Chapter 1:

What is devops:

Devops is a set of practices that combines software development (Dev) and IT operations (Ops) to improve collaboration, communication and efficiency in delivering software products.

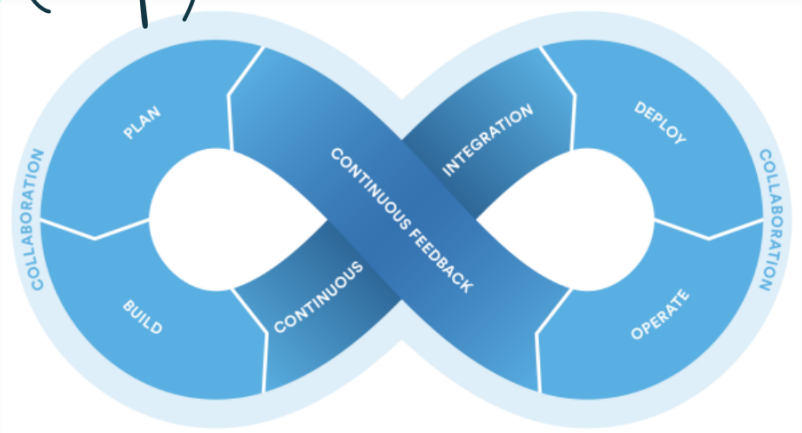
Devops = people ∪ process ∪ products

Agile = right software (business)
ATM = quality software (developers)
} Devops = right + fast software (IT Ops)

Advantages of using Devops:

- Higher speed and quality of products releases. (+CICD)
- Faster responsiveness to customer needs.
- Better working environment.

Devops phases: (7 steps)



1- Plan: business owners and software development team discuss project goals and create a plan

2- Code + Build:
- Programmers design & code the application and use tools like Git to store application code.
- Build tools like Maven and Gradle, take code from different repositories and combine them to build the complete application.

3- Test: Application is tested using automation testing tools like Selenium and JUnit to ensure software quality.

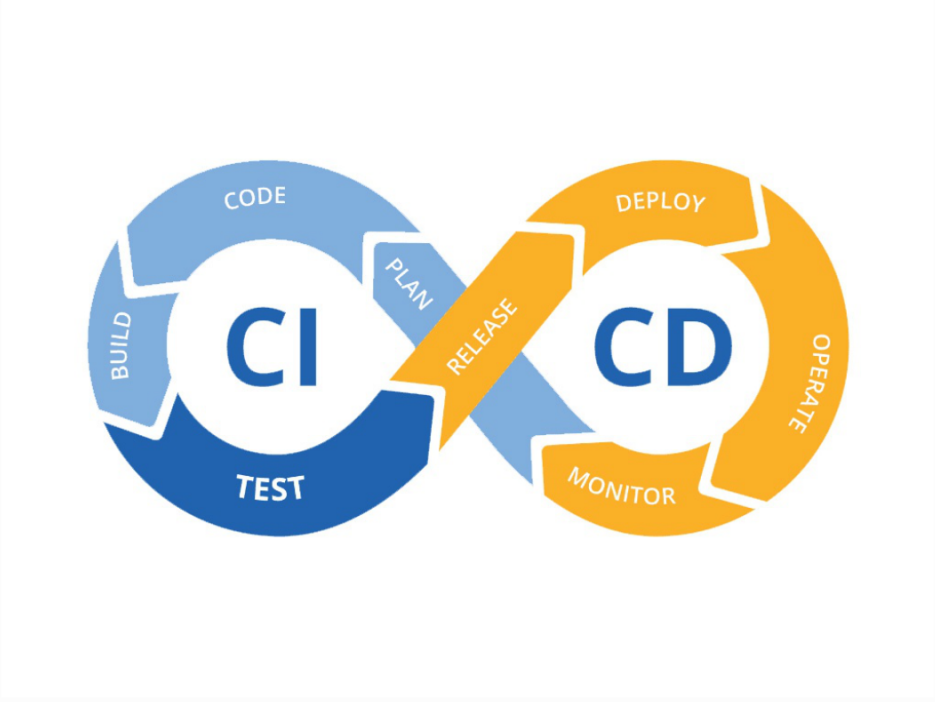
4- Integrate: When testing is complete, new features are integrated automatically to the already existing codebase. (Jenkins)

5- Deploy: Application is packaged after release and deployed from development server to production server (Docker)

6- Operate: Once software is deployed, operations team perform activities such as configuring servers and provisioning them with the required ressources.

7- Monitoring: Monitoring allows IT organization to identify specific issues of specific releases and understand the impact on end-users. (Nagios)

CI/CD:



Software deployment vs software release:

- Software deployment: the act of bringing the compiled source code into production. So that we can use that source code in production.
  Shift from one environment to another.
  Visible to selected users.
- software release: the act of enabling the feature in production
  Visible to all users

Software deployment methods:

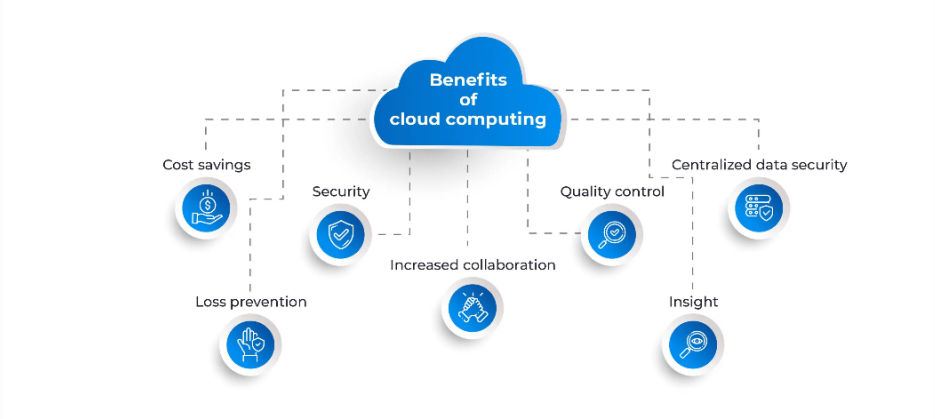
- Network or domain-based deployment: connects various IT devices to distribute and release software.
- Agent-Based deployment: utilizes an internet connection to distribute software to target devices.

Types of software deployment:

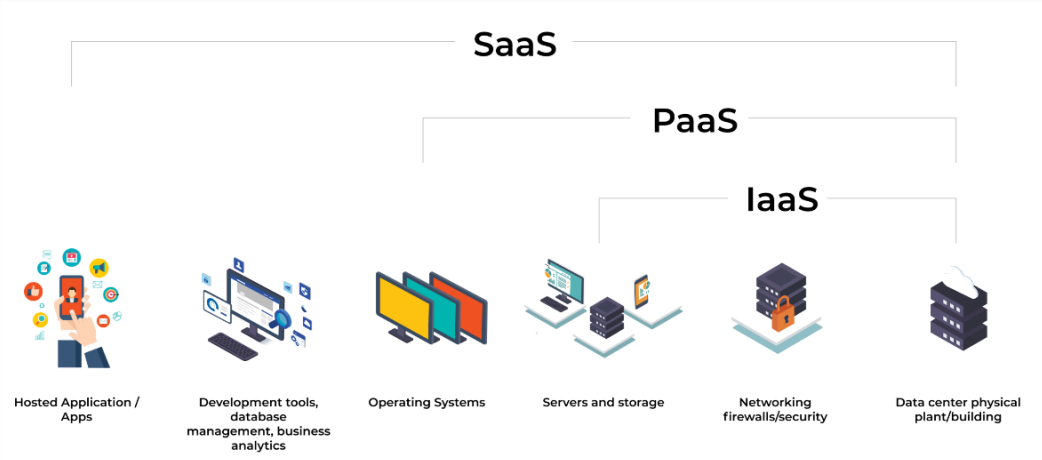
Deployment Method	Definition
Basic	Traditional method of deploying software updates directly to production.
Multi-service	Simultaneously deploying updates to multiple inter-connected services or microservices.
Rolling	Gradual deployment of updates across servers or instances, minimizing downtime.
Blue-Green	Maintaining two identical production environments, switching between them for updates.
Shadow	Deploying updates to a shadow environment to test performance and functionality.
Canary	Releasing updates to a small subset of users to asses stability
A/B testing	Simultaneously releasing two versions (A & B) to different user groups for comparision.

Chapter 2

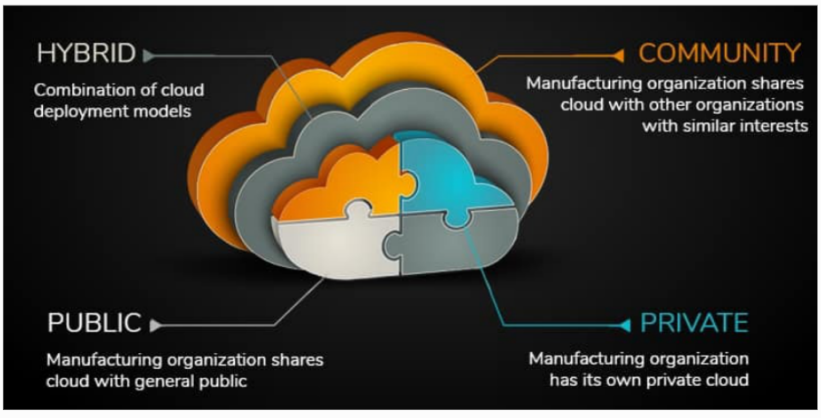
Benefits of Cloud Services:



Models of Cloud Services:



Types of Cloud deployment:



Azure virtual machines:

- Iaas
- Each virtual machine provide its own virtual hardware including CPUs, memory, hard drives, network interfaces & other devices.

Azure VM status:

- Running : running normally
- stopped : VM stopped but still consuming compute ressources within Azure
- Stopped (Deallocated) : VM stopped & not consuming compute ressources within Azure
  Default state.

Azure DevOps:

↓ services → develops software faster (CI/CD)

↳ Azure pipelines / Azure Repos...

Azure pipelines:

- lets you build, test & deploy application to Azure cloud & other supported platforms.

① Source control: the pipline starts with source control, where developers write & store their code (Ex: Git)

② Continuous Integration (CI): Automated tests are run to check if the new code integrates smoothly with the existing codebase

③ Build: Code is compiled, packaged & prepared for deployment.

④ Deployment: the code is deployed to a testing or staging environment

⑤ Continuous Deployment: Changes are automatically deployed to production.

⑥ Monitoring & feedback: metrics & logs are collected

- It supports various programming languages, platforms and cloud providers
- Classical interface: offers a visual & guided approach to defining pipelines
  YAML: provides more flexible & code-centric method with greater control & flexibility.

- Azure pipeline agents → Microsoft-hosted Agents
  → Self-Hosted Agents

	Microsoft-hosted Agents	Self-Hosted Agents
Description	Managed by Microsoft in Azure	Installed & managed by users
Location	Microsoft's cloud infrastructure	User-controlled machines
Configuration	Limited	Highly customizable
Scalability	Automated scaling	Depends on user's infrastructure
Cost	Included in Azure pipelines	User's responsibility
Usage	Standard projects	Specialized requirements
Maintenance	Managed by Microsoft	User's responsibility.

Chapter 3

Git vs Github:

- Git: . Version control system
  - . track changes
  - . History + Backup - "Snapshots"
  - . Team development
  - . Trunk based development (branches)
  - . Flexible / Local development

- Github / Gitlab: . . Hosted Git (Cloud + clone + push)

Git	Github
Software	service
Installed locally on system	hosted on Web
Command line tool	Provides graphical interface
Tool to manage different versions of edits, made to files in a git repo	Space to upload a copy of the Git Repo
Provides functionalities like Version Control System Source Code Management	Provides functionalities of Git like VCS, Source Code Management as well as adding few on its own features

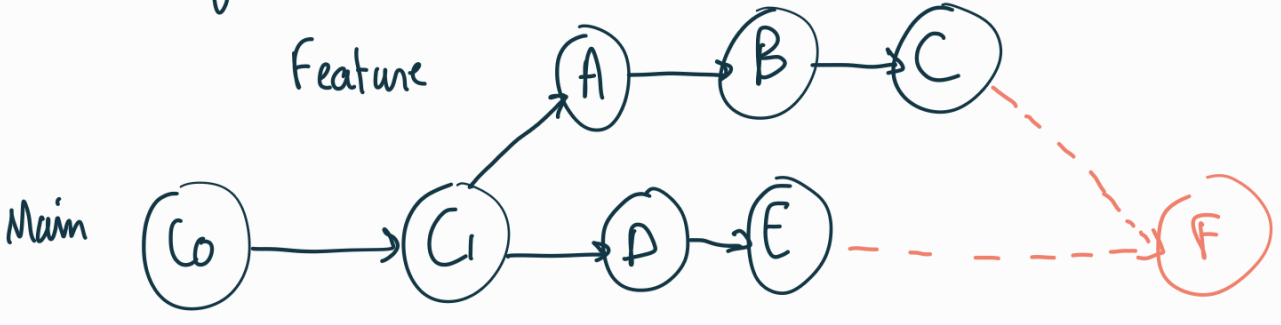
Centralized vs Distributed VCS:

	Centralized VCS	Distributed VCS
Repository	Single central repository	Each dev has local repo
Access	Requires constant ex to server	Works offline, syncs changes later
Collaboration	Limited Collab capabilities	Facilitates collaboration
Branching	Managed on server, limited support	Support to robust branching & merging
Speed	Depends on network latency & server load	faster, local operations
Offline Work	Requires server ex for most tasks	Full offline capabilities
Risk of data loss	Higher risk if central server fails	Lower risk, each developer has full copy
Examples	SVN, CVS	Git ...

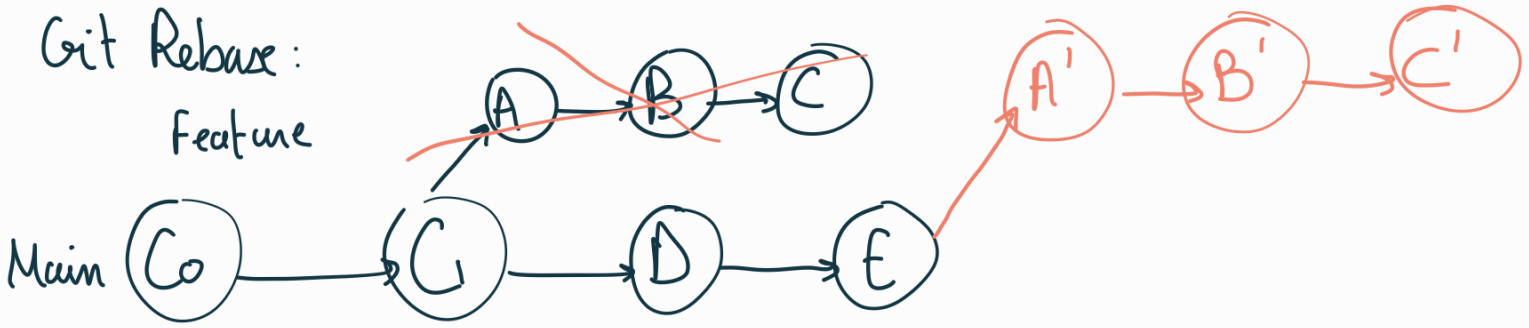
⚠ Version Control Terminology + Git Commands from pdf.

Git Merge vs Git Rebase:

Git Merge:



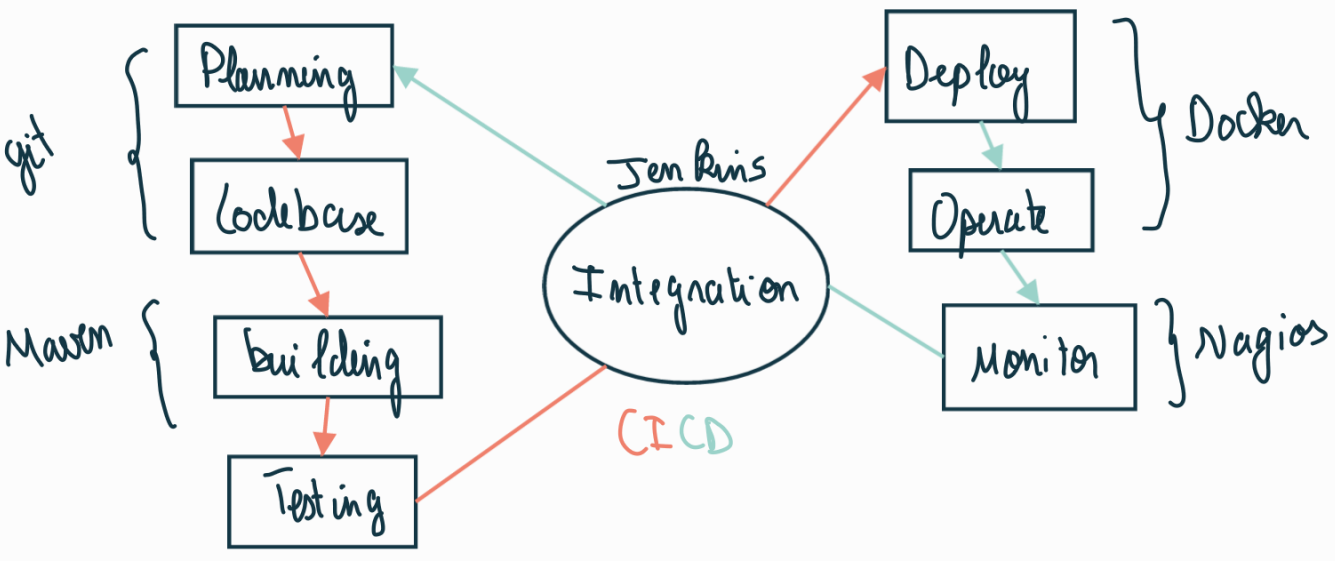
Git Rebase:



Maven:

- Project management tool (used during build process)
- Manages project dependencies, structuring projects & automating the build process
- Optimizes the build & packaging process for distribution within complex systems, promoting efficient development & deployment practices.

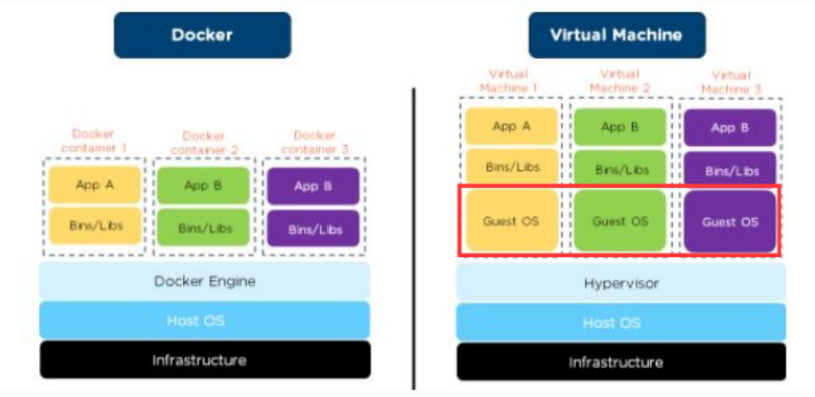
Chapter 4



Docker:

- a tool which is used to automate the deployment of applications in lightweight containers so that applications can work efficiently in ≠ env.
  (Container: software package that consists of all the dependencies required to run an application)
- Why do we use docker? The same code doesn't work on all systems due to the difference in computer environments.

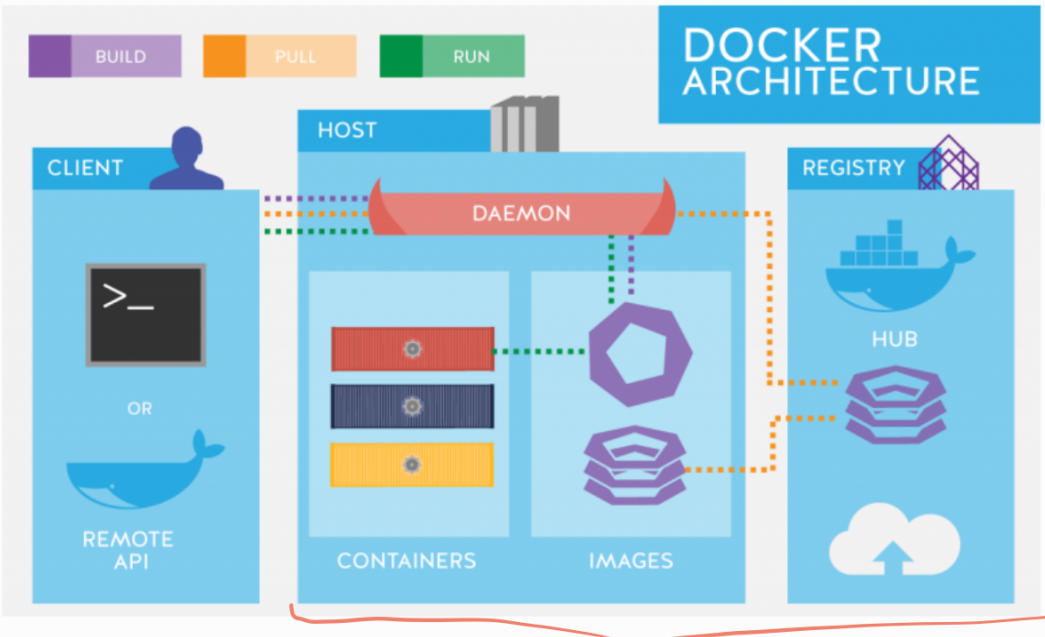
Solution → VM
→ Docker ✓
↓
provides same functionality but more lightweight



Criteria	Virtual Machine	Docker
OS support	Occupies a lot of memory space	Docker Containers occupy less space
Boot-up time	Long boot-up time	Short boot-up time
Performance	Running multiple virtual machines leads to unstable performance	Containers have a better performance as they are hosted in a single Docker engine
Scaling	Difficult to scale up	Easy to scale up
Efficiency	Low efficiency	High efficiency
Portability	Compatibility issues while porting across different platforms	Easily portable across different platforms
Space allocation	Data volumes cannot be shared	Data volumes can be shared and reused among multiple containers

VS

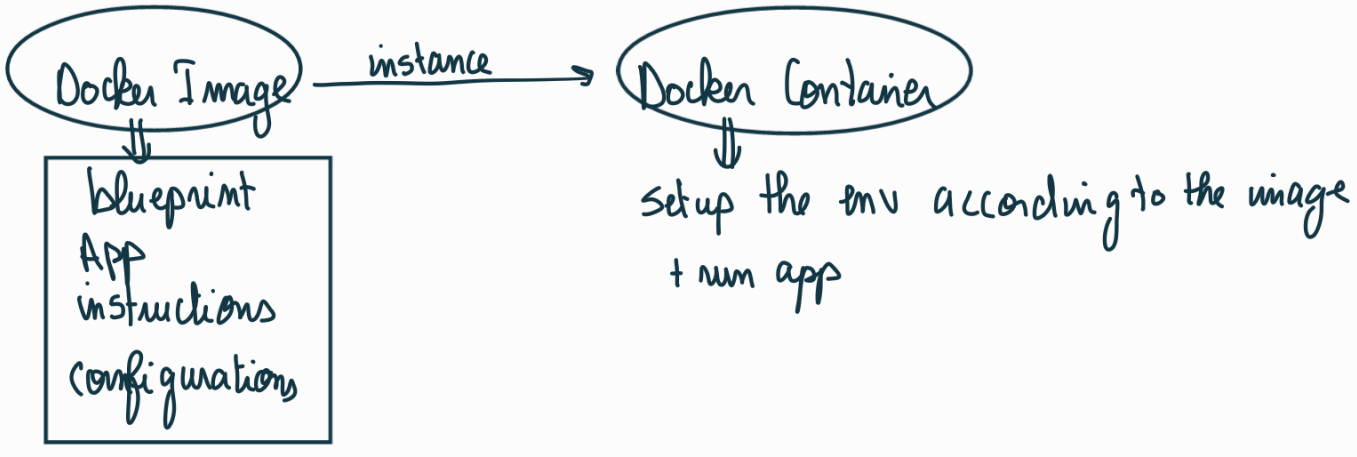
Docker Architecture:



- Docker Client is a service which runs a command. The command is translated using REST API & is sent to the Docker Daemon (server). Then the Docker Daemon checks the client request & interacts with the OS in order to create or manage containers.
- Components of Docker:
  - Docker client & server: Docker client is accessed from the terminal & a Docker Host runs the Docker Daemon & registry.
    A user can build Docker Images & run Docker containers by passing

commands from the Docker client to the Docker server.

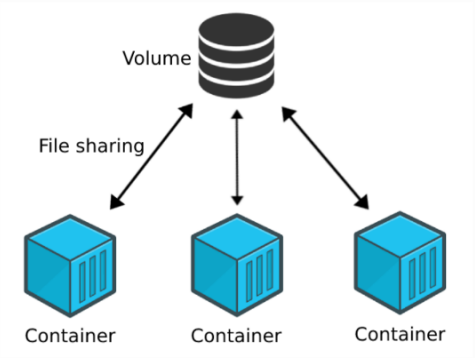
- **Docker Image:** Docker Image is a template with instructions, which is used for creating Docker Containers.
  A Docker image is built using a file called Docker file.
  Docker image is stored in a Docker Hub or in a repository
  The image also contains other configurations for the container, such as env variables, a default command to run, & other metadata.
- **Docker Container:** standalone executable software package which includes applications & their dependencies.
  Allows apps to be portable to any system running a Linux or Windows OS.
  Each Docker container starts with a Dockerfile.
  A Dockerfile that includes the instructions to build a Docker image.



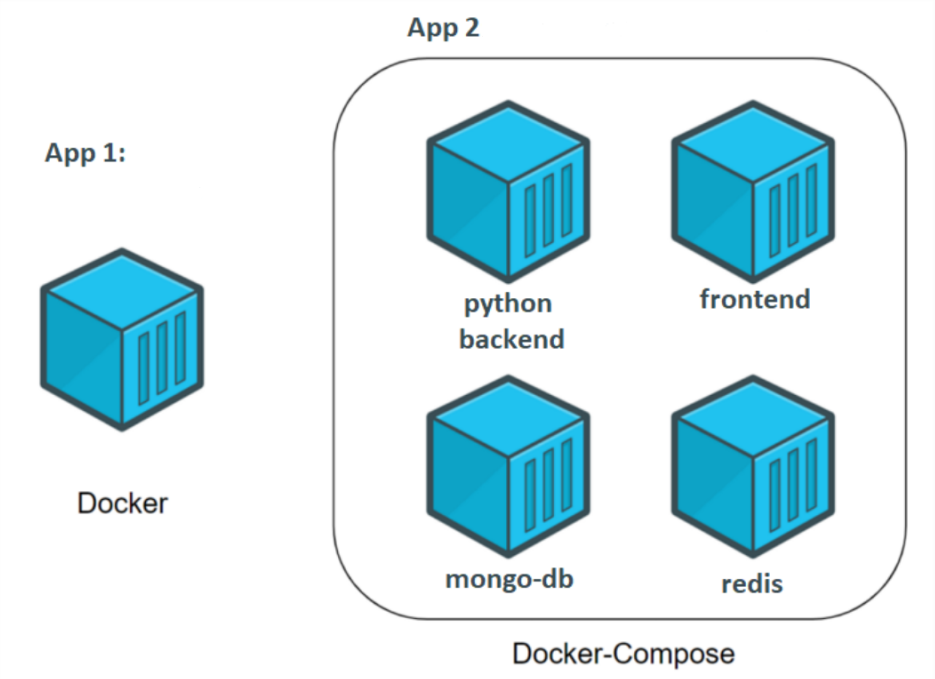
- **Docker Registry:** is an open source server-side service used for hosting and distributing images. (Give access to images for all the team)
  Images can be stored in private or public repo.

To build a container: pull Docker Image from Docker Repo
To update an image: push Docker Image to Docker Repo

Docker Volume:

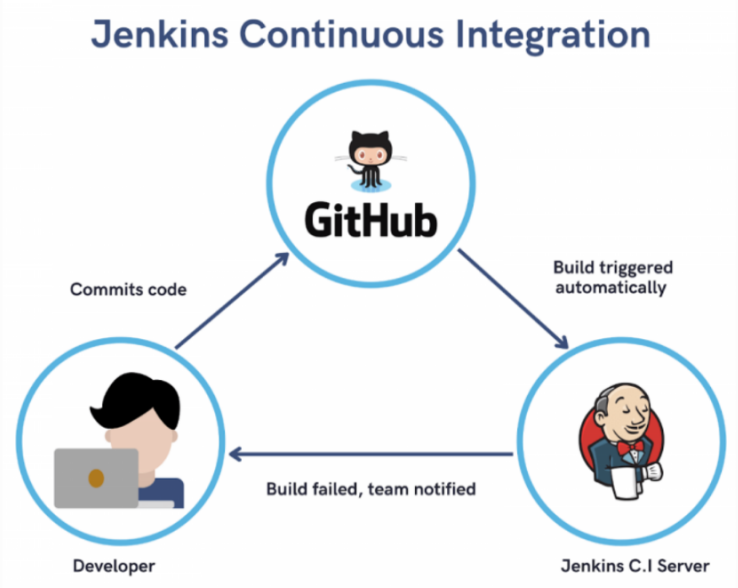


Docker Compose:



⚠ Docker Commands from pdf

Jenkins:

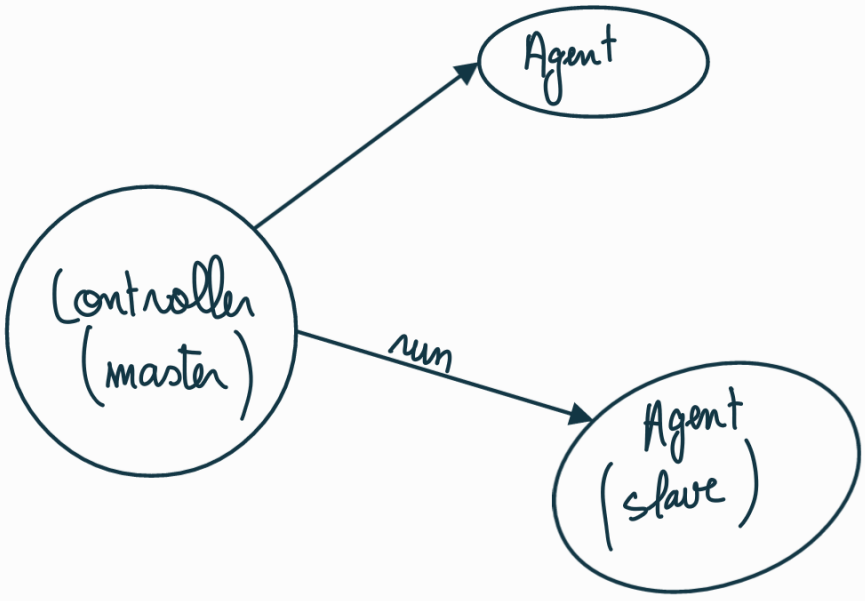


Jenkins Core Concepts:

nodes = (controller or agent

project = task

plugins



Jenkins Pipeline example:

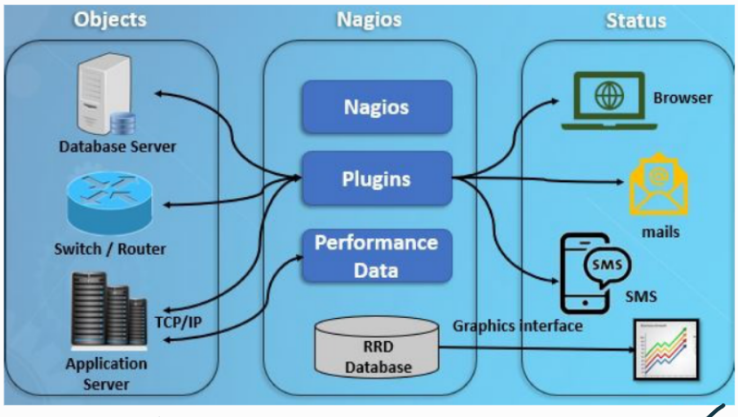
Chapter 5

Why we need continuous monitoring?

- Failure of CI/CD pipelines
- Infrastructure issues
- Application issues
- Static code analysis

Nagios:

Monitors your entire IT infrastructure to ensure systems, applications, services, and business processes are functioning properly.



what to monitor

executing checks, collecting data, scheduling

alerting & reporting

NRPE:

remotely execute Nagios plugins on other Linux/Unix machines. This allows you to monitor remote machine metrics (disk usage, CPU load, etc.)
NRPE can also communicate with Windows agent addons like NSClient++

Nagios - Checks and States:

- Active checks: run on a regular scheduled basis.
- Passive checks: sends all the passive checks in the queue to process them later.

- Soft state: When a host or service is down for a very short duration of time and its status is not known or different from previous one
  The host or the services will be tested again & again till the status is permanent.
- Hard state: When max_check_attempts is executed and status of the host or service is still not OK, then the hard state is used