboration between the development team and the customer to ensure that the product meets their needs and expectations.

- STEPS:

    - Planning
    - Designing
    - Development
    - Testing
    - Release
    - Feedback

- Agile Software Development is widely used by software development teams and is considered to be a flexible and adaptable approach to software development that is well-suited to changing requirements and the fast pace of software development.

- Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once.

- SCRUM :

    - Scrum is a framework for agile software development that involves iterative cycles called sprints, daily stand-up meetings, and a product backlog that is prioritized by the customer.
    - There are three roles in it, and their responsibilities are:
        - Scrum Master: The scrum can set up the master team, arrange the meeting and remove obstacles for the process
        - Product owner: The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
        - Scrum Team: The team manages its work and organizes the work to complete the sprint or cycle.

4. What is Data Center?

    - Where your IT devices and applications are located
    - For a non-technical person it is the cloud where the user's files/data is stored
    - Components
        - Servers
        - Security
        - WAN
        - Storage
        - File Sharing

5. What is the functionality of a hypervisor?

    - software / application which enables hardware virtualization
    - which will enable user to create multiple VMs on a single host

6. What is Virtualization?

    - Virtualization is a technique that allows you to run multiple operating systems and applications on the same physical hardware, by creating virtual machines (VMs) that emulate the hardware resources.
    - Virtualization is the "creation of a virtual (rather than actual) version of something, such as a server, a desktop, a storage device, an operating system or network resources".
    - Types:
        - network Virtualization
        - storage Virtualization
        - data Virtualization
        - desktop Virtualization
        - application Virtualization
        - hardware Virtualization
        - platform or OS Virtualization -> Containerization -> docker / podman / rkt / LMCTFY

- What is difference between Type I, Type II virtualization(hypervisor), and containerization?
    - type 1:
        - The hypervisor runs directly on the underlying host system. It is also known as a “Native Hypervisor” or “Bare metal hypervisor”.
        - It does not require any base server operating system.
        - It has direct access to hardware resources.
        - Examples VMware ESXi, Microsoft Hyper-V hypervisor.
    - type 2:
        - A Host operating system runs on the underlying host system. It is also known as ‘Hosted Hypervisor”.
        - Such kind of hypervisors doesn’t run directly over the underlying hardware rather they run as an application in a Host system(physical machine).
        - Basically, the software is installed on an operating system. Hypervisor asks the operating system to make hardware calls.
        - An example of a Type 2 hypervisor includes VMware, Oracle Virtualbox
    - containerization:
        - When code is developed in a specific computing environment and transferred to a different environment, there is a high possibility of the code to result in bugs and errors due to missing dependencies, libraries, or any configuration setting files.
        - Containerization is the process of bundling the application code along with the libraries, configuration files, and dependencies required for the application to run cross-platform.
        - Docker is the example of a containerization platform.

7. What is cloud computing? Explain cloud service models.(Day03)

    - The practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer.
    - Is the delivery of on-demand computing resources – everything from data centers over the internet on a pay for use basis
    - Cloud computing is an umbrella term used to refer to Internet based development and services.
    - Models:
        - Service Based:

        - IaaS: operations
        - PaaS: developers
        - SaaS: end users
        - FaaS: developers -> Lambda
        - DaaS: developers or db administrators: - RDBMS (mysql, mariaDB, ms sql server, postgre, oracle - No SQL (DynamoDB)
    - Cloud Deployment Models:
        - Public
        - Private
        - Hybrid

8. What are the differences between vertical and horizontal scaling? Which one would you prefer and why?

    - Vertical => static load, sustains downtime
        - vertical scaling involves adding more power (CPU, RAM, storage, etc.) to an existing machine.
        - Vertical scaling is not only easy but also cheaper than Horizontal Scaling. It also requires less time to be fixed.
    - Horizontal => dynamic load, -> highly available -> no downtime
        - horizontal scaling involves adding more machines or nodes to a system.
        - When new server racks are added to the existing system to meet the higher expectation, it is known as horizontal scaling.

    Diff. b/w Horizontal and Vertical Scaling

9. How to create EC2 instance in AWS?

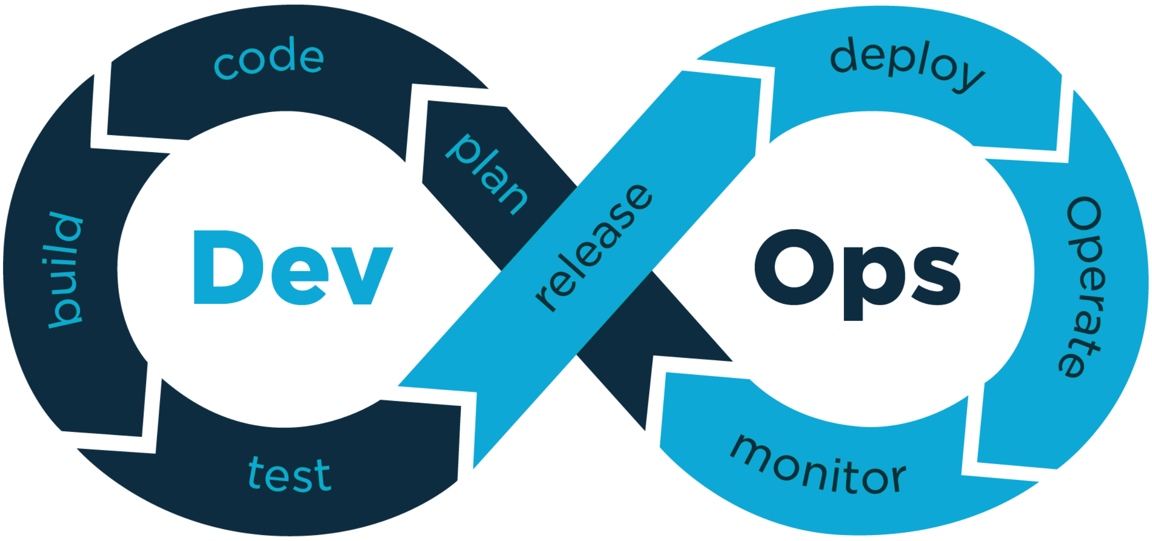
    - manual -> using Management Console
    - using IaaC
        - using standard applications -> creation: terraform, vagrant, -> configuration: puppet, chef, ansible
        - using custom script: Cloud SDK + scripting
        - using cli

10. What is use of these AWS services -

- EC2: (elastic compute cloud) used to create virtual machines (instances)
- S3: (simple storage service) used to provide storage means (object storage) -> buckets
- EBS: (elastic block storage service) used to provide storage means (block storage) -> volume
- EFS: (elastic file service) used to store and share files (file storage)
- SNS: (simple notification service) used to send notifications like SMS, Android Firebase Cloud messages (FCM), iOS Push notification (APNS)
- SES: (simple email service) used to send bulk emails
- ECS: (elastic container service) used to create docker swarm on AWS
- EKS: (elastic kubernetes service) used to setup kubernetes cluster on AWS

11. What is DevOps? Explain DevOps life cycle.(Day07-DevOps)

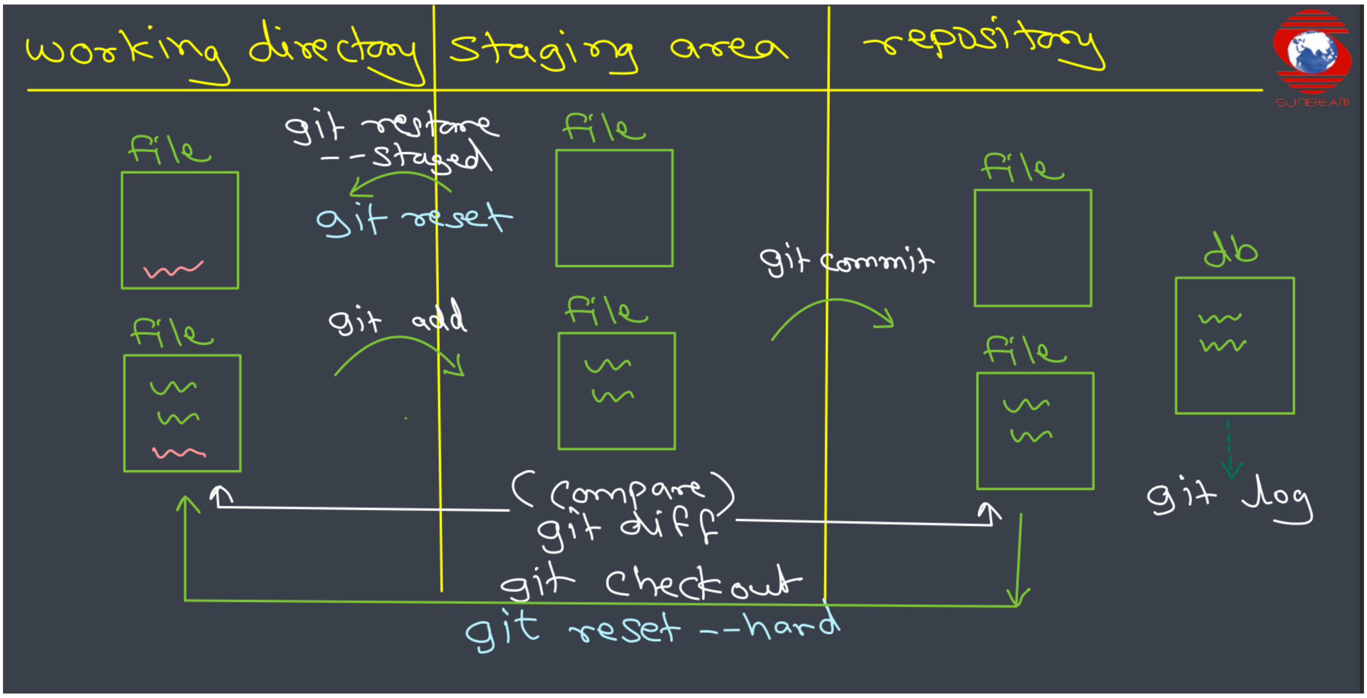
- DevOps



- what are the types of VCS or SCM?
  - Local VCS: source safe
  - Centralized VCS: svn (subversion), cvs, bazaar
  - distributed VCS: git, bitkeeper

12. What is GIT? Explain GIT workflow. (Day04)
    - Git is a distributed revision control(DVCS) and source code management system(SCM)
    - Git was initially designed and developed by Linus Torvalds for Linux kernel development
    - Git is a free software distributed under the terms of the GNU General Public License version 2
    - Git helps you keep track of code changes. Git is used to collaborate on code.

- Workflow



13. What is GIT branching? How to implement it? Write commands to create branch, merge branch, and delete branch.(Day05)

- In Git, a branch is a new/separate version of the main repository.
- Branches allow you to work on different parts of a project without impacting the main branch.
- When the work is complete, a branch can be merged with the main project.
- You can even switch between branches and work on different projects without them interfering with each other.
- Branching in Git is very lightweight and fast

```
# Create Branch
> git branch <branch_name>

# Merge Branch
> git merge <branch_name>

# Delete Branch
> git branch -d <branch_name>
```

14. What is Docker? Explain Docker architecture. (Day08)
    - Docker:
        - Docker is an open source platform that enables developers to build, deploy, run, update and manage containers— standardized, executable components that combine application source code with the operating system (OS) libraries and dependencies required to run that code in any environment

- architecture:
    - docker (client) and dockerd(Docker daemon) (server) -> REST APIs
        - Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers.
        - The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon.
        - The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.

15. What is Docker image and container? Explain life cycle of Docker container. (Day08)
    - Docker Image:
        - A Docker image is a file used to execute code in a Docker container
        - Docker images is a read-only template with set of instructions to build a Docker container
        - A Docker image contains application code, libraries, tools, dependencies and other files needed to make an application run.
        - Docker images also act as the starting point when using Docker. An image is comparable to a snapshot in virtual machine
    - Docker Container:
        - A container is a runnable instance of an image.
        - You can create, start, stop, move, or delete a container using the Docker API or CLI.
        - You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

        - It consists of:
            - application code
            - Dependencies
            - Configurations (Networking, Volumes)

16.An application has two components i.e. MySQL database and Python Web application. Explain steps to containerize this application. How to use Docker compose?

- backend service

    - python-app (server)

        - server.py
        - Dockerfile

    - database

        - db.sql: database schema
        - Dockerfile

    - docker-compose.yml

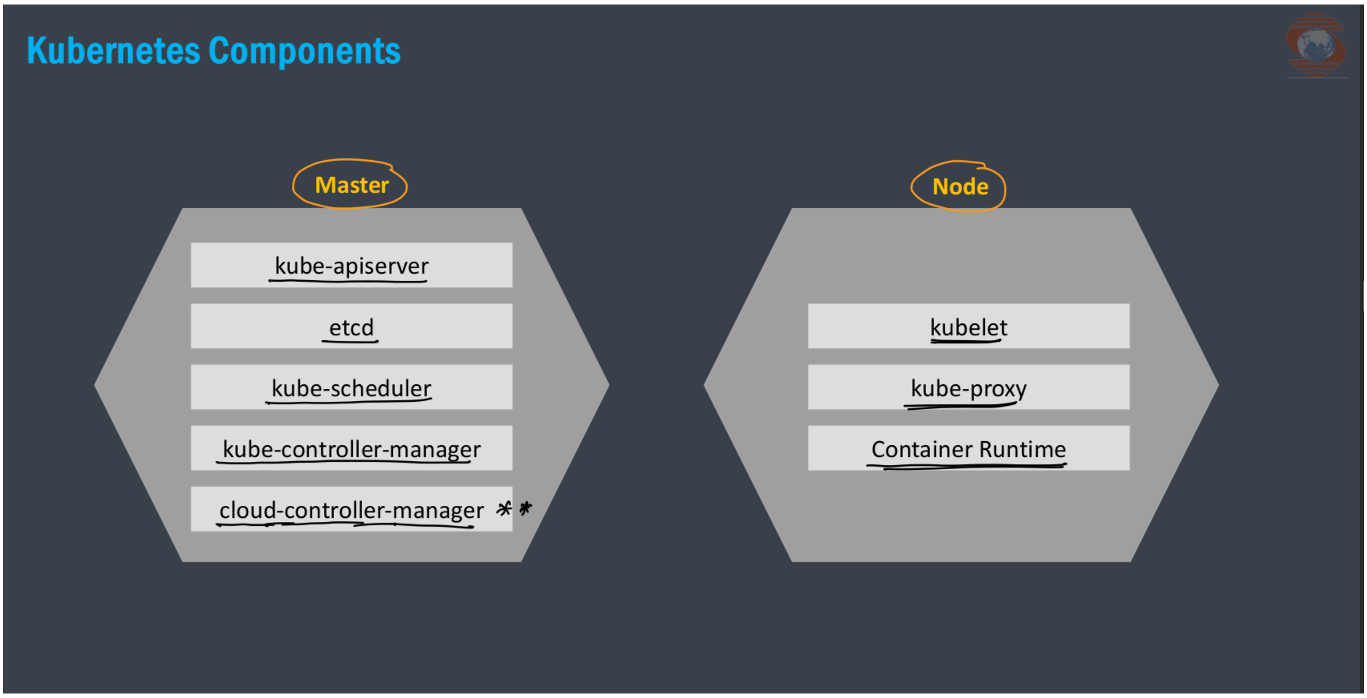
    > docker compose up

17. What is orchestration? Which tools can be used for the same.
    - Container orchestration is all about managing the lifecycles of containers, especially in large, dynamic environments.
    - Software teams use container orchestration to control and automate many tasks.
    - Tools:
        - docker swarm
        - apache marathon
        - apache mesos
        - kubernetes

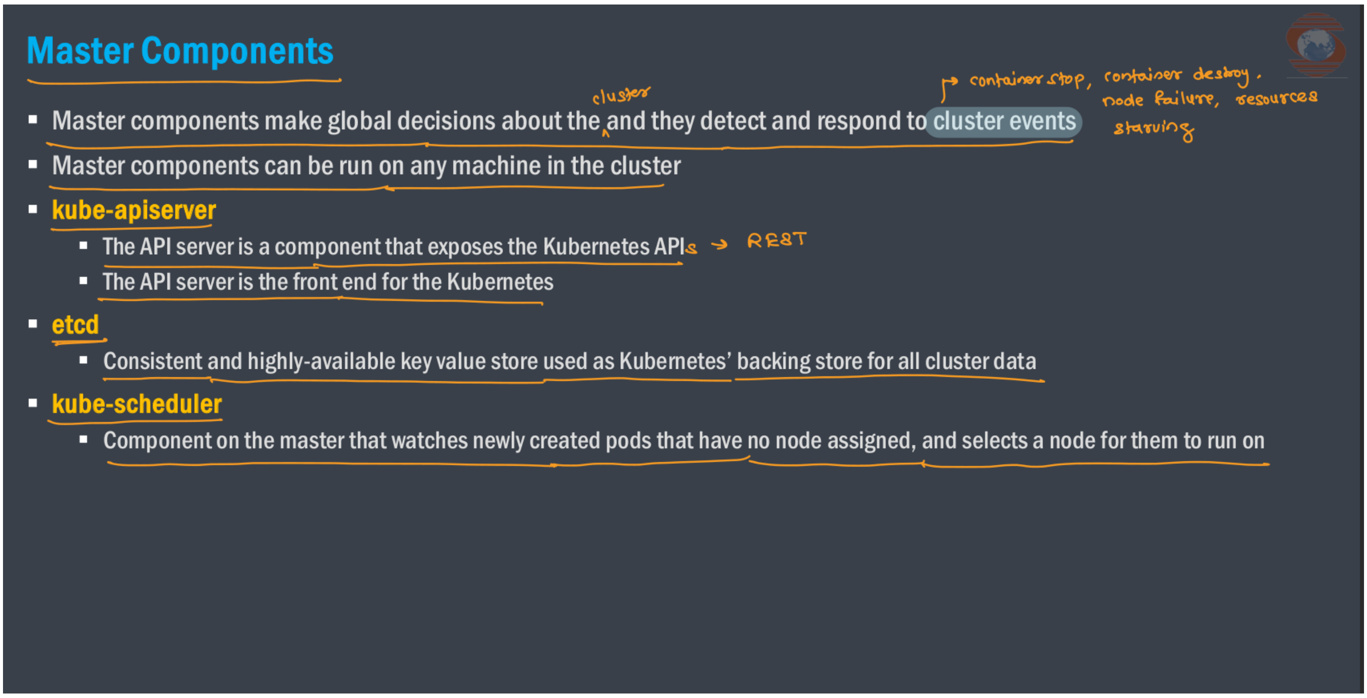
- when to choose docker swarm and kubernetes ?
    - Docker Swarm:
        - Smaller organizations with less experience with container orchestration
        - Simple applications with fewer containers
        - Organizations that need a lightweight and easy-to-use solution
    - Kubernetes:
        - Larger organizations with more experience with container orchestration
        - Complex applications with many containers
        - Organizations that need a highly scalable and feature-rich solution

18. What is Kubernetes? Explain Kubernetes architecture.(Day10)
    - Portable, extensible, open-source platform for managing containerized workloads and services
    - Facilitates both declarative configuration and automation
    - It has a large, rapidly growing ecosystem
    - Google open-sourced the Kubernetes project in 2014
    - Linux - only supported platform***

Kubernetes architecture

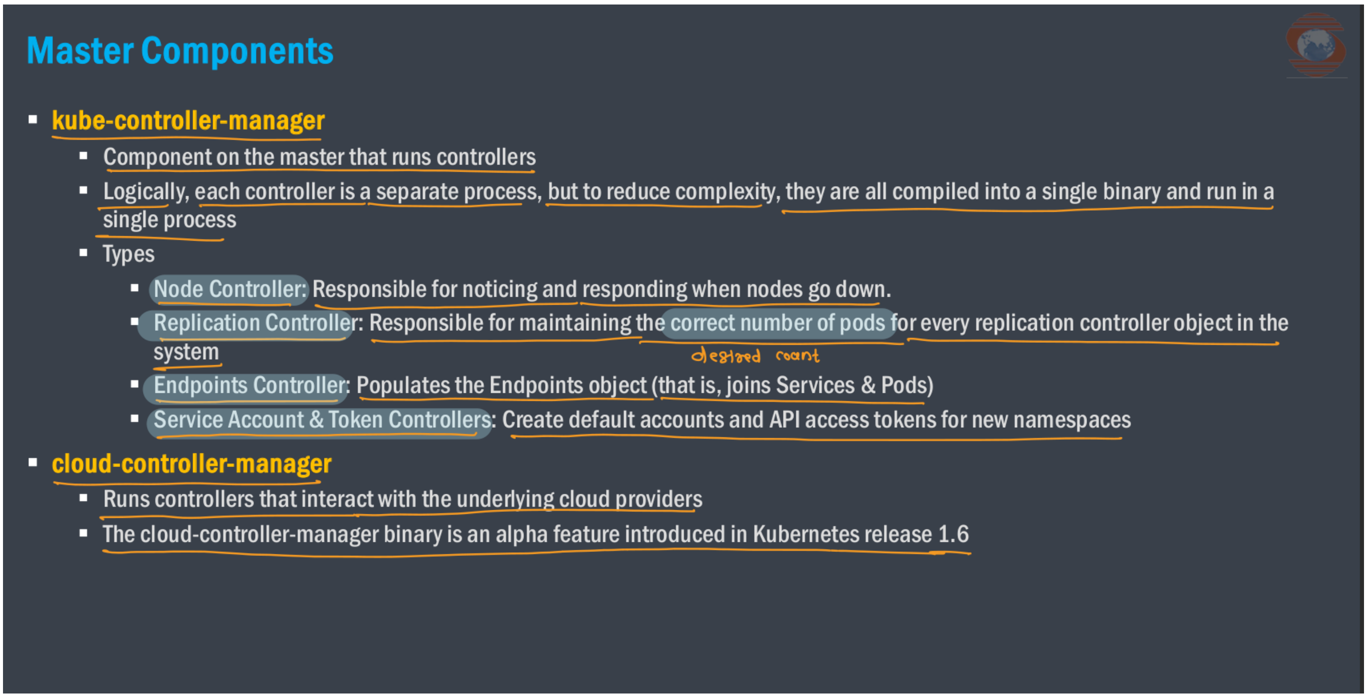


Kubernetes Components



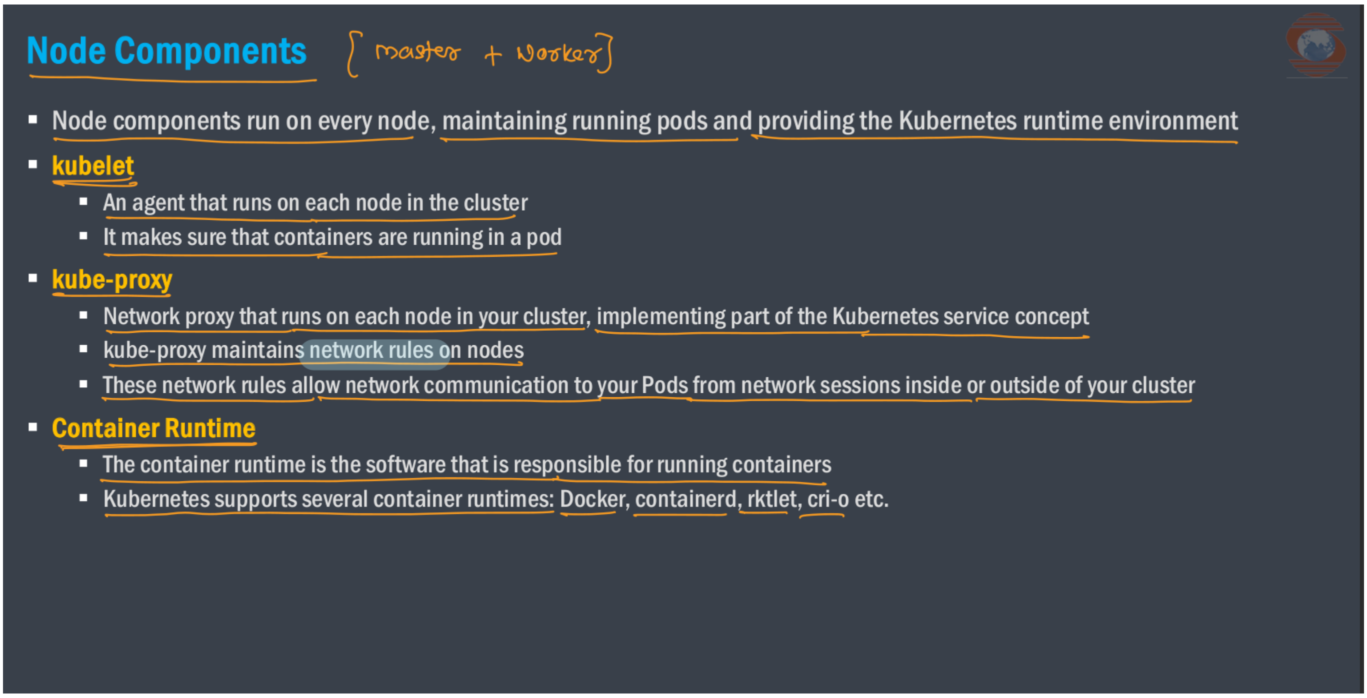
Master Components

- Master components make global decisions about the cluster and they detect and respond to cluster events → container stop, container destroy, node failure, resources starving
- Master components can be run on any machine in the cluster
- **kube-apiserver**
  - The API server is a component that exposes the Kubernetes APIs → REST
  - The API server is the front end for the Kubernetes
- **etcd**
  - Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data
- **kube-scheduler**
  - Component on the master that watches newly created pods that have no node assigned, and selects a node for them to run on



Master Components

- kube-controller-manager
  - Component on the master that runs controllers
  - Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process
  - Types
    - Node Controller: Responsible for noticing and responding when nodes go down.
    - Replication Controller: Responsible for maintaining the correct number of pods for every replication controller object in the system
      desired count
    - Endpoints Controller: Populates the Endpoints object (that is, joins Services & Pods)
    - Service Account & Token Controllers: Create default accounts and API access tokens for new namespaces
- cloud-controller-manager
  - Runs controllers that interact with the underlying cloud providers
  - The cloud-controller-manager binary is an alpha feature introduced in Kubernetes release 1.6

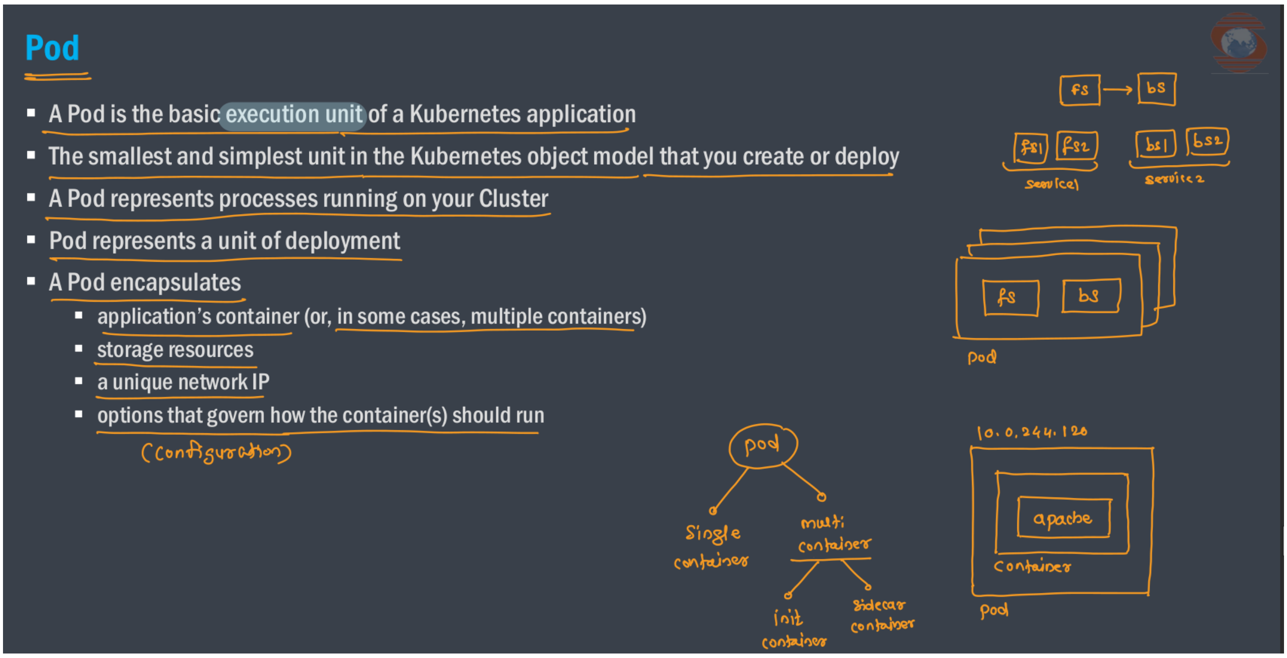


Node Components

[master + worker]

- Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment
- kubelet
  - An agent that runs on each node in the cluster
  - It makes sure that containers are running in a pod
- kube-proxy
  - Network proxy that runs on each node in your cluster, implementing part of the Kubernetes service concept
  - kube-proxy maintains network rules on nodes
  - These network rules allow network communication to your Pods from network sessions inside or outside of your cluster
- Container Runtime
  - The container runtime is the software that is responsible for running containers
  - Kubernetes supports several container runtimes: Docker, containerd, rktlet, cri-o etc.

19. What is pod? How do you scale pods?

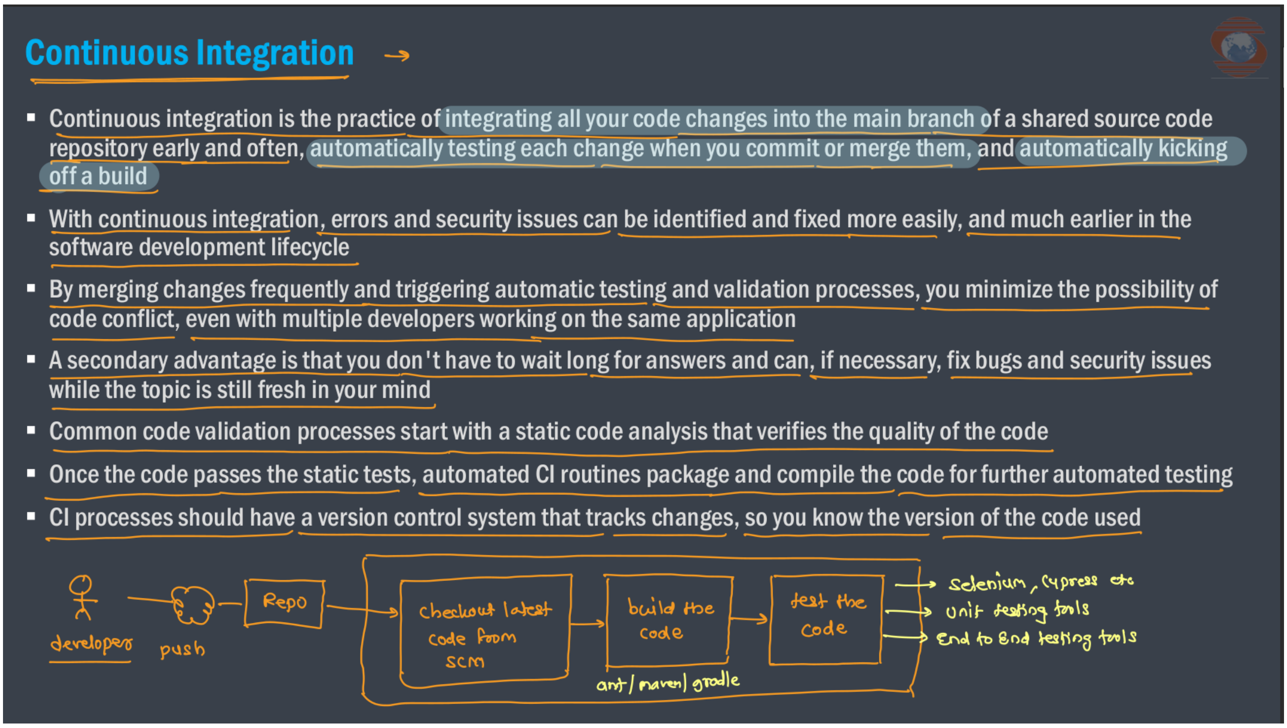


Pod

- A Pod is the basic execution unit of a Kubernetes application
- The smallest and simplest unit in the Kubernetes object model that you create or deploy
- A Pod represents processes running on your Cluster
- Pod represents a unit of deployment
- A Pod encapsulates
  - application's container (or, in some cases, multiple containers)
  - storage resources
  - a unique network IP
  - options that govern how the container(s) should run

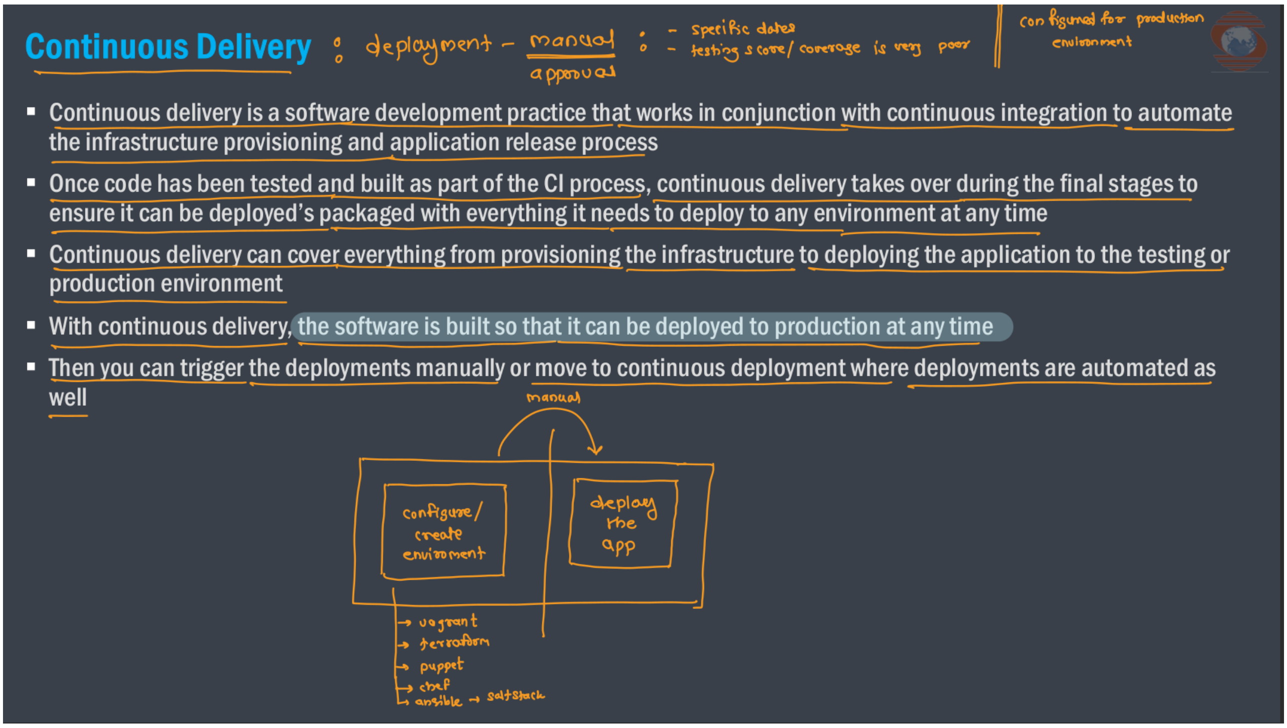
- There are two ways to scale Pods:
  - Manually:
    - You can manually scale Pods by editing the Deployment or StatefulSet that manages them. To scale up, increase the number of replicas. To scale down, decrease the number of replicas.
  - Automatically:
    - You can automatically scale Pods using a HorizontalPodAutoscaler (HPA). An HPA monitors the resource utilization of Pods and automatically scales them up or down as needed.

20. What is CI? What is CD? Which are popular tools for CI/CD pipeline? (Dat11)



Continuous Integration

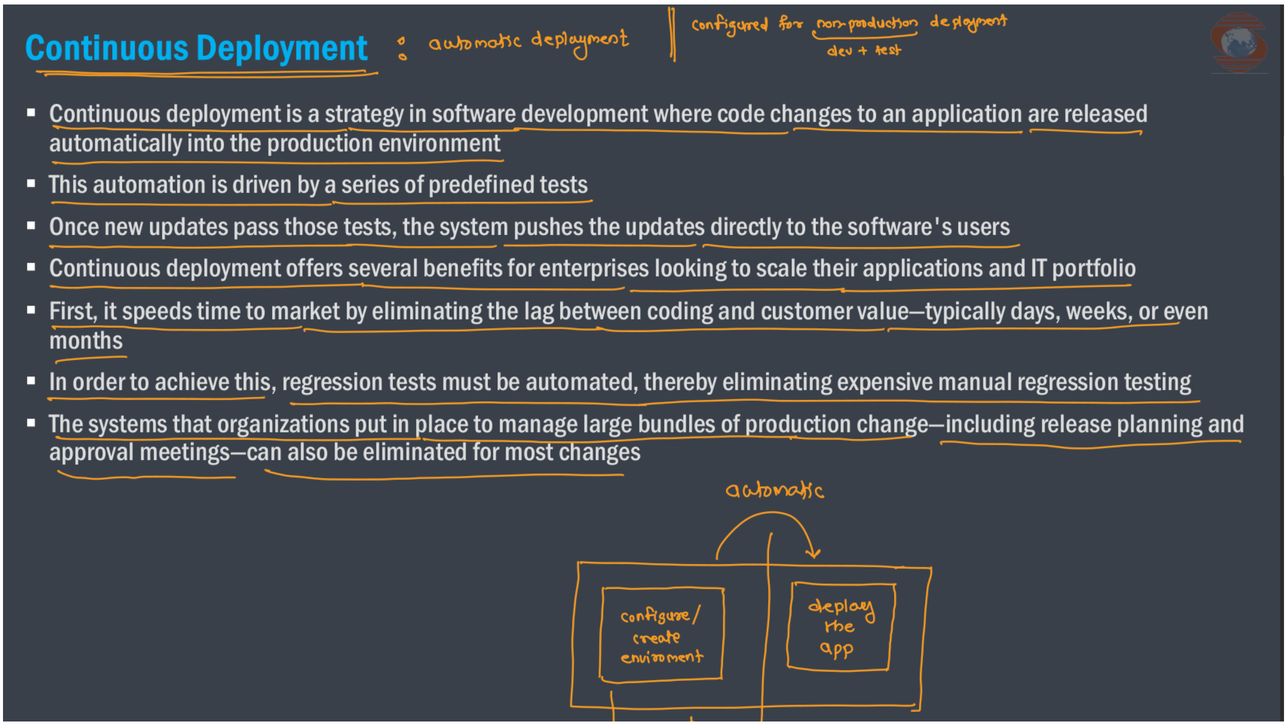
- Continuous integration is the practice of integrating all your code changes into the main branch of a shared source code repository early and often, automatically testing each change when you commit or merge them, and automatically kicking off a build
- With continuous integration, errors and security issues can be identified and fixed more easily, and much earlier in the software development lifecycle
- By merging changes frequently and triggering automatic testing and validation processes, you minimize the possibility of code conflict, even with multiple developers working on the same application
- A secondary advantage is that you don't have to wait long for answers and can, if necessary, fix bugs and security issues while the topic is still fresh in your mind
- Common code validation processes start with a static code analysis that verifies the quality of the code
- Once the code passes the static tests, automated CI routines package and compile the code for further automated testing
- CI processes should have a version control system that tracks changes, so you know the version of the code used



Continuous Delivery

: deployment - manual / approval : - specific dates - testing s core/ coverage is very poor || configured for production environment

- Continuous delivery is a software development practice that works in conjunction with continuous integration to automate the infrastructure provisioning and application release process
- Once code has been tested and built as part of the CI process, continuous delivery takes over during the final stages to ensure it can be deployed's packaged with everything it needs to deploy to any environment at any time
- Continuous delivery can cover everything from provisioning the infrastructure to deploying the application to the testing or production environment
- With continuous delivery, the software is built so that it can be deployed to production at any time
- Then you can trigger the deployments manually or move to continuous deployment where deployments are automated as well



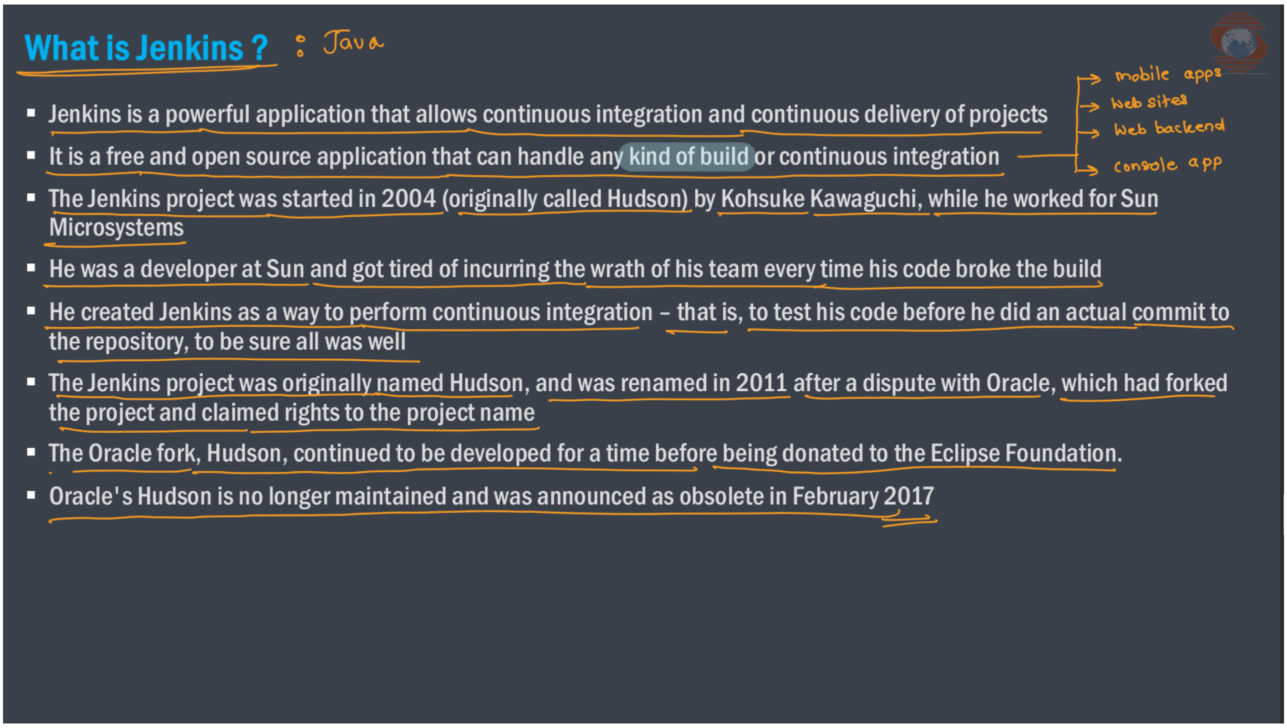
Continuous Deployment

: automatic deployment || configured for non-production deployment / dev + test

- Continuous deployment is a strategy in software development where code changes to an application are released automatically into the production environment
- This automation is driven by a series of predefined tests
- Once new updates pass those tests, the system pushes the updates directly to the software's users
- Continuous deployment offers several benefits for enterprises looking to scale their applications and IT portfolio
- First, it speeds time to market by eliminating the lag between coding and customer value—typically days, weeks, or even months
- In order to achieve this, regression tests must be automated, thereby eliminating expensive manual regression testing
- The systems that organizations put in place to manage large bundles of production change—including release planning and approval meetings—can also be eliminated for most changes

- Tools:
  - Jenkins:
    - Jenkins is an open-source Continuous Integration server that helps to achieve the Continuous Integration process (and not only) in an automated fashion.
    - Jenkins is free and is entirely written in Java
  - Bamboo
  - CircleCi
  - Travis CI

21. What is Jenkins? Explain Jenkins architecture.



What is Jenkins ?

- Jenkins is a powerful application that allows continuous integration and continuous delivery of projects
- It is a free and open source application that can handle any kind of build or continuous integration
- The Jenkins project was started in 2004 (originally called Hudson) by Kohsuke Kawaguchi, while he worked for Sun Microsystems
- He was a developer at Sun and got tired of incurring the wrath of his team every time his code broke the build
- He created Jenkins as a way to perform continuous integration – that is, to test his code before he did an actual commit to the repository, to be sure all was well
- The Jenkins project was originally named Hudson, and was renamed in 2011 after a dispute with Oracle, which had forked the project and claimed rights to the project name
- The Oracle fork, Hudson, continued to be developed for a time before being donated to the Eclipse Foundation.
- Oracle's Hudson is no longer maintained and was announced as obsolete in February 2017

- Jenkins Architecture
  - Jenkins is a continuous integration and continuous delivery (CI/CD) tool that is used to automate the building, testing, and deployment of software.
  - It has a master-slave architecture, where the master node controls the build process and the slave nodes execute the builds.
  - Master node:
    - The master node is the central component of Jenkins. It is responsible for managing the build jobs, scheduling the builds on the slave nodes, and collecting the build results.
  - Slave node:
    - The slave nodes are responsible for executing the builds. They can be any type of machine, such as a physical server, a virtual machine, or a cloud instance.

22. How would you create pipeline in Jenkins?

- Create a Jenkinsfile:
  - A Jenkinsfile is a text file that defines the stages and steps of your pipeline.
  - It is typically checked into your source code repository, so that it can be automatically triggered by changes to your code.
    - Ex:

```
pipeline {
  agent any

  stages {
    stage('Build') {
      steps {
        checkout scm
        sh 'mvn clean install'
      }
    }

    stage('Test') {
      steps {
       sh 'mvn test'
      }
    }

    stage('Deploy') {
      steps {
       echo 'Deploying to production...'
      }
    }
  }
}
```

- This Jenkinsfile defines a three-stage pipeline:
  1. Build:
     - This stage checks out the code from the source code repository and compiles it.
  2. Test:
     - This stage runs the unit tests for the application.
  3. Deploy:
     - This stage deploys the application to production.
  4. Configure Jenkins to use your Jenkinsfile.
     - To do this, you need to create a new job in Jenkins and select the Pipeline option. Then, you need to specify the path to your Jenkinsfile.
  5. Trigger the pipeline.
     - Once you have created the job, you can manually trigger it by clicking the Build Now button. You can also configure Jenkins to automatically trigger the job when changes are detected in your source code repository.

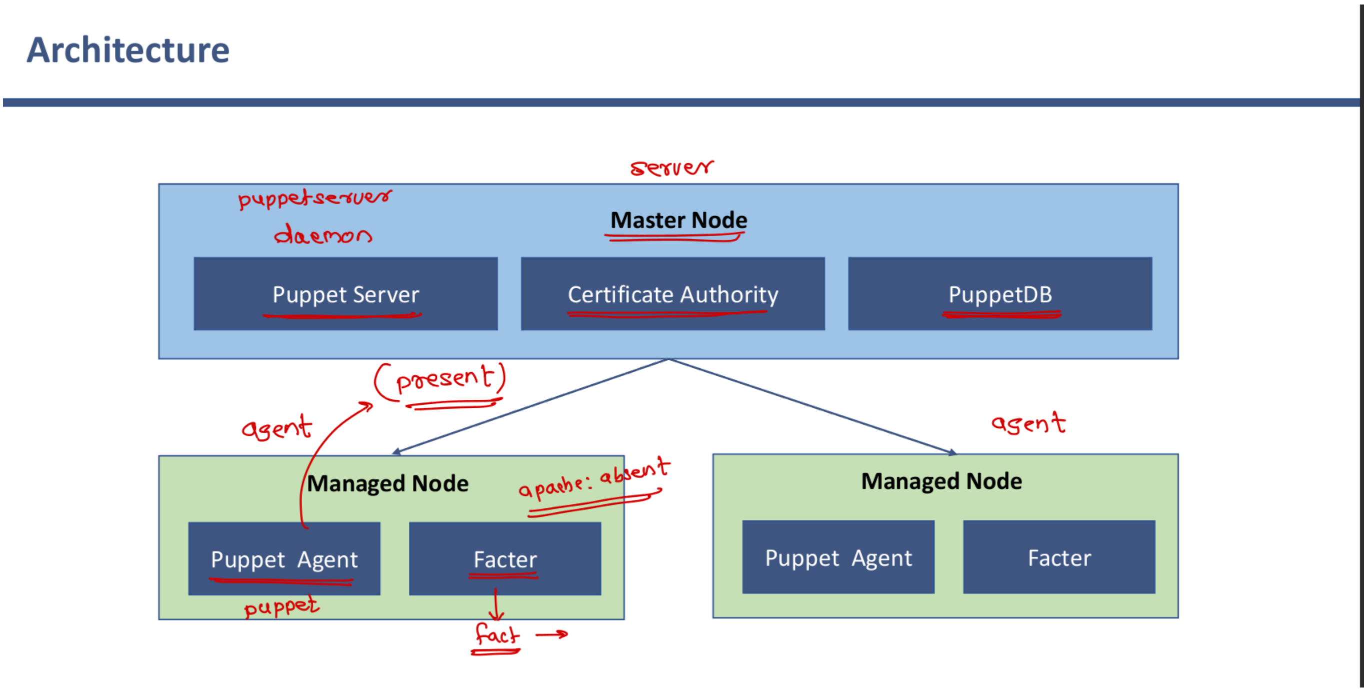
23. What's the difference between continuous integration, continuous delivery, and continuous deployment?

    - Continuous integration:
      - is the practice of merging code changes from developers into a shared repository frequently and automating the build and test process.
      - This helps to identify and fix bugs early on, before they are integrated into the main codebase.
    - Continuous delivery:
      - is the practice of building, testing, and deploying software to a production environment frequently.
      - This allows teams to deliver new features and bug fixes to customers quickly and with minimal disruption.
    - Continuous deployment:
      - is a subset of continuous delivery where changes are automatically deployed to production every time they pass the build and test stages.

- This is the most automated of the three practices and allows for the fastest deployment of new features and bug fixes.

Practice	Level of Automation
Continuous integration	Build and test automation
Continuous delivery	Build, test, and deployment automation
Continuous deployment	Full automation from build to deployment

24. What is Puppet? How it works?
    - Puppet is Continuous Configuration Tool
    - Puppet is an open-source DevOps system management tool.
    - It is used to centralize and automate the configuration management procedure.
    - This tool is developed using Ruby DSL (domain-specific language).
    - Puppet tool deploys, configures, and manages the servers.
    - With the help of automation, Puppet enables system administrators to operate easier and faster.
    - Working:
        - Puppet works by using a master-agent architecture.
        - The Puppet master server stores the desired state of the systems that it manages, and the Puppet agents on the managed systems periodically check with the master server to see if there are any changes.
        - If there are any changes, the agent makes the necessary changes to the system to achieve the desired state.



25. What is Nagios? How it works?
    - Nagios is an open-source computer-software application that monitors systems, networks, and infrastructure.
    - Nagios offers monitoring and alerting services for servers, switches, applications, and services.

  - It alerts users when things go wrong and alerts them a second time when the problem has been resolved.
  - Nagios is a popular choice for IT infrastructure monitoring because it is easy to use, scalable, and reliable.
  - Working:
    - Nagios works by using a master-agent architecture.
    - The Nagios server periodically checks the status of the Nagios agents installed on the systems that you want to monitor.
    - If the Nagios server detects a problem, it will send an alert to the system administrator.