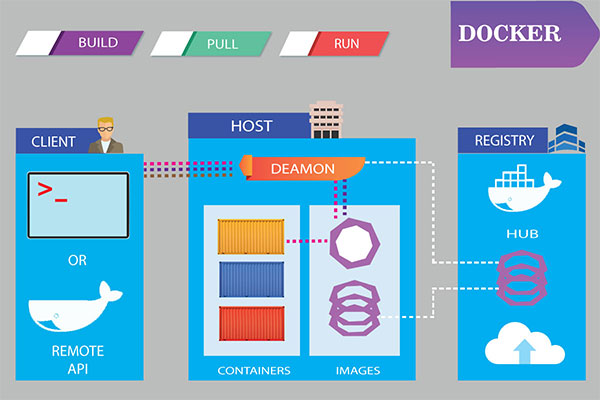
**What is Docker?**

* Docker is an open-source containerization platform/tool
* Docker is an in-demand, DevOps technology
* Docker is for Build, Ship, and Run Any App, Anywhere using containers.
* Docker is a bit like a virtual machine.
* Docker is a tool designed to make it easier to deploy, and run applications by using containers.
* Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.
* dockerizing, an application means deploying and running an application using containers.
* The initial release of Docker was in March 2013.

**Why Docker?**

* + **Docker** enables developers to easily pack, ship, and run any application as a lightweight, portable, self-sufficient container
  + Docker containers are lightweight and modular.



**When do I use Docker?**

* Using Docker, you can move your application from a local development server to a production server without any error.
* Docker is highly recommended for all application development. Docker offers so many tools (software such as NodeJS, PHP, Java, Any Database, etc) as images.
* Using Docker we can easily install the needed software using Docker and create a container for the application development. Databases are a great fit for Docker.

**Features of Docker**

* Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
* With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
* You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
* Since Docker containers are pretty lightweight, they are very easily scalable.
* With the consistent environment of Docker, the development lifecycle of applications is easy and seamless.

**Docker Advantages**

Docker offers many advantages for developers and DevOps teams.

* is in demand for companies large and small
* Consistency across multiple environments.
* Isolated environments simplify debugging.
* simplifies configuration
* Large community support.
* Containers are lighter and use less resources than virtual machines.
* provides access to thousands of configured images with Docker Hub
* The platform supports CI/CD.

**Tools and terminology commonly used with Docker**

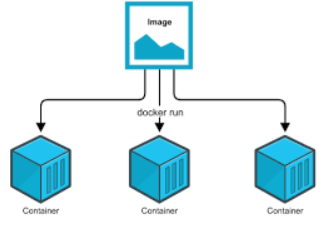
* **Docker Engine** − It is used for building Docker images and creating Docker containers.
  + The runtime environment that allows developers to build and run containers.
* **Docker file:** A simple text file that defines everything needed to build a Docker container image, such as OS network specifications and file locations. It’s essentially a list of commands that Docker Engine will run in order to assemble the image.
* **Docker Hub** − This is the registry which is used to host various Docker images.
* **Docker Compose:** A tool for defining and running multi-container applications. It creates a YAML file to specify which services are included in the application and can deploy and run containers with a single command via the Docker CLI.

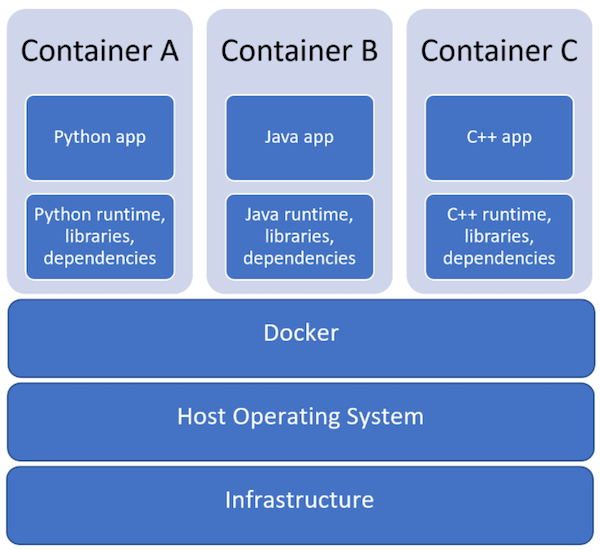
**Docker Images**

1. Images are read-only templates used to create containers using the [Docker build command](https://monkelite.com/build-docker-images-by-using-dockerfile/)
2. It contains a list of commands and instruction on how to build and run a container.
3. So basically Images contains all the data and metadata required to fire up a container.
4. We can't lunch a container without specifying Images.
5. Images may come from two sources:
   1. image repository, which is Docker Hub.
   2. you can create your own images using Dockerfiles.
6. images are stored in a Docker registry such as Docker hub or private repositories such as Azure or AWS.
7. Docker Command: docker images (list all the images)
8. Each image has the following attributes
   1. **TAG** − This is used to logically tag images.
   2. **Image ID** − This is used to uniquely identify the image.
   3. **Created** − The number of days since the image was created.
   4. **Virtual Size** − The size of the image.

**Docker Container**

* A container is a lightweight, resource-controlled runtime environment, running on a host or virtual machine.
* Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package.
* Containers allow us to package all the parts of an application and deploy it as one entity.
* Containers are creating from Images so we can say container is the running instance of an Images.
* Container is a runtime construct.
* You can think of a Docker image as a class, where as a Docker container is an instance of that class.
* inside that container, we have all the libraries/Binaries and dependencies we need to run our application.





**Dockerfile**

* Docker also gives you the capability to create your own Docker images, and it can be done with the help of Docker Files.
* A Docker File is a simple text file with instructions on how to build your images.
* The docker file is a text document that contains all the instructions use to create an image.
* A Dockerfile is a text file that defines a Docker image. You’ll use a Dockerfile to create your own custom Docker image.
* A Dockerfile is a file that you create which in turn produces a Docker image when you build it.
* A Dockerfile is a text file that Docker reads in from top to bottom. It contains a bunch of instructions which informs Docker HOW the Docker image should get built.
* So, the docker can build images automatically by reading the set of instructions from a Dockerfile.
* A Dockerfile must not have any extension and must be named as Dockerfile with capital D.
* Below are other possible commands which can be used in Dockerfile

|  |  |
| --- | --- |
| Command | Description |
| FROM | The first instruction must be FROM to specify the base image from which you are building. |
| COPY | copy files from our local machine into the image. |
| ADD | Copies a file from the host system onto the container |
| CMD | CMD command is used to display a message to the user. |
| ENTRYPOINT | tells the container what to run when it starts up |
| ENV | Sets an environment variable in the new container |
| EXPOSE | Opens a port for linked containers |
| MAINTAINER | An optional value for the maintainer of the script |
| ONBUILD | A command that is triggered when the image in the Dcokerfile is used as a base for another image |
| RUN | The RUN command is used to run instructions against the image. |
| USER | Sets the default user within the container |
| VOLUME | Creates a shared volume that can be shared among containers or by the host machine |
| WORKDIR | Set the default working directory for the container  which sets the working directory for other instructions like RUN, COPY, or ENTRYPOINT. |

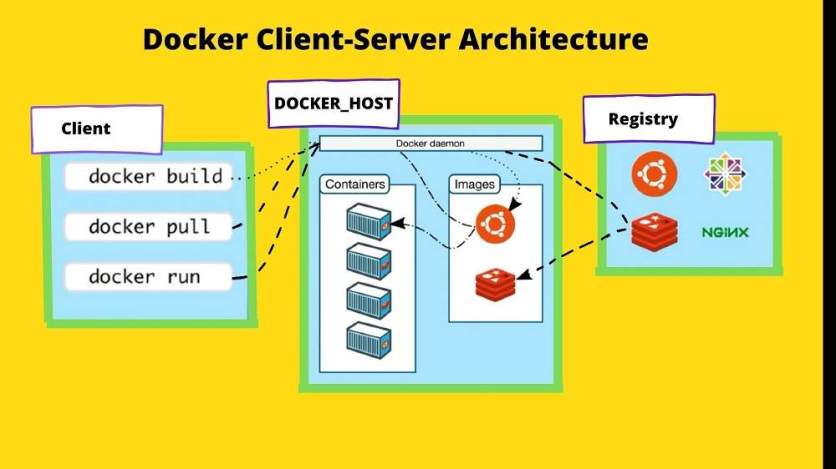
### Docker Registries

1. A registry is where we store our images, you can host your own registry, or you can use Docker’s public registry which is called DockerHub.
2. Inside a registry, images are stored in repositories. Docker repository is a collection of different Docker images with the same name, that have different tags, each tag usually represents a different version of the image.

### Docker Hub

1. [Docker hub](https://monkelite.com/push-docker-image-into-docker-hub/) is a public registry that contains a large number of images you can use.
2. Docker Hub is a registry service on the cloud that allows you to download Docker images that are built by other communities.
3. You can also upload your own Docker built images to Docker hub.
4. The official site for Docker hub is − [https://hub.docker.com](https://hub.docker.com/)
5. Images can be downloaded from Docker Hub using the Docker run command.
   1. Docker run hello-world
6. Pull the images from Docker Hub
   1. docker pull hello-world
   2. docker pull nginx
   3. docker pull Jenkins

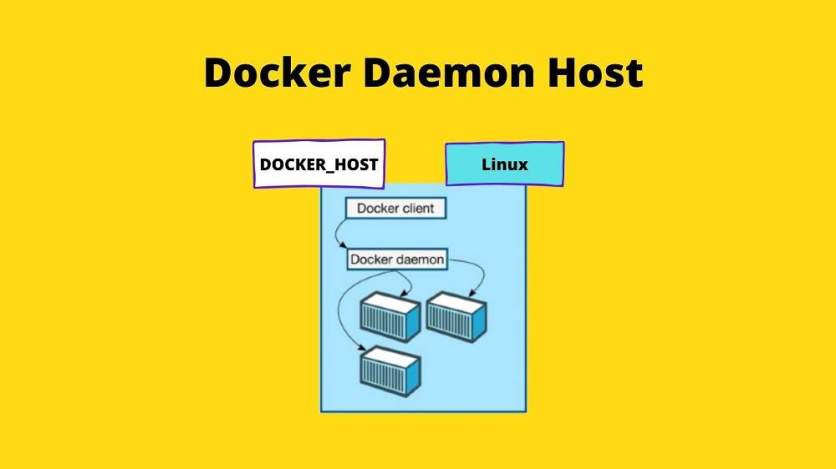
## **Docker Software’s Client-Server Architecture**



* Docker technology uses a client-server architecture with the daemon being the server(**The Docker daemon** is a service that runs on your [host operating system](https://nickjanetakis.com/blog/comparing-virtual-machines-vs-docker-containers)).
* The user interacts with the daemon through the Docker client. So, the docker client is the primary user interface to Docker. It receives commands from the user and communicates back and forth with a Docker daemon.
* The client and daemon (Docker host) does not need to be on the same box.
* There are two types of Docker clients.
  + The typical command-line client and Kitematic which is a Docker client with a graphical interface. So if you don’t like working with commands, kinematic is something you should check out.

**Docker Daemon**

* Docker daemon is often referred to as the Docker engine or Docker server.
* The daemon is the persistent process that does the heavy lifting of building, running, and distributing your Docker containers.



* On a typical Linux installation, the Docker client, the Docker daemon, and any containers run on the same host.
* we can also directly connect a Docker client to a remote Docker daemon. But you can’t run Docker natively in OSX or Windows. Because Docker daemon uses Linux-specific kernel features.
* So, on OSX or Windows installation, the Docker daemon is running inside a Docker machine. The Docker machine is a lightweight Linux VM made specially to run the Docker daemon on OSX or Windows.

Note:

* Docker is only designed to run on Linux kernel version 3.8 and higher.