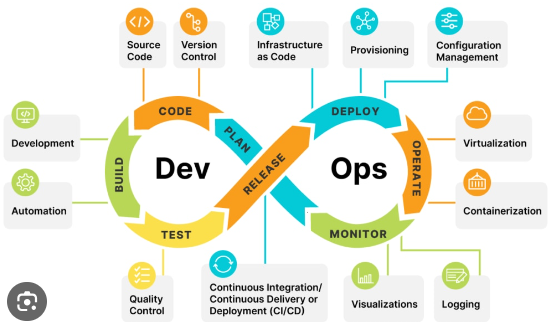
**DevOps:**

Development and Operations , responsible for bridging the gap between the dev team and IT operations team

DevOps aims to automate the process of software delivery and infrastructure changes, enabling organizations to deliver applications and services more rapidly, frequently, and reliably. It involves using various tools and technologies for continuous integration, continuous delivery (CI/CD), automated testing, infrastructure as code (IaC), and monitoring.



By adopting DevOps practices, organizations can achieve benefits such as faster time to market, higher quality software, improved collaboration between teams, increased efficiency, and better alignment with business objectives.

**Need of Devops**

The need for DevOps arises from several factors, even though software development and IT operations have been around for a long time. Here are some reasons why DevOps has become essential:

1. \*\*Increased Complexity:\*\* Modern software systems have become more complex, with distributed architectures, microservices, and cloud infrastructure. Managing these systems manually without automation can lead to inefficiencies, errors, and inconsistencies.

2. \*\*Faster Release Cycles:\*\* Businesses today demand faster delivery of software updates and new features to stay competitive. Traditional development and release cycles were too slow to meet these demands, requiring a shift towards automation and continuous delivery practices enabled by DevOps.

3. \*\*Higher Expectations for Quality:\*\* With the rise of digital experiences and online services, users have higher expectations for software quality, reliability, and performance. DevOps practices such as continuous integration and automated testing help ensure that software meets these expectations.

4. \*\*Collaboration between Teams:\*\* In the past, development and operations teams often worked in silos, leading to communication gaps, conflicts, and delays in software delivery. DevOps emphasizes collaboration and shared responsibility between development, operations, and other teams involved in the software delivery process.

5. \*\*Efficiency and Cost Savings:\*\* DevOps practices enable organizations to streamline their development and operations processes, reducing manual effort, minimizing downtime, and optimizing resource utilization. This can result in significant cost savings and improved operational efficiency.

6. \*\*Risk Management:\*\* By automating processes and implementing rigorous testing and monitoring, DevOps helps mitigate the risk of software failures, security breaches, and downtime. This is especially critical in industries such as finance, healthcare, and e-commerce, where downtime or data breaches can have severe consequences.

7. \*\*Competitive Advantage:\*\* Organizations that adopt DevOps practices can deliver software more quickly, reliably, and efficiently than their competitors. This agility and responsiveness to market changes can provide a significant competitive advantage in today's fast-paced business environment.

While software development and IT operations have existed for decades, the challenges and requirements of the modern digital landscape have necessitated the adoption of DevOps practices to meet the demands of businesses and users effectively.

Lifecycle

1. Continuous Development

Planning and Coding : Version Control

1. Continuous Testing: Testing tools like selenium and jeera

Docker containers are used simulating the test environments

The whole testing process can be automated using jenkins

1. CI: Commting changes to the code, adding on new features on top of existing features, Jenkins is used
2. CD: code is deployed to production servers
3. Monitoring:

A diagram of a lifecycle

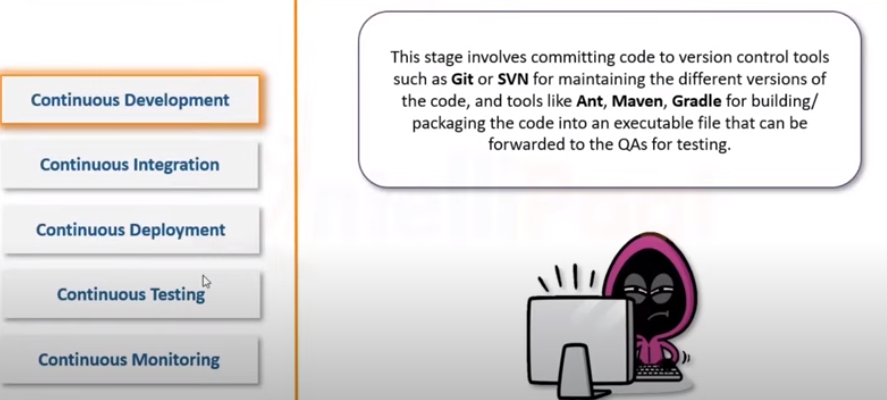
Description automatically generated

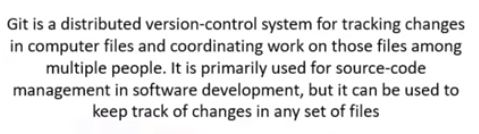
A diagram of a software development process

Description automatically generated

When theres a change by a developer in code and it is pushed using the version control the CI identifies the changes and builds it and testing on the testing server, if tests succeeds to CI transfers to production server ,if there;s error the feedback is given using the monitoring tools to the developers.

On the productions server (the main server) it gets continuously monitored and logs are generated and stored.





A screenshot of a computer

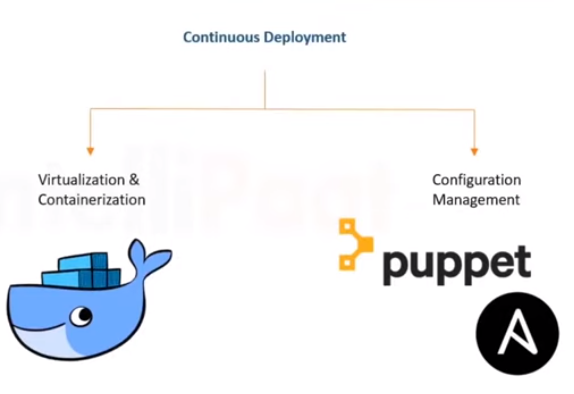
Description automatically generated

A white background with black text

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A white sign with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

A white rectangular sign with black text

Description automatically generated

**DOCKER**

Docker is like a lightweight, portable box that contains everything an application needs to run, such as code, libraries, and dependencies. It allows you to package your application and its environment into a standardized unit called a "container," which can run consistently across different environments, like your laptop, a server, or the cloud.

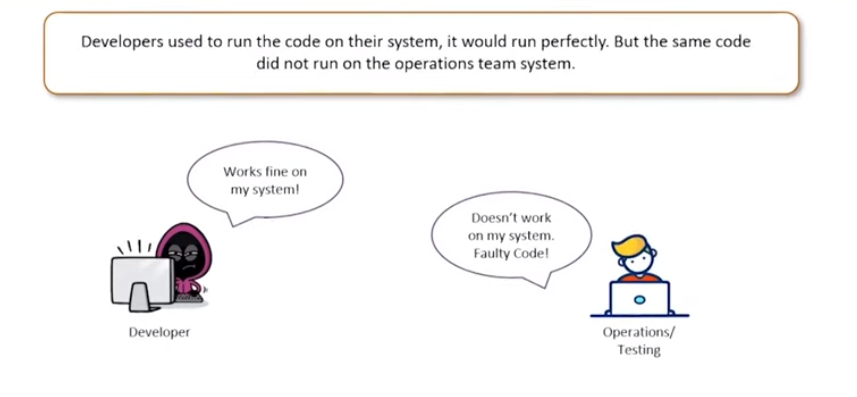
Docker provides OS level virtualization also known as containerization.

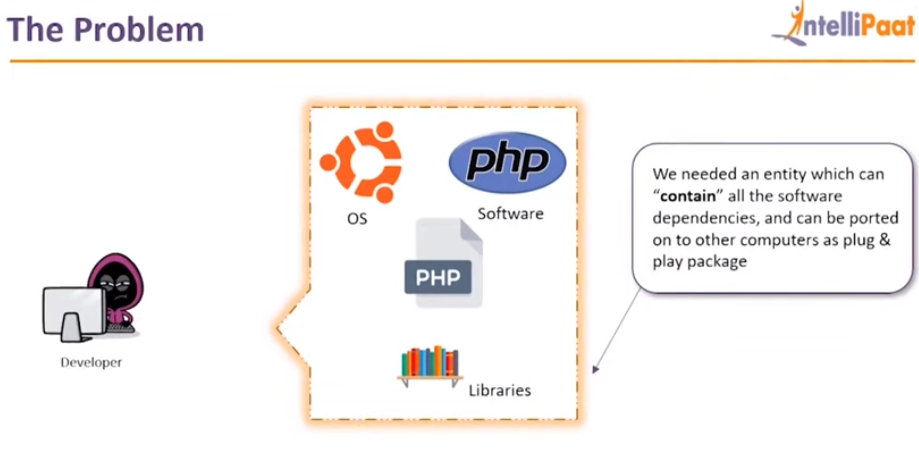
Lets say developer creates a php website on his system with all config. And then the php file is sent for testing to a tester then the tester would have to configure(OS, dependencies, libraries) his system with all the same requirement to run the php website.

Even if the tester does the same then there might be differences in versions of the softwares used.

A computer screen with a person working on a computer

Description automatically generated

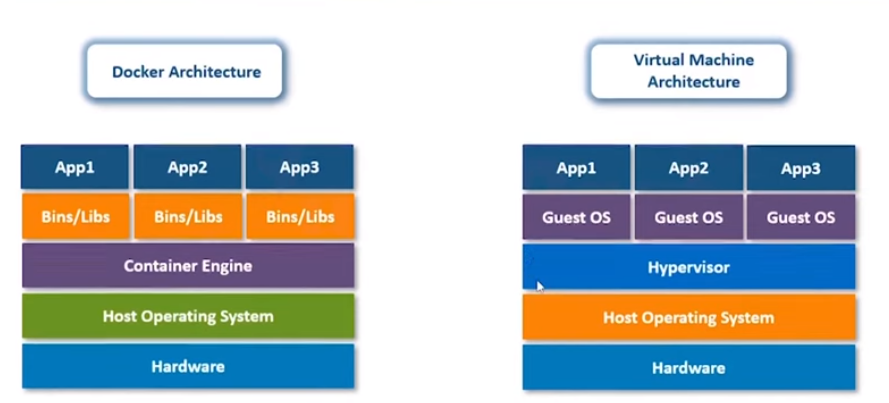




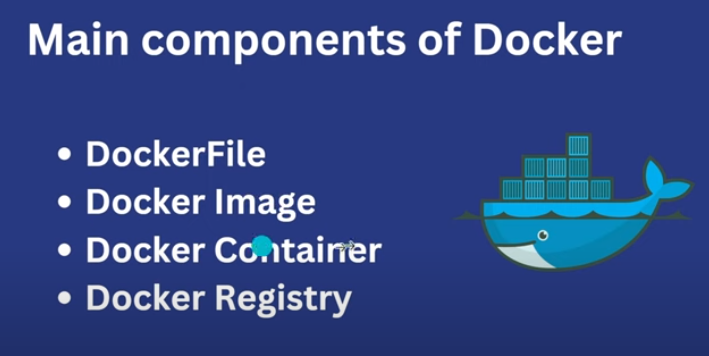
**Containers**

Containers are a lightweight, portable, and isolated way to package and run applications, along with their dependencies, in a consistent environment across different computing environments. They offer a way to package software in a format that can be easily moved from one environment to another without worrying about dependencies or compatibility issues.

Containers share the OS i.e. they run on top of the host OS sharing the resources like CPU ram storage etc, (that’s why their size is small) while VM run on the Guest OS having virtualized resources (separate ram, storage) as a result they VM are slow and heavyweight as compared to containers.



**Components of Docker**



a Dockerfile is a blueprint for building a Docker image, while a Docker image is a packaged and executable version of the application along with its dependencies. The Dockerfile defines how the Docker image should be constructed, while the Docker image is the result of building the Dockerfile and is used to instantiate Docker containers.

* A Dockerfile is a text file that contains instructions for building a Docker image.
* It defines the configuration of the image, including the base image, dependencies,
* A Docker image is a read-only template used to create Docker containers.
* It contains everything needed to run an application, including the code, runtime, libraries, and dependencies.
* A Docker container is a runnable instance of a Docker image.
* It encapsulates an application and its dependencies, running in an isolated environment on a host operating system.
* Docker registry is a repository for Docker images, where images are stored, managed, and distributed. It serves as a centralized location for storing and sharing container images,

#taking node js as the base , if latest version you leave it as it else you have to sepify the version

FROM node

#assume container as empty space, where you'll run the applc. so make a folder in the container

# and copy all the coding files in this working directory

WORKDIR /myapp

COPY . .

RUN npm install

# we are using the below command as we wont run the appl. ,first we need to install the dependencies

CMD [ "npm","start" ]

---------------------------------------------------------------

# run command "docker build ."to create the image

# image will contain everything in the application i.e. dependencies

# check if image is created or not "docker image ls"

# after creating the image run  it : run image imageID

# it wont run this time since it is running in the container and unable to connect with the outside enviroment i.e browser upon which the address run

# first stop the container: "docker stop container\_name" get name using docker ps

# run command : docker run -p 3000:3000 imageID ---- : -p is port binding

# as you can see after running the container it runs in foreground the stucks the terminal , so to make it run in the background use : docker -d -p 3000:3000 imageID using -d

# to run multiple containers: "docker run -d -p 3002:3000 ImageId", 3000 is same since they are different containers and they are independent fo each other: specifying the port

# to remove the container from the list automatically after stopping it use : "docker run -d --rm -p 3002:3000 ImageId" (add --rm)

# to add custom name to the container : "docker run -p  --name "name" --rm -d  imageId"

# to give name to the image instead of using the imageID: "docker build -t name:version ."

# removing the container : docker rm imageName/ID

# removing the image : docker rmi imageID/name

# if theres an update in code then build the image again and change the

version number : "docker build -t name:version2 ."

# you can use predefined images like python,nginx: "docker pull python/nginx"

# for interactive prgorams eg. taking input from user use interactive mode : "docker run -it imageID"

# pushing image to the dockerHub / repo : uploading it to be used to other members of the team:

1.create the repo in dokcerHub : the name of the web app should be same as name of image:

2.docker push philipaul/webapp-demo:01 here philipaul/webapp-demo is the name of the image as well as the repo created in dockerHub , upon using this command the image is pushed to the repo.

# renaming the image use tag : "docker tag mywebapp:02 tomato:05"

# docker VOlume: it is used in case of programs where in theres a need to store the info . eg user inputs a name and it is saved in a file

# command :"docker run -it --rm -v myvolume:/myapp/ imageID " here it is to be considered that the loc of volume should be the same as that of Working directory as specified in Dokerfile

# mount binding : take eg. wherein a file named server.txt contains the names of servers and is changed by the user and this file is used as source for a python prg which displays the name

# so if we just display the names and servers runs but if the file is changed it wont show the changes but would display the names befoer changing so the absolute path is used : docker run -v /pathOFServer.txt file:/myapp/ --rm imageID

# .dockerignore is used to ignore the files which are not required o be used in the image created from the dockerFile

------------------------------

# When API is used in application : edit the dokcerfile add : pip install requests after COPY command

# database is used(dataset): edit the dokcerfile add : pip install pymysql after COPY command and edit the code of the actual file

If the dataase is in the local machine change host in appl to host.docker.interal

If datanase is in another container change host to the IP address of that container (container with db) and add as well

-------------------------------

If env. Is required in some cases the you need to specify the name and pass of thet variable eg.

“Docker run -d –rm –env MYAPIpass=”root” --env APIurl=:”url” -- name myImage imageID”

Docker volume, docker multiplw containers, docker compose