DevOps


प्रगत संगणन विकास केंद्र
सी डैक
CDAC
॥ ज्ञानादेव तु कैवल्यम् ॥
CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING

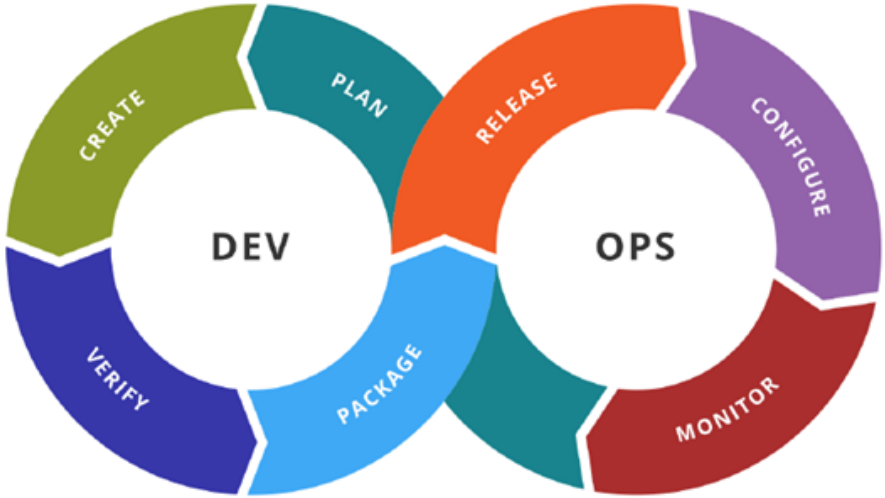
DevOps

DevOps is short for **Dev**elopment and **Op**eration**s**. It concentrates on collaboration between developers and other parties involved in building, deploying, operating, and maintaining software systems.

DevOps - History

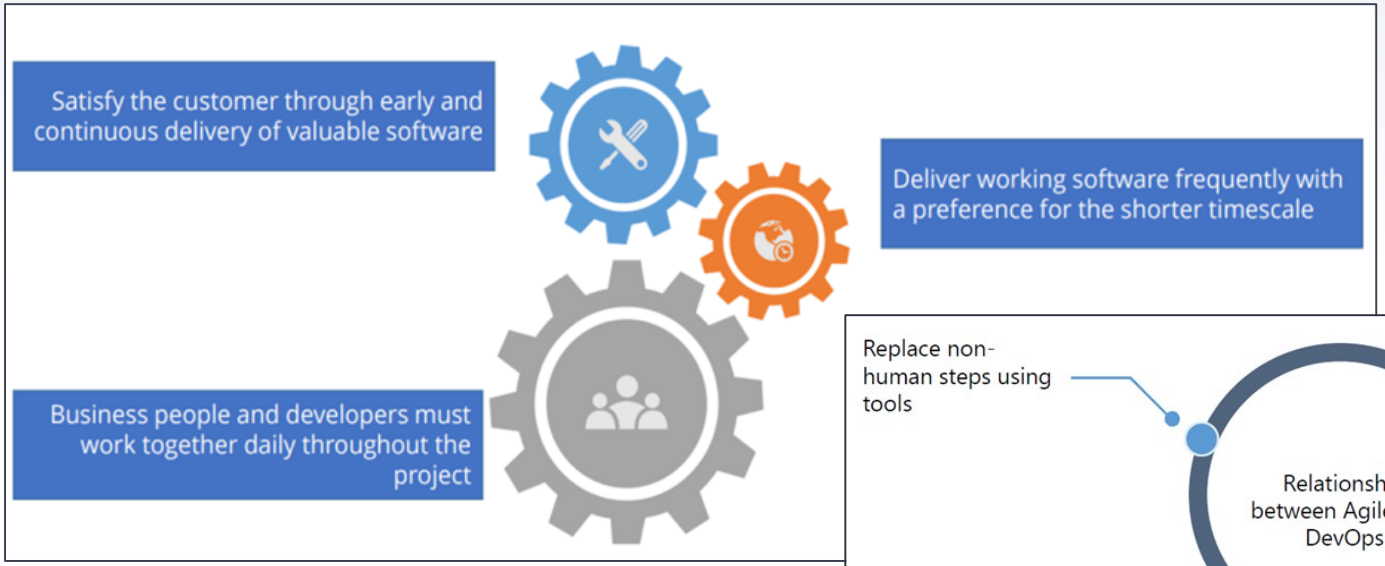
- Patrick Debois, a Belgian consultant, project manager, and agile practitioner is one among the initiators of DevOps.
- A presentation on "10+ Deploys per Day: Dev and Ops Cooperation at Flickr" helped in bring out the ideas for DevOps and resolve the conflict of " It's not my code, it's your machines! "
- DevOps blends lean thinking with agile philosophy.

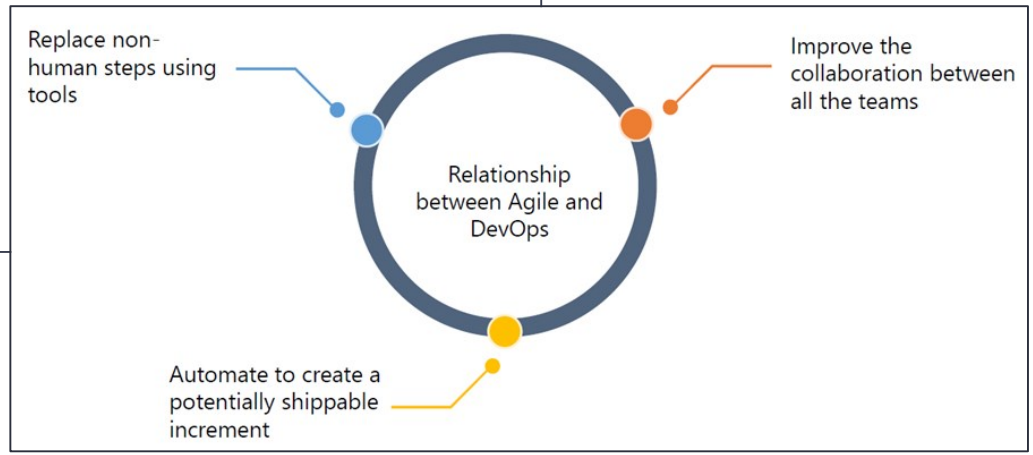
DevOps - Overview



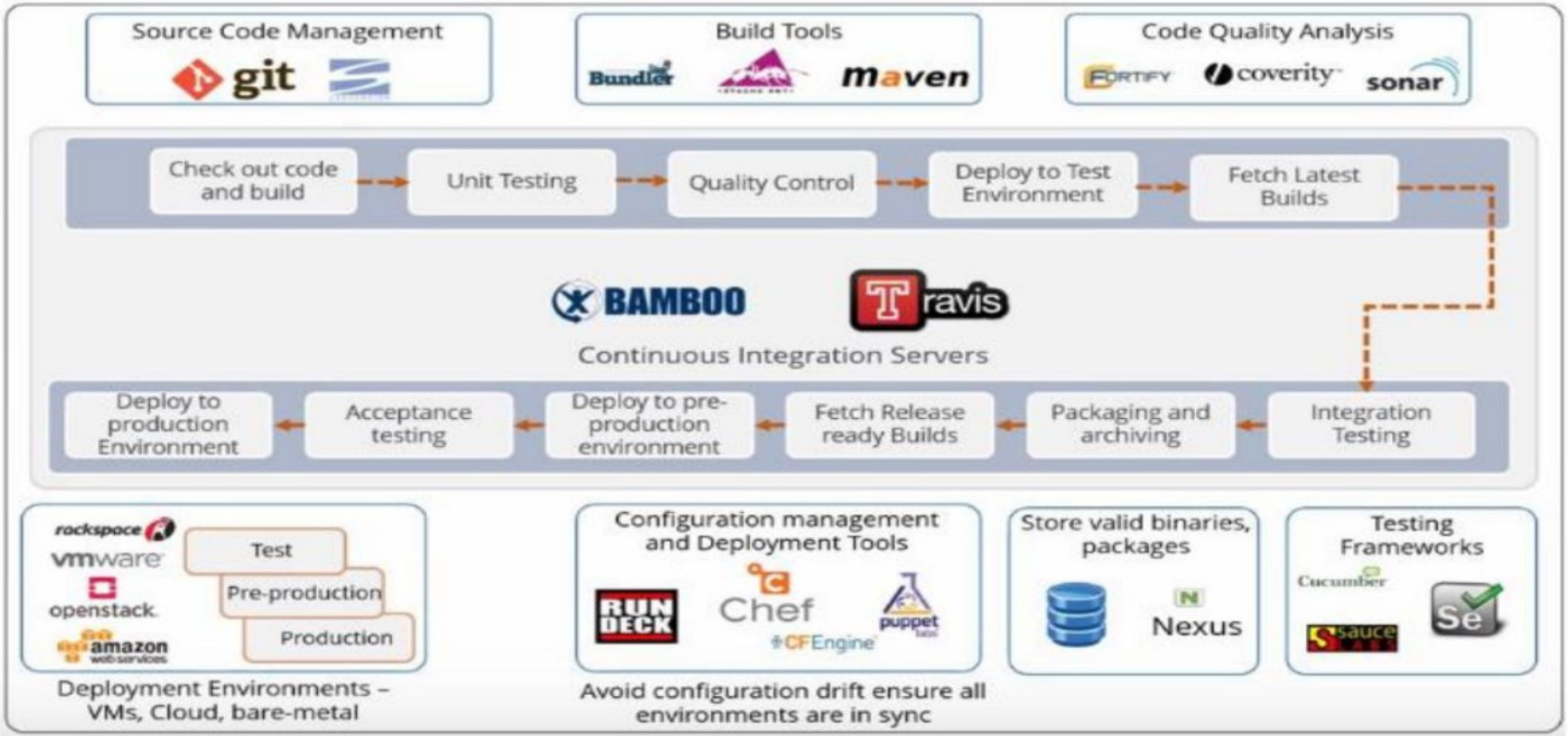
- DevOps is an agile relationship between development and IT operations
- DevOps is the abbreviation for **Dev**elopment and **Op**erations
- The Development includes Plan, Create, Verify, and Package
- The Operations include Release, Configure, and Monitor

Relationship between Agile and DevOps





Agile and DevOps Example



DevOps Toolchains

Monitoring Performance

Releasing into Production

Building Applications

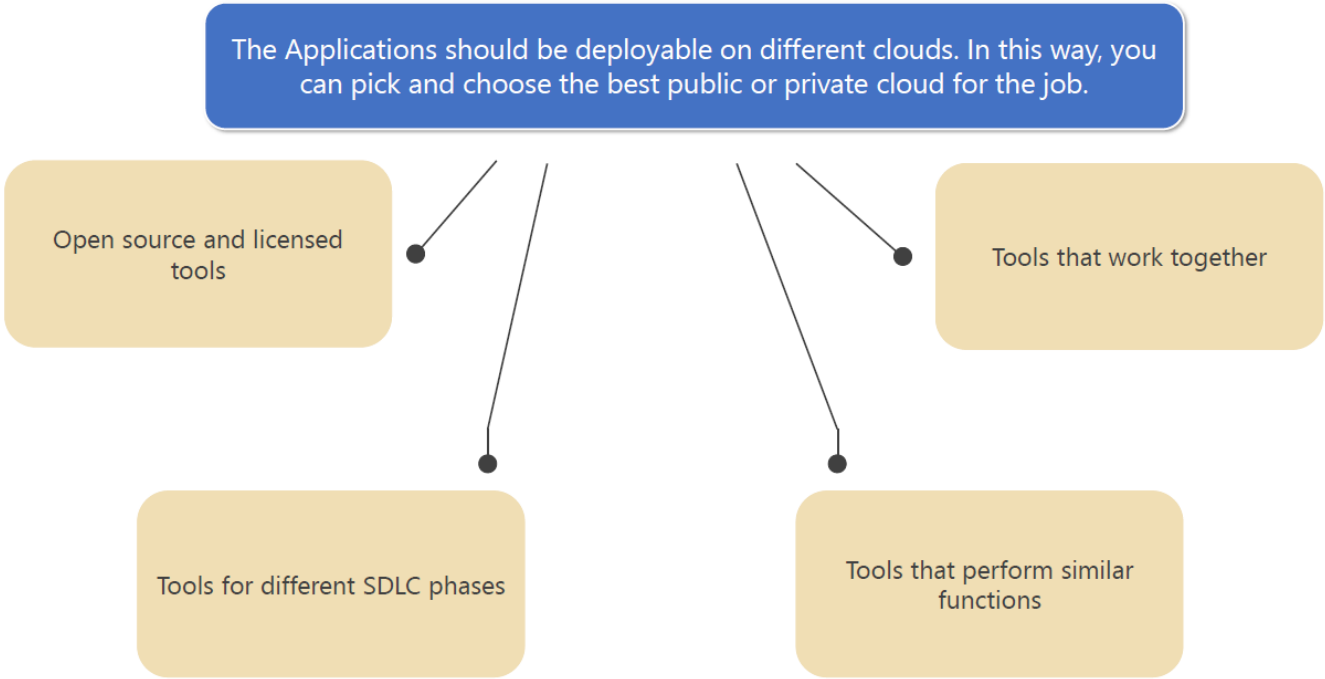
Code Development and Unit Testing

Configuration Management

Integration and Performance Testing

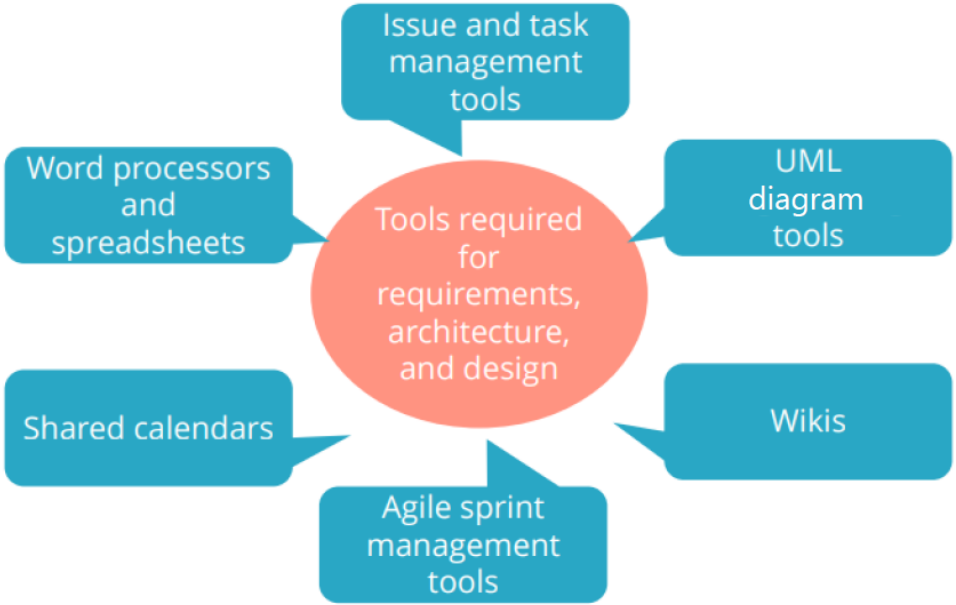
Packing the Application

Proper DevOps Tools Selection

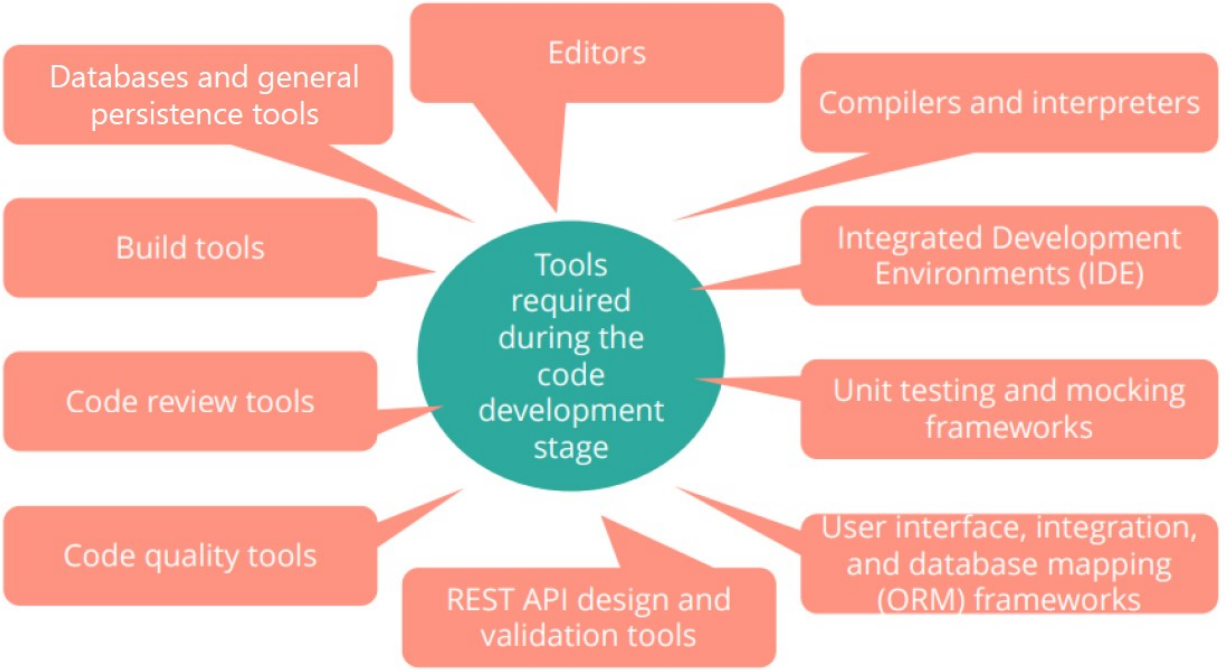


Requirements Tools

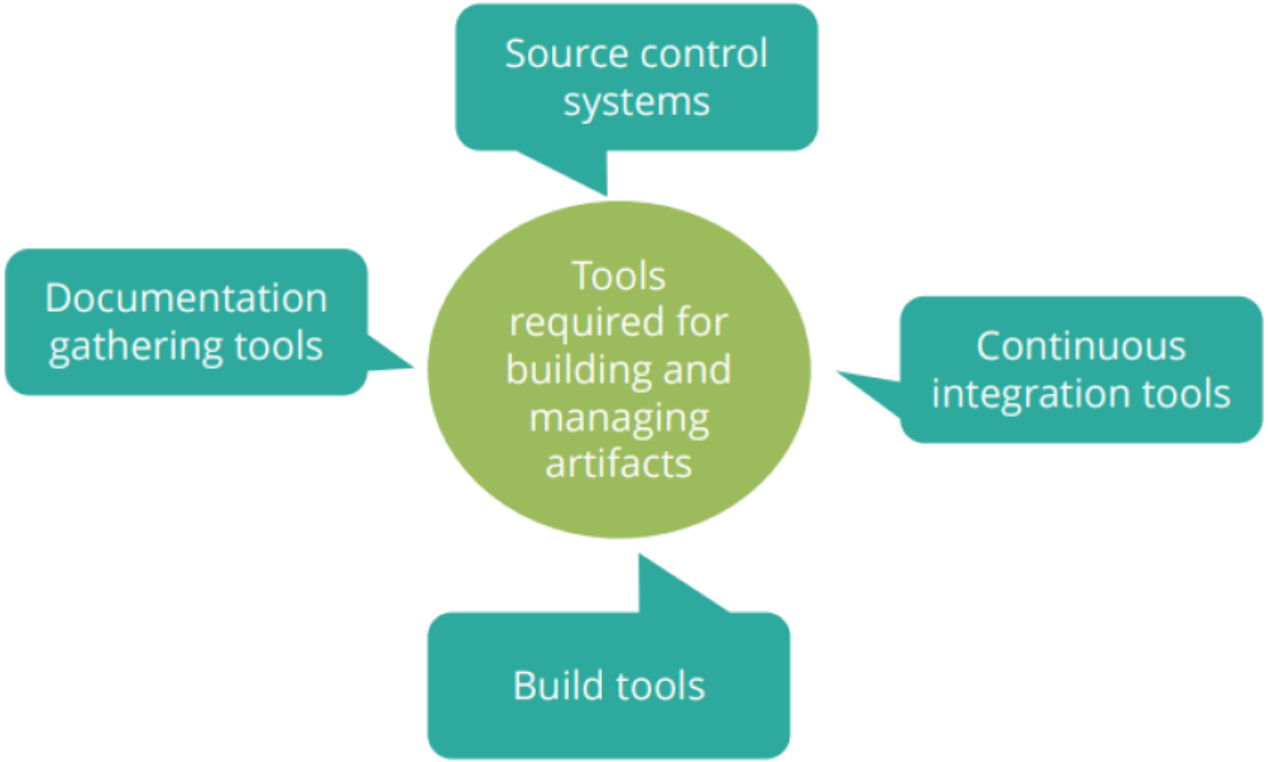
Tools are used to share the files and communicate within the team and other teams.



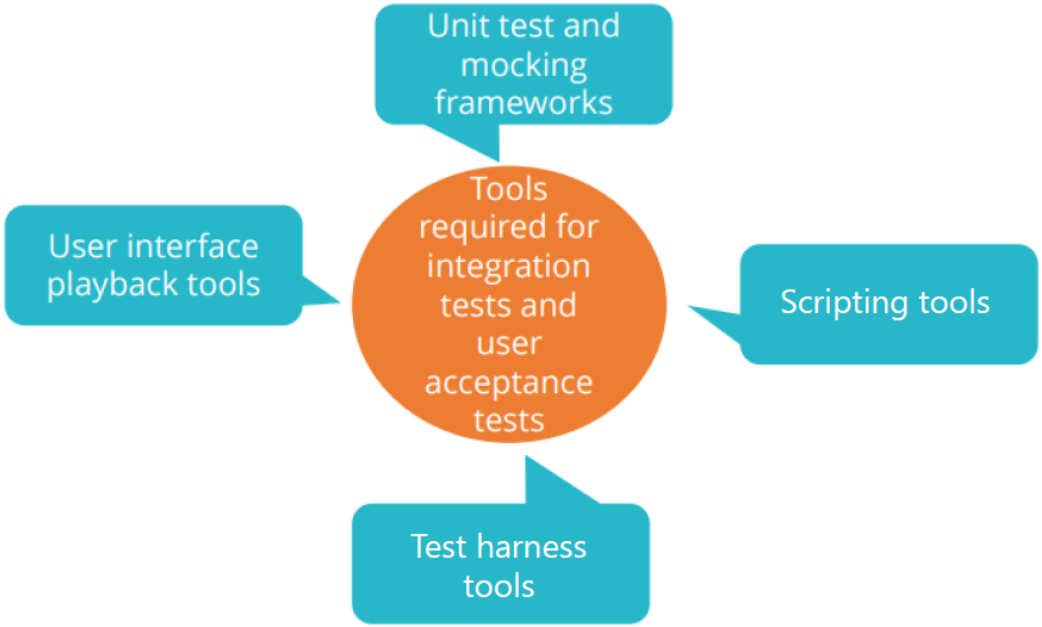
Code Development Tools



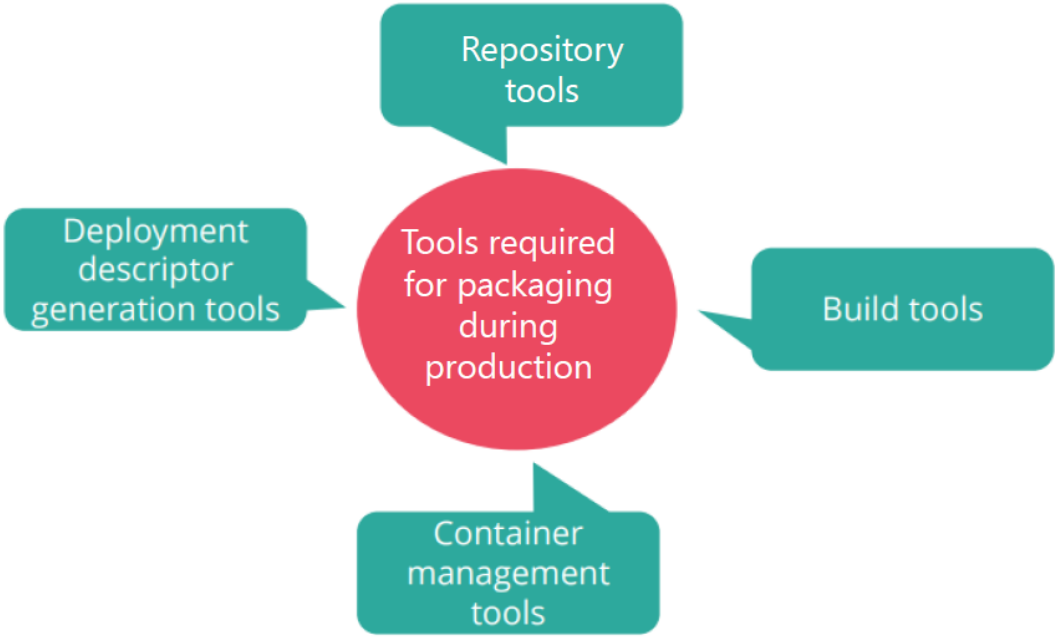
Artifact Creation Tools



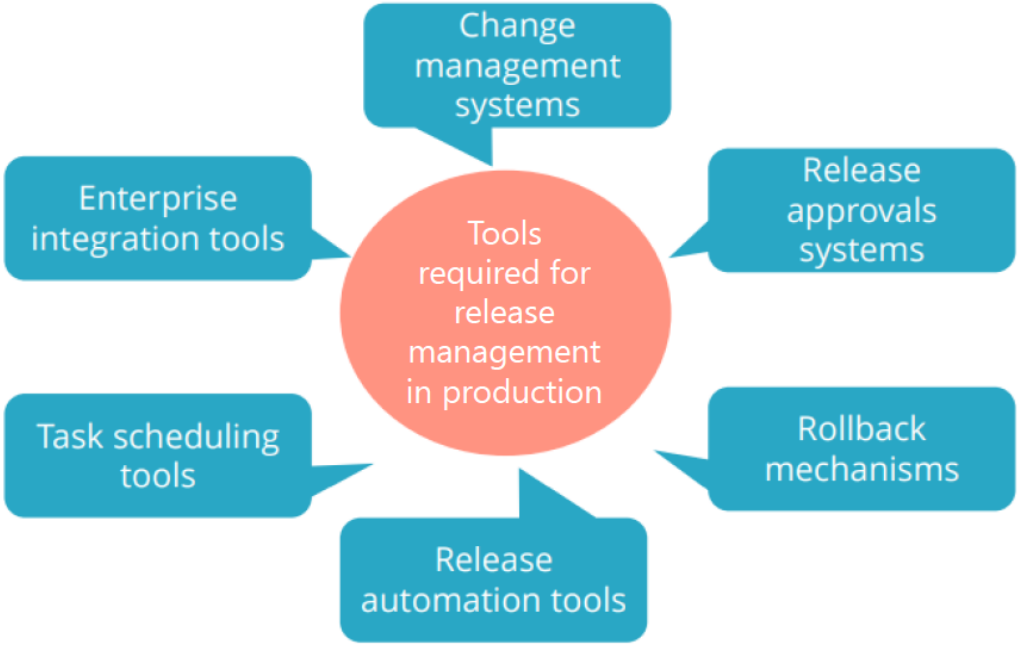
Testing Tools



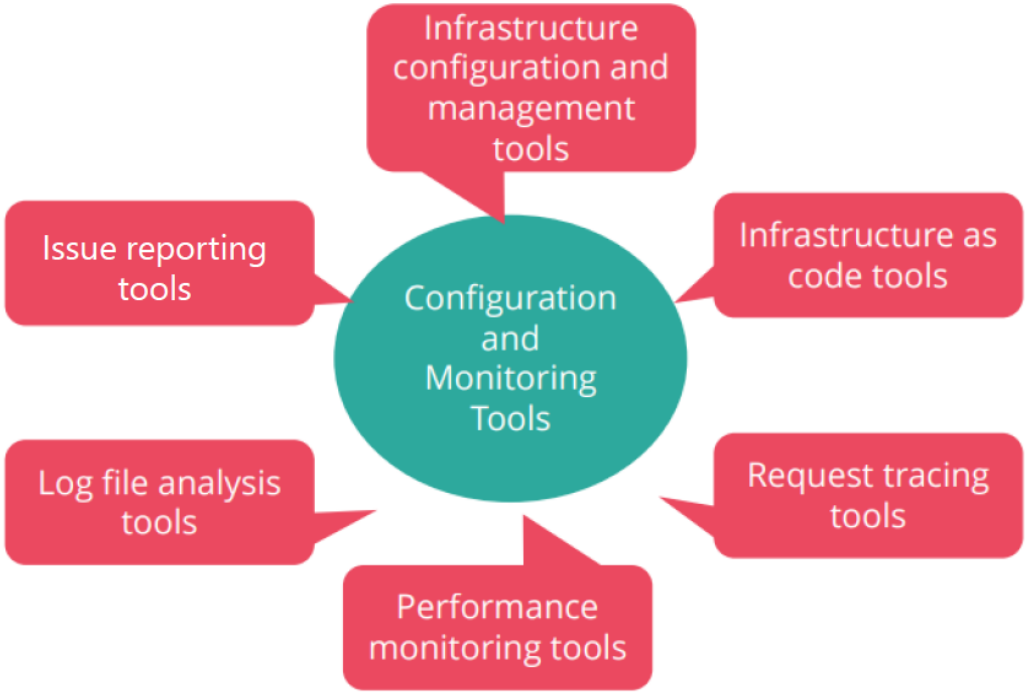
Packaging Tools



Release Management Tools



Configuration and Monitoring Tools



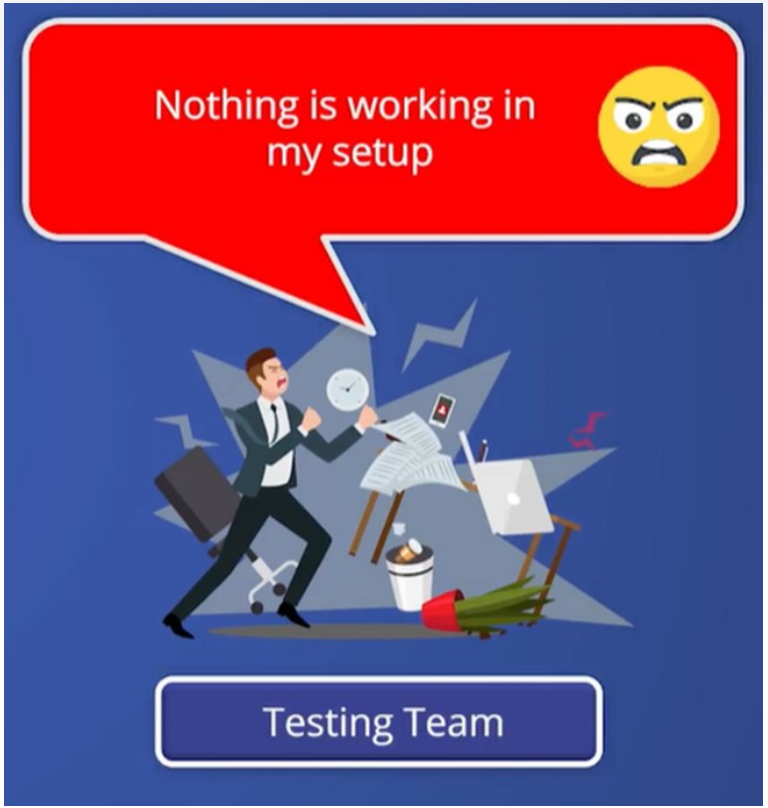
Introduction To Containerisation

docker

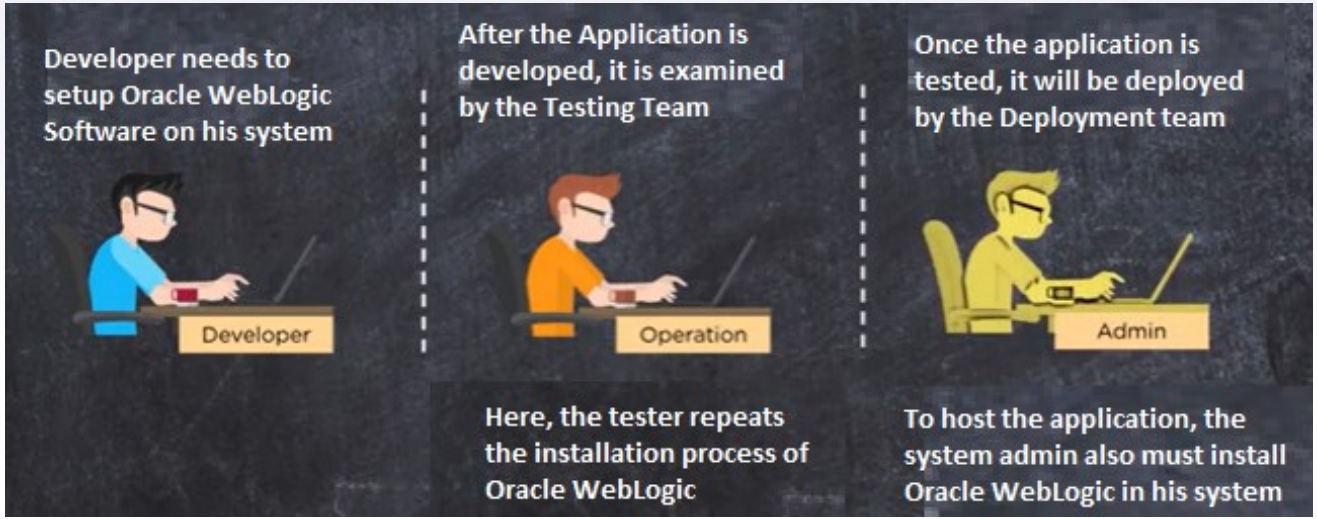
Before Docker

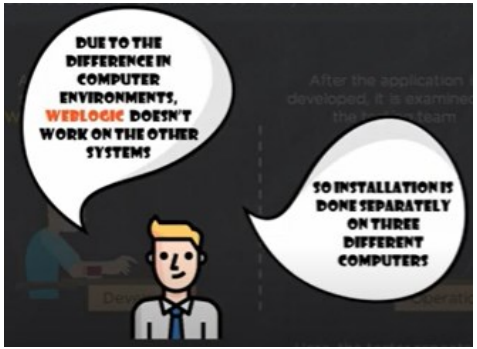
Result


Code is working wonderful here!!
Development Team


Nothing is working in my setup
Testing Team

Eg: Oracle WebLogic Software





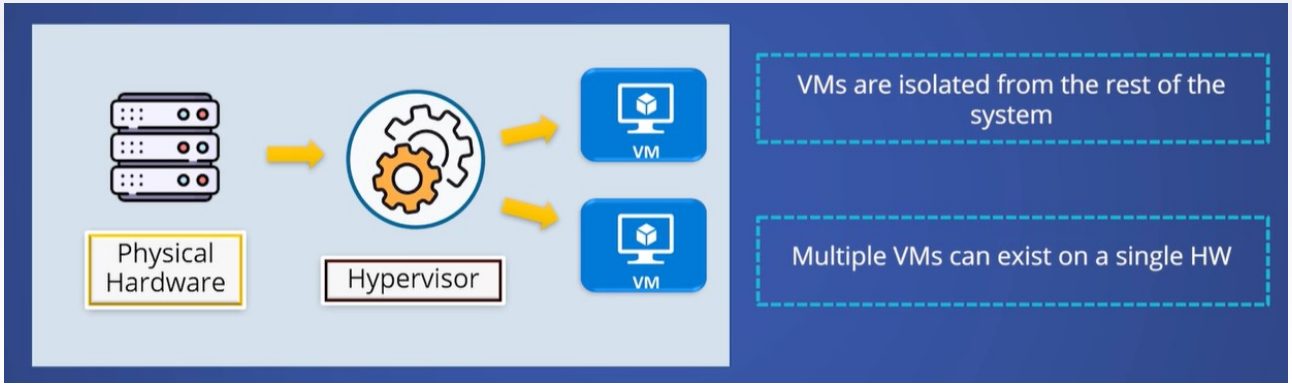
Solution 1: Virtual Machine

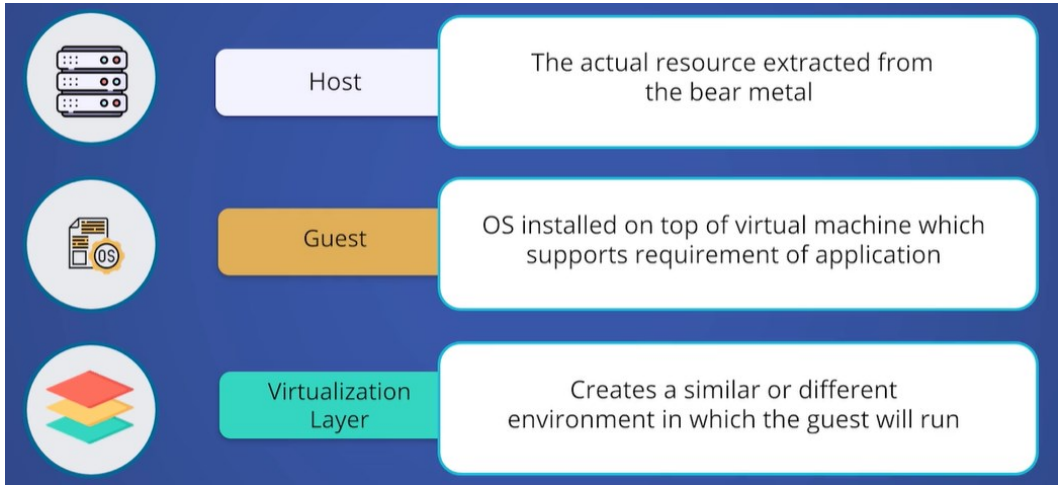
A virtual machine (VM) works as a virtual computer system

Hypervisor
Software/application that separates the machine's resources from the hardware and provisions them appropriately so they can be used by the VM

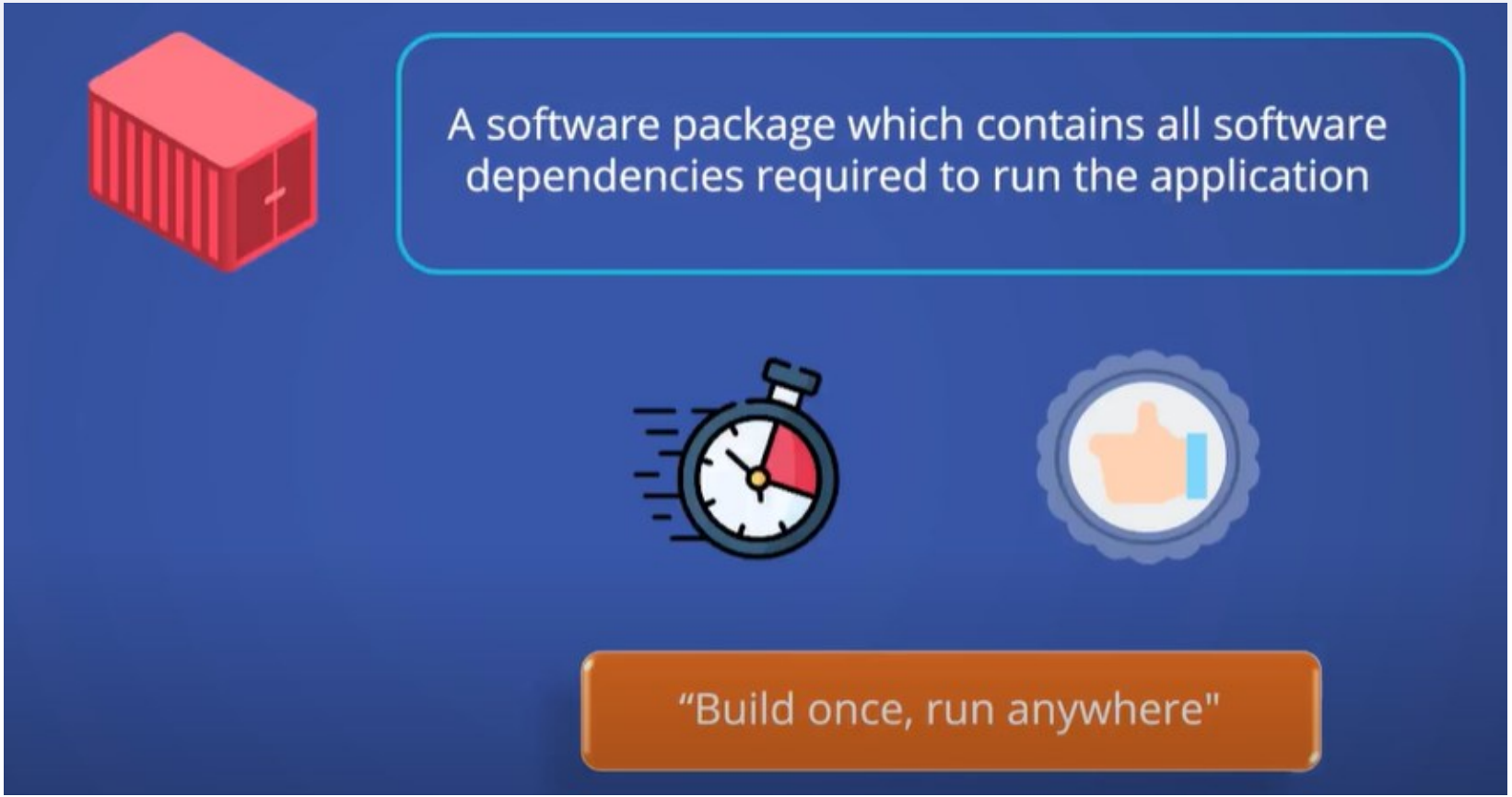
VM has its own CPU, memory, network interface, and storage, created on a physical hardware system

What is VM?

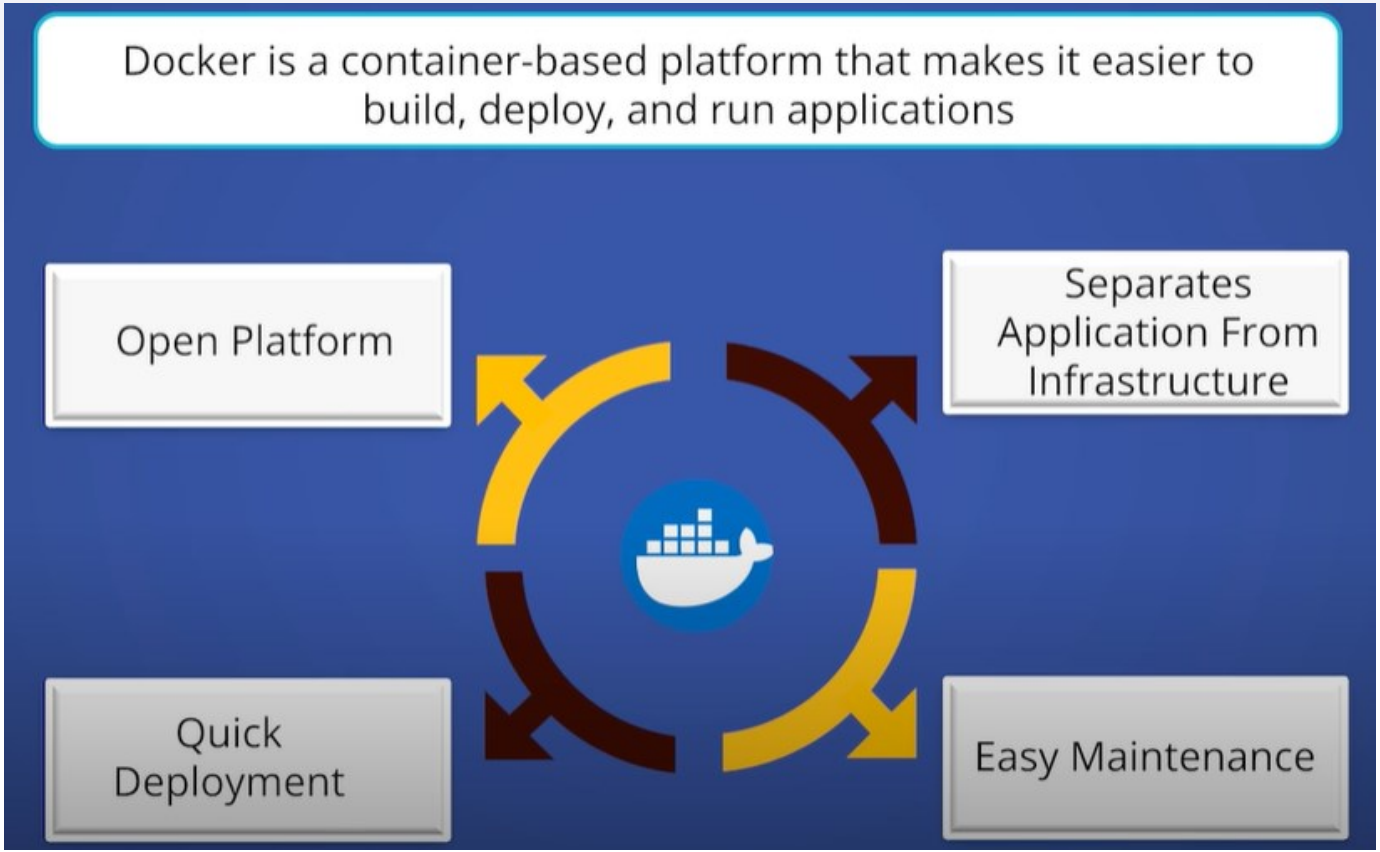

Physical Hardware
Hypervisor
VM
VM
VMs are isolated from the rest of the system
Multiple VMs can exist on a single HW

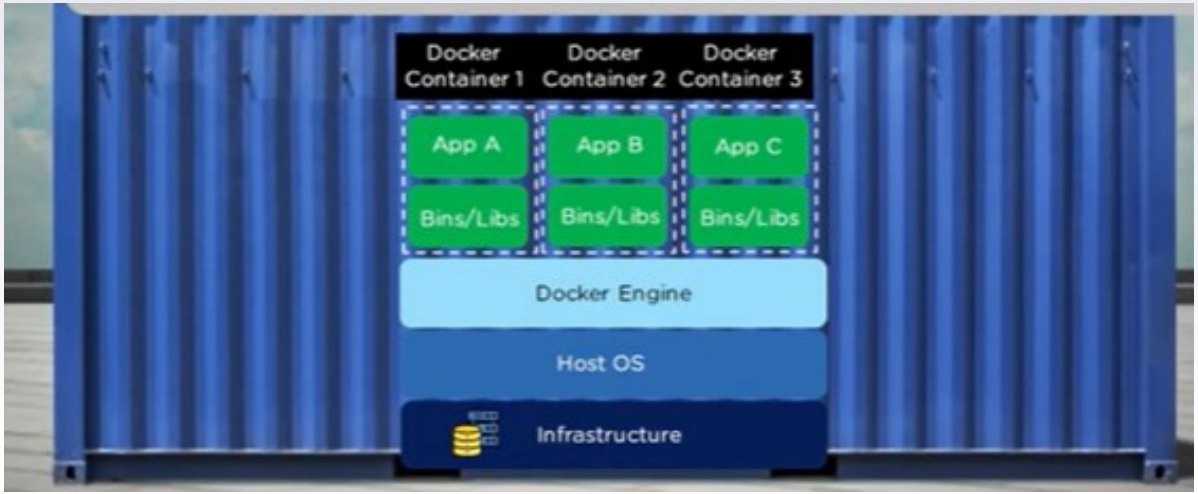

Host
The actual resource extracted from the bear metal
OS
Guest
OS installed on top of virtual machine which supports requirement of application
Virtualization Layer
Creates a similar or different environment in which the guest will run

Solution 2: Container



Docker By Mirantis

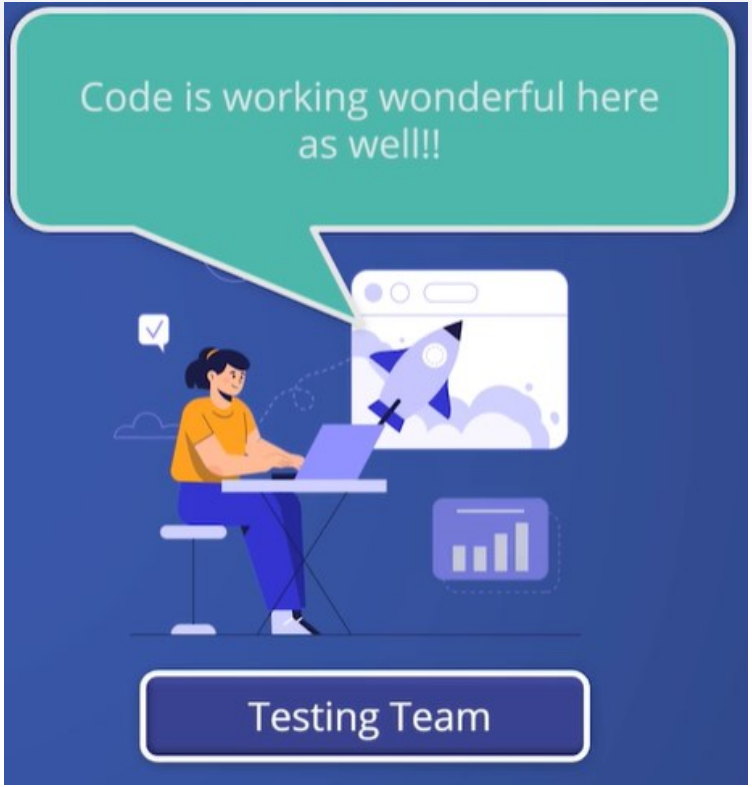




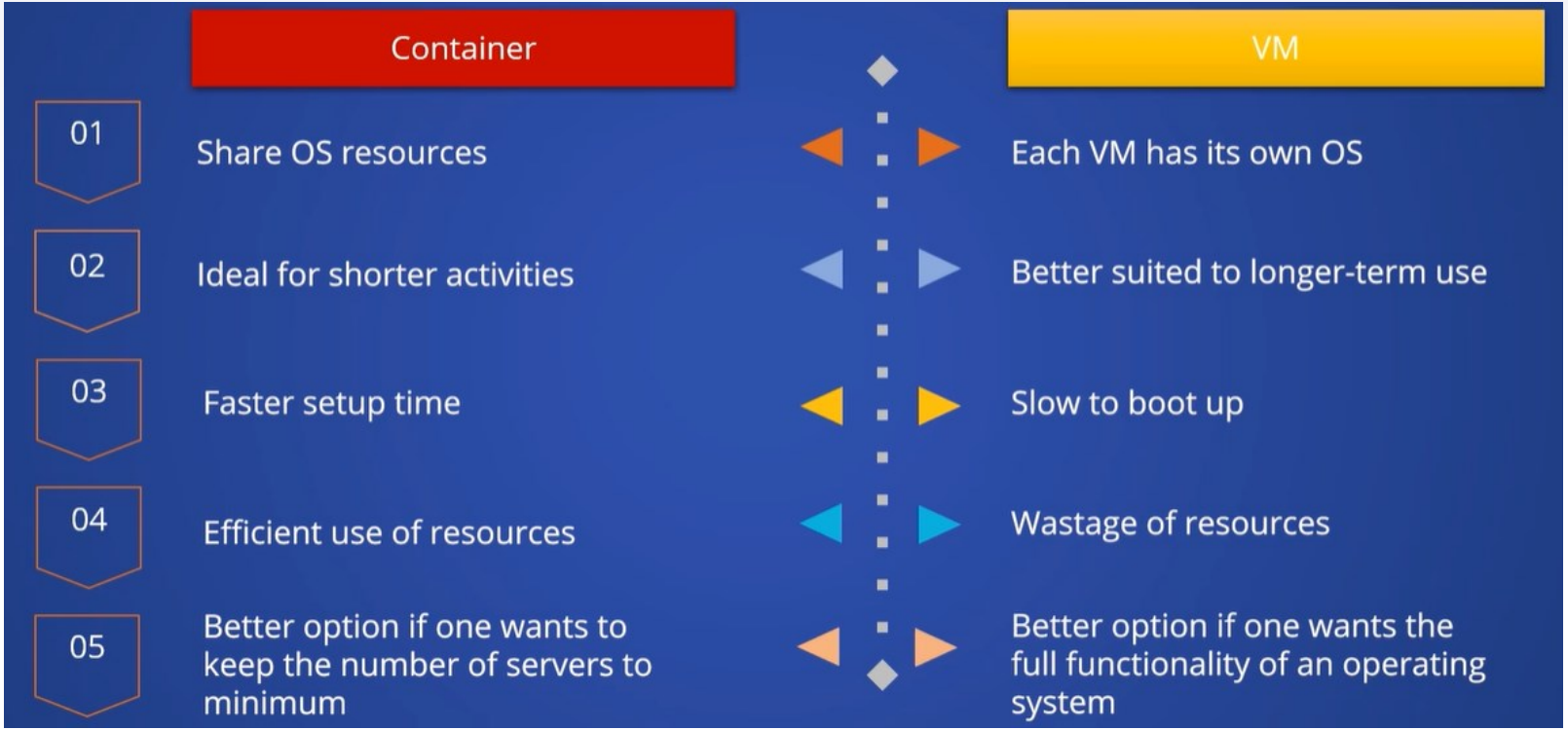
- **Docker** is an open-source Platform that helps the user to package an application and its dependencies into a Docker Container for the Development and Deployment of Software.
- Containerisation includes all dependencies (libraries, frameworks etc..) required to run an application in an efficient and bug-free manner.
- With Docker containers, applications can work efficiently in different computer environments.

With Docker

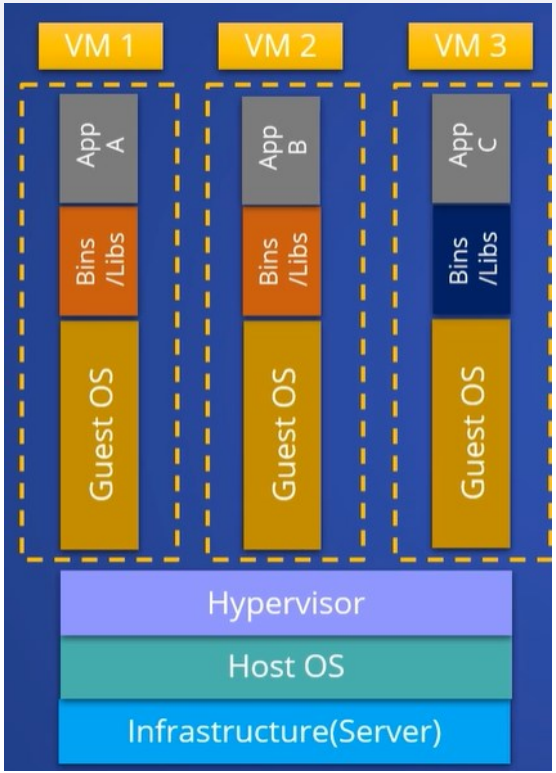

Code is working wonderful here!!
Development Team

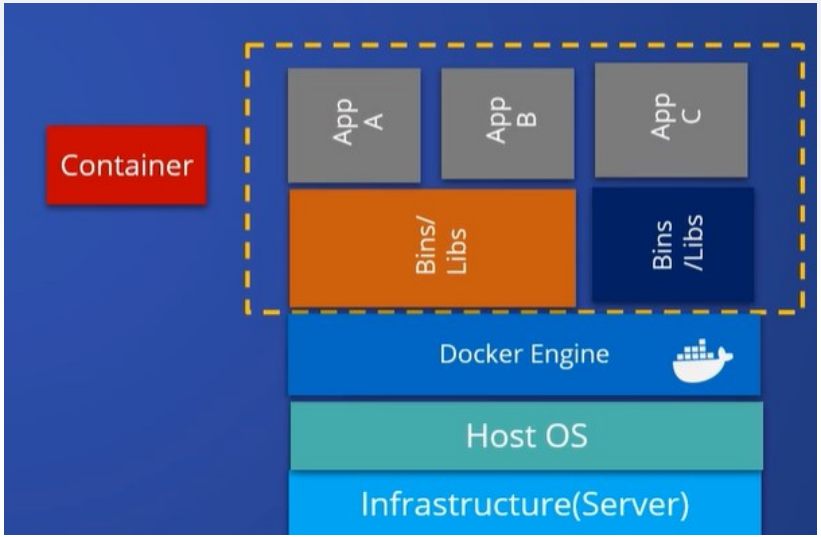

Code is working wonderful here as well!!
Testing Team

VM vs Container



VM vs Container





Each virtual machine has its guest OS which makes it heavy

Container share host operating system

Containerisation

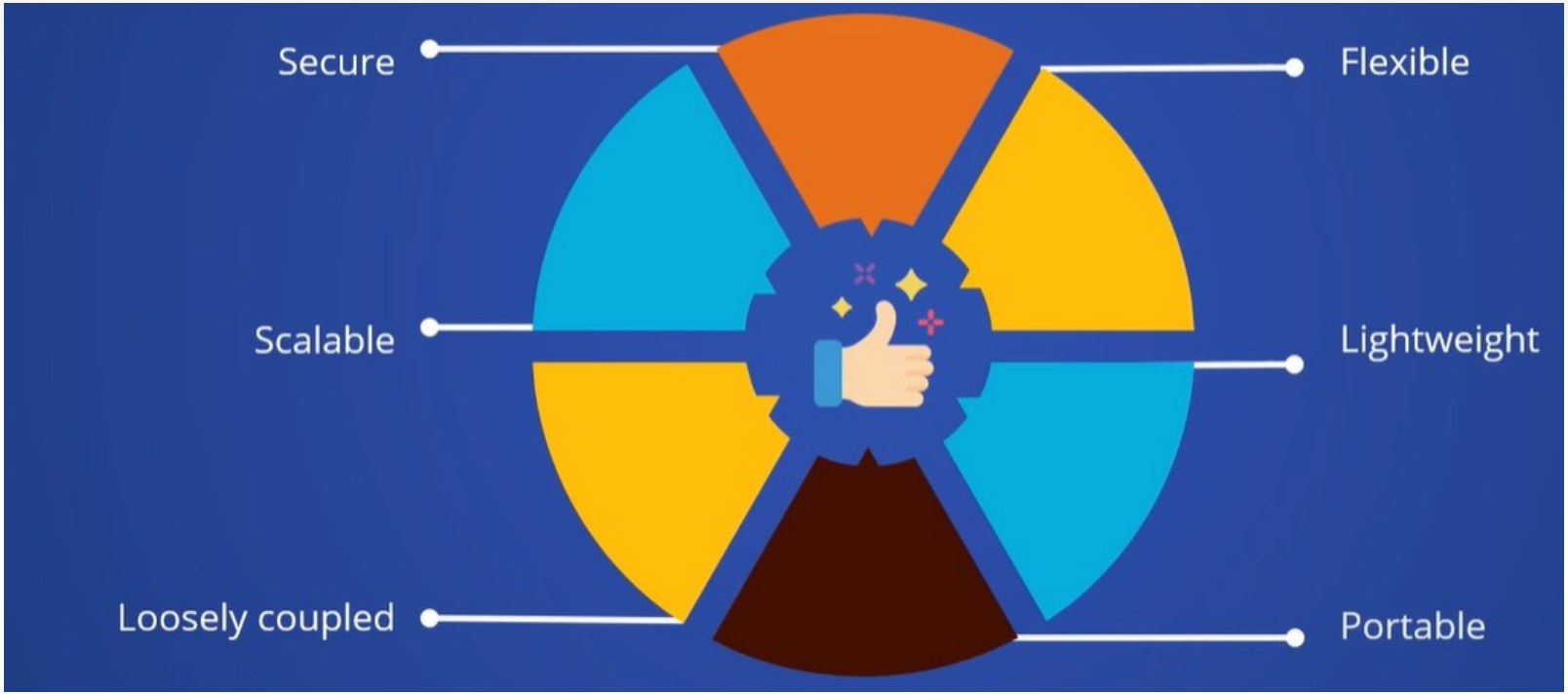
Containers

A container is a standard unit of software that packages - **code** and **all its dependencies**

Containerization is the use of containers to deploy applications

Enables application to run quickly and reliably from one computing environment to another

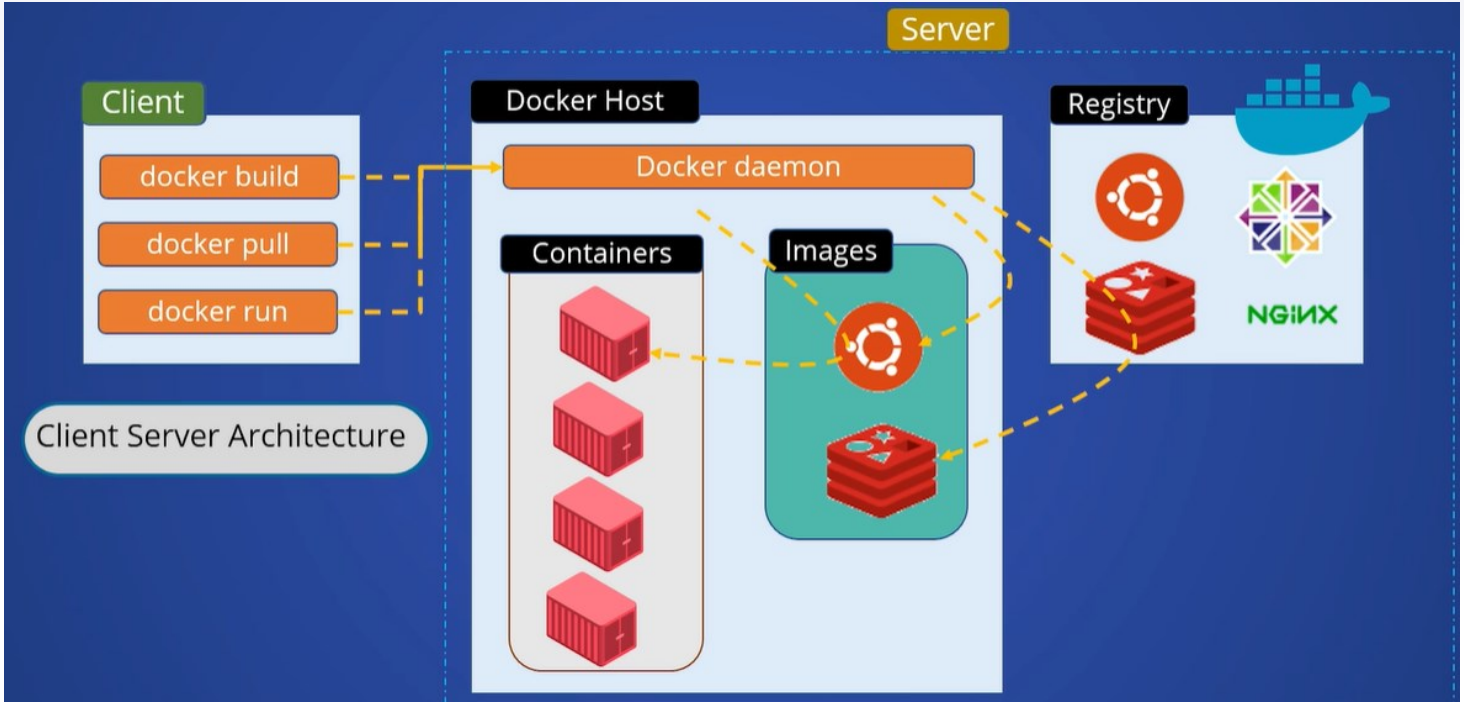
Advantages


Secure
Flexible
Scalable
Lightweight
Loosely coupled
Portable

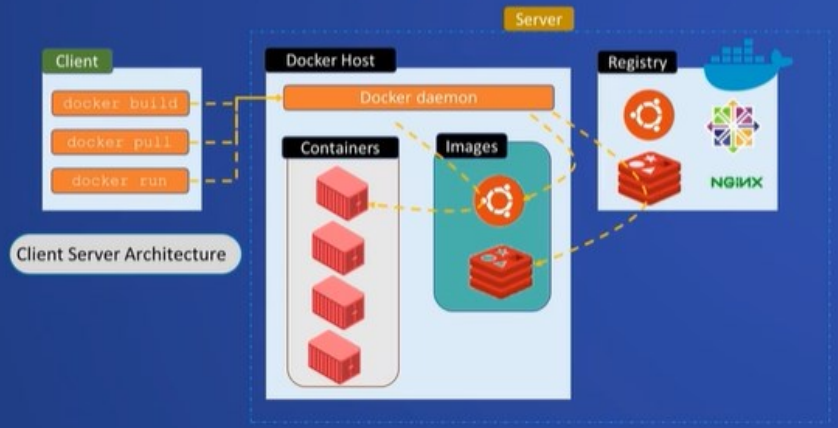
Why Docker?


Brings cloud like capabilities
Containerization bundles an entire runtime environment into a single package
Docker addresses the issue of dependencies by having contained within containers
Each app runs independent of the others

Docker Architecture



Docker Client



Docker users should use a client to communicate with Docker

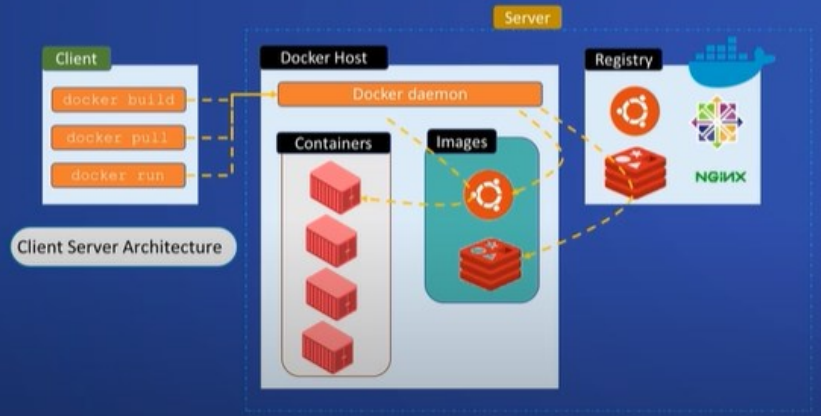
The Docker client has the ability to connect with several daemons

When a client executes a Docker instruction, it sends it to the dockerd daemon, which executes it

Docker commands make use of the Docker API

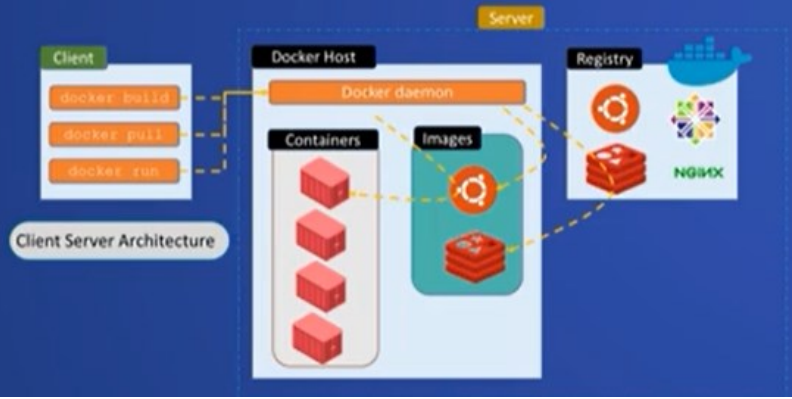
Docker Host

Docker host offers a full environment in which applications can be executed and run



- Docker daemon
- Images
- Containers
- Networks
- Storage

Docker Daemon



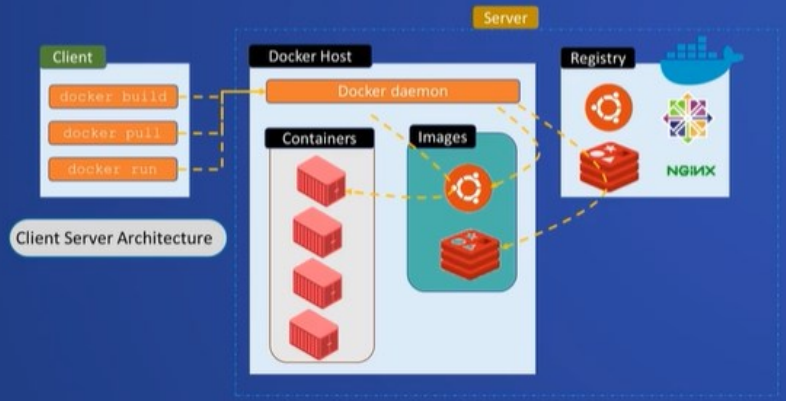
To control its services, it can also interact with other daemons

The daemon accepts commands through the CLI or REST API

The daemon oversees all container-related behavior

Images

A binary template used to create containers

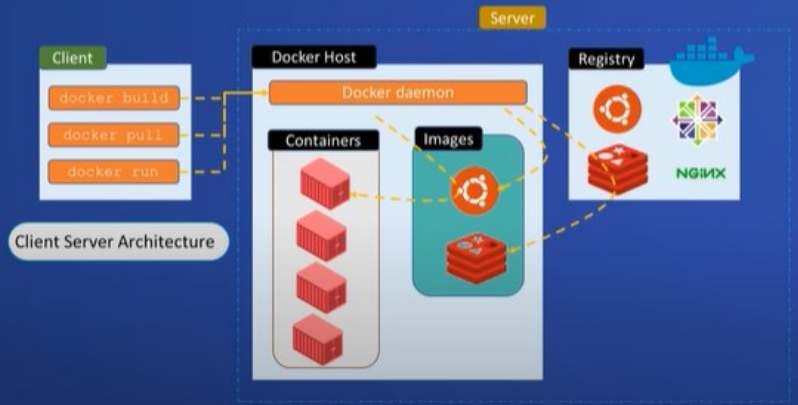


Provide metadata about the container's functionality and requirements

Used to create a container on its own, or it can be customized to add additional elements to the current configuration

Container Registry

Repository to store the Docker images



Private Container Registry

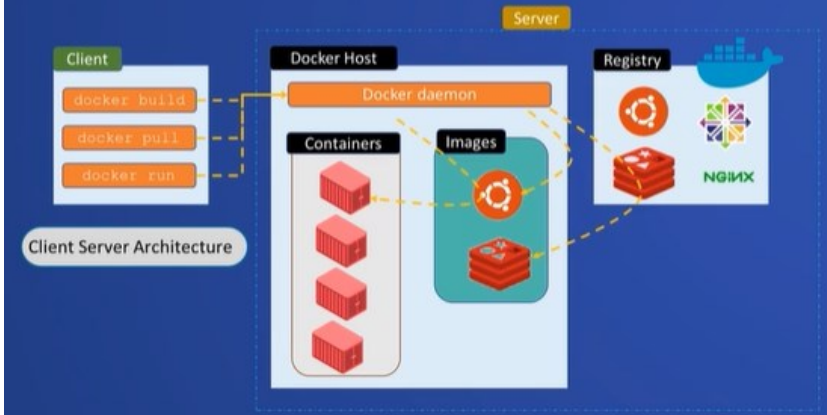
Used to share container images through teams within an organization

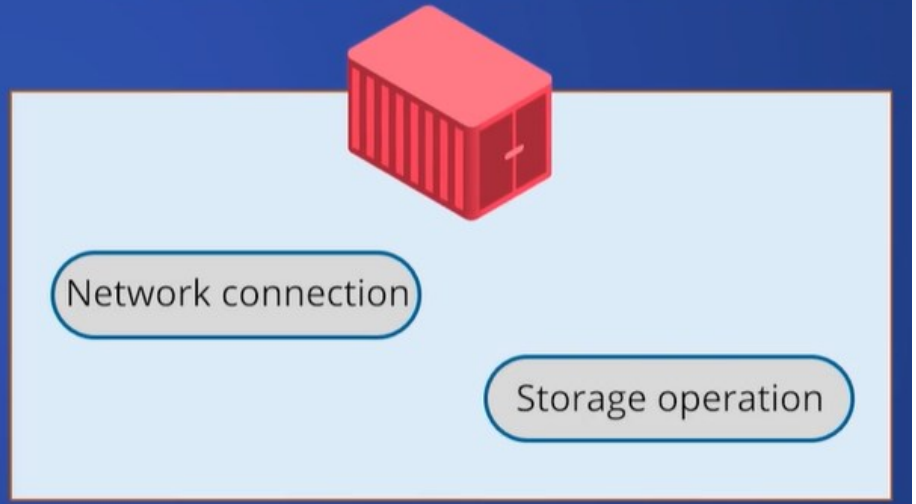
Public Container Registry

Like Docker Hub and can be used to share images with the rest of the world

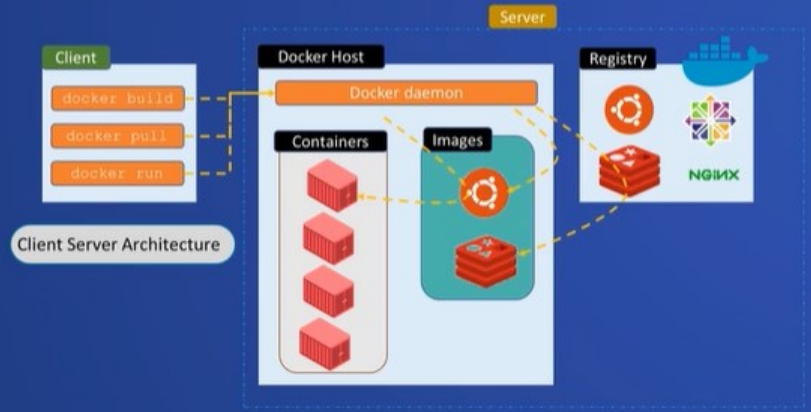
Containers

Encapsulated environments in which applications are run





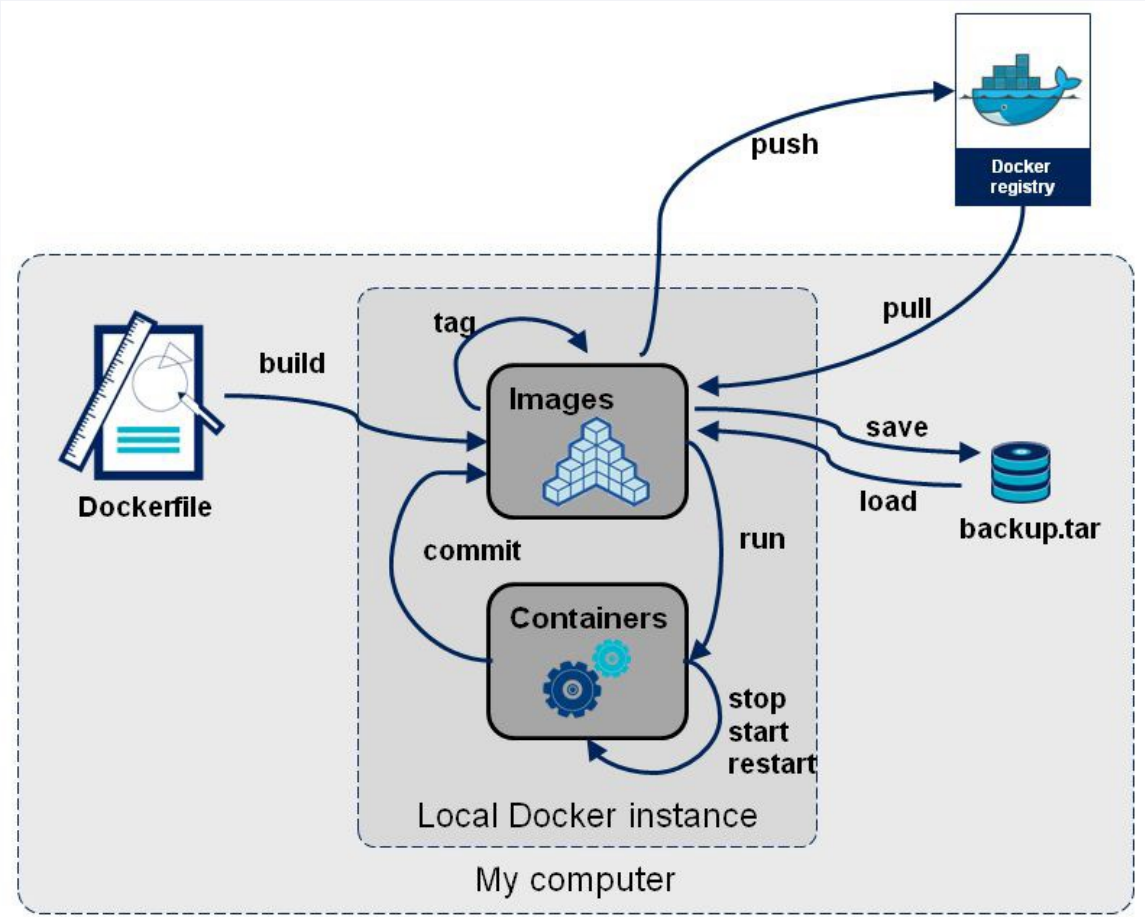
Containers



Containers only have access to the resources defined in the image

A new image can be created based on a container's current state

Docker Container Lifecycle



Dockerize your Application

- A Dockerfile is a fundamental building block used when dockerizing your Java applications, and it is how you can create a Docker image that can be used to create the containers you need for automatic builds.
- **Introduction to Dockerfiles**
- Docker builds images by reading instructions from a Dockerfile. A Dockerfile is a simple text file that contains instructions that can be executed on the command line. Using **docker build**, you can start a build that executes all of the command-line instructions contained in the Dockerfile.

YAML

- YAML Ain't Markup Language (YAML) is a serialization language that has steadily increased in popularity over the last few years. It's often used as a format for configuration files, but its object serialization abilities make it a viable replacement for languages like JSON.
- YAML has broad language support and maps easily into native data structures. It's also easy to for humans to read, which is why it's a good choice for configuration.
- The YAML acronym was shorthand for Yet Another Markup Language. But the maintainers renamed it to YAML Ain't Markup Language to place more emphasis on its data-oriented features.

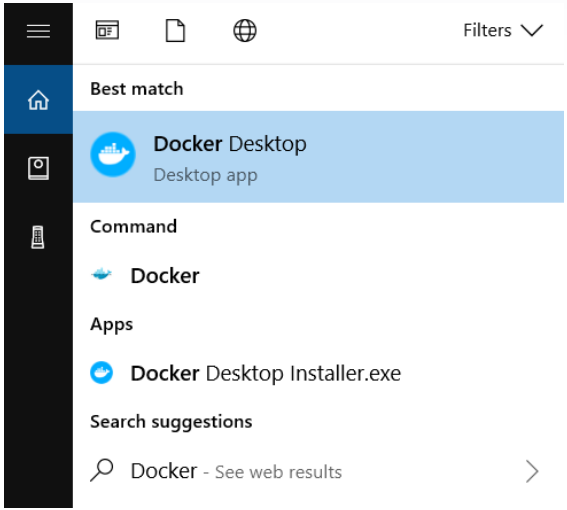
- Common Dockerfile instructions start with RUN, ENV, FROM, MAINTAINER, ADD, and CMD, among others.
- **FROM** - Specifies the base image that the Dockerfile will use to build a new image. For this post, we are using phusion/baseimage as our base image because it is a minimal Ubuntu-based image modified for Docker friendliness.
- **MAINTAINER** - Specifies the Dockerfile Author Name and his/her email.
- **RUN** - Runs any UNIX command to build the image.
- **ENV** - Sets the environment variables. For this post, JAVA_HOME is the variable that is set.
- **CMD** - Provides the facility to run commands at the start of container. This can be overridden upon executing the docker run command.
- **ADD** - This instruction copies the new files, directories into the Docker container file system at specified destination.
- **EXPOSE** - This instruction exposes specified port to the host machine.

Installing Docker Desktop

- https://docs.docker.com/desktop/install/windows-install/
- Double-click Docker Desktop Installer.exe to run the installer.
- When prompted, ensure the **Use WSL 2** instead of Hyper-V option on the Configuration page is selected or not depending on your choice of backend.
- Follow the instructions on the installation wizard to authorize the installer and proceed with the install.
- When the installation is successful, click Close to complete the installation process.
- If your admin account is different to your user account, you must add the user to the docker-users group. Run Computer Management as an administrator and navigate to Local Users and Groups > Groups > docker-users. Right-click to add the user to the group. Log out and log back in for the changes to take effect.

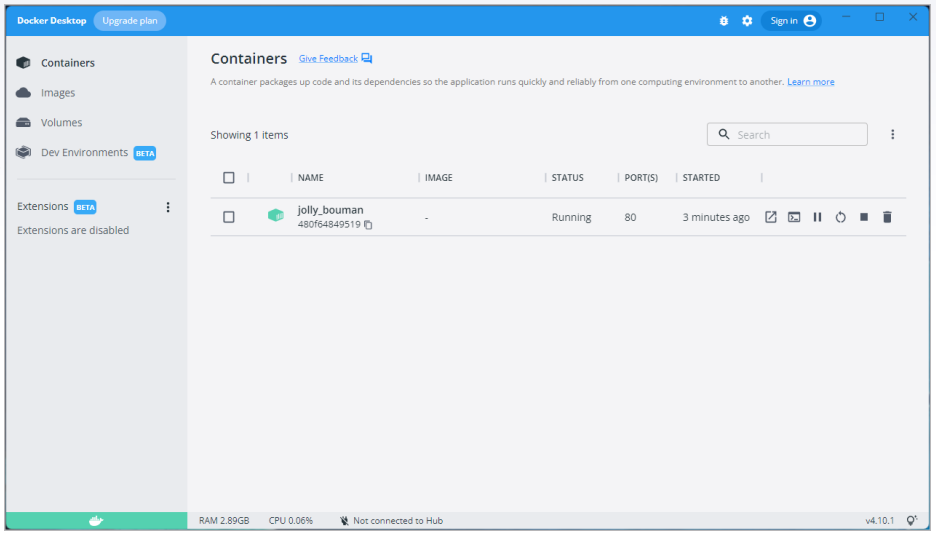
Start Docker Desktop

- Docker Desktop does not start automatically after installation. To start Docker Desktop:
- Search for Docker, and select **Docker Desktop** in the search results.



The Docker Dashboard

- A **container** is a sandboxed process on your machine that is isolated from all other processes on the host machine.
  - is a runnable instance of an image. You can create, start, stop, move, or delete a container using the DockerAPI or CLI.
  - can be run on local machines, virtual machines or deployed to the cloud.
  - is portable (can be run on any OS).
  - is isolated from other containers and runs its own software, binaries, and configurations.



Running your first container

- At first, we are going to run an Alpine Linux container (a lightweight linux distribution) on your system and get a taste of the docker run command.
- To get started, let's run the following in our terminal:
- **docker pull alpine**
- The pull command fetches the alpine image from the Docker registry and saves it in our system. You can use the **docker images** command to see a list of all images on your system.
- Let's now run a Docker container based on this image. To do that you are going to use the docker run command.
- **docker run alpine ls -l**

```
F:\DockerContainers\doctest1>docker run alpine ls -l
total 56
drwxr-xr-x    2 root     root          4096 Jul 18 15:34 bin
drwxr-xr-x    5 root     root           340 Aug  2 10:47 dev
drwxr-xr-x    1 root     root          4096 Aug  2 10:47 etc
drwxr-xr-x    2 root     root          4096 Jul 18 15:34 home
drwxr-xr-x    7 root     root          4096 Jul 18 15:34 lib
drwxr-xr-x    5 root     root          4096 Jul 18 15:34 media
drwxr-xr-x    2 root     root          4096 Jul 18 15:34 mnt
drwxr-xr-x    2 root     root          4096 Jul 18 15:34 opt
dr-xr-xr-x  206 root     root             0 Aug  2 10:47 proc
drwx------    2 root     root          4096 Jul 18 15:34 root
drwxr-xr-x    2 root     root          4096 Jul 18 15:34 run
drwxr-xr-x    2 root     root          4096 Jul 18 15:34 sbin
drwxr-xr-x    2 root     root          4096 Jul 18 15:34 srv
dr-xr-xr-x   11 root     root             0 Aug  2 10:47 sys
drwxrwxrwt    2 root     root          4096 Jul 18 15:34 tmp
drwxr-xr-x    7 root     root          4096 Jul 18 15:34 usr
drwxr-xr-x   12 root     root          4096 Jul 18 15:34 var

F:\DockerContainers\doctest1>
```

What happened? Behind the scenes, a lot of stuff happened. When you call run,

- The **Docker client** contacts the **Docker daemon**
- The **Docker daemon** checks **local store** if the **image** (alpine in this case) is available locally, and if not, **downloads** it from Docker Store. (Since we have issued docker pull alpine before, the download step is not necessary)
- The **Docker daemon** creates the **container** and then **runs a command** in that container.
- The **Docker daemon** streams the **output** of the command to the **Docker client**
- When you run docker run alpine, you provided a command (ls -l), so Docker started the command specified and you saw the listing.

Run a static website in a container

- First, we'll use Docker to run a static website in a container. The website is based on an existing image. We'll pull a Docker image from Docker Store, run the container, and see how easy it is to set up a web server.
- The image that you are going to use is a single-page website that was already created for this demo and is available on the Docker Store as dockersamples/static-site. You can download and run the image directly in one go using docker run as follows.
- **docker run -d dockersamples/static-site**

So, what happens when you run this command?

- Since the image doesn't exist on your Docker host, the Docker daemon first fetches it from the registry and then runs it as a container.
- Now that the server is running, do you see the website? What port is it running on? And more importantly, how do you access the container directly from our host machine?

- First, stop the container that you have just launched. In order to do this, we need t... **container ID**.
- Since we ran the container in detached mode, we don't have to launch another terminal to do this. Run **docker ps** to view the running containers.
- Check out the CONTAINER ID column. You will need to use this CONTAINER ID value, a long sequence of characters, to identify the container you want to stop, and then to remove it. The example below provides the CONTAINER ID on our system; you should use the value that you see in your terminal.
- **docker stop 0d9ae56459f4**
- **docker rm 0d9ae56459f4**
- Now, let's launch a container in detached mode as shown below:
- **docker run --name static-site -e AUTHOR="Your Name" -d -P dockersamples/static-site**

- -d will create a container with the process detached from our terminal
- -P will publish all the exposed container ports to random ports on the Docker host
- -e is how you pass environment variables to the container
- --name allows you to specify a container name
- AUTHOR is the environment variable name and Your Name is the value that you can pass

Now you can see the ports by running the docker port command.

- **docker port static-site**
- You can now open http://localhost:49154/, if the port is 49154.
- You can also run a second webserver at the same time, specifying a custom host port mapping to the container's webserver.
- **docker run --name static-site-2 -e AUTHOR="Sunitha" -d -p 8888:80 dockersamples/static-site**
- To deploy this on a real server you would just need to install Docker, and run the above docker command(as in this case you can see the AUTHOR is Docker which we passed as an environment variable).

Questions ???