**What is Docker?**

Docker is a set of Platforms as a service (PaaS) products that use Operating system-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and therefore use fewer resources than a virtual machine.

**Table of Content**

* [What is Docker?](https://www.geeksforgeeks.org/introduction-to-docker/#what-is-docker)
* [Key Components of Docker](https://www.geeksforgeeks.org/introduction-to-docker/#key-components-of-docker)
* [What is a Dockerfile?](https://www.geeksforgeeks.org/introduction-to-docker/#what-is-a-dockerfile)
* [What is Docker Architecture and How Docker Works?](https://www.geeksforgeeks.org/introduction-to-docker/#what-is-docker-architecture-and-how-docker-works)
* [What is Docker Image?](https://www.geeksforgeeks.org/introduction-to-docker/#what-is-docker-image)
* [What is Docker Container?](https://www.geeksforgeeks.org/introduction-to-docker/#what-is-docker-container)
* [What is Docker Hub?](https://www.geeksforgeeks.org/introduction-to-docker/#what-is-docker-hub)
* [What is Docker Compose?](https://www.geeksforgeeks.org/introduction-to-docker/#what-is-docker-compose)
* [Docker Commands](https://www.geeksforgeeks.org/introduction-to-docker/#docker-commands)
* [Docker Engine](https://www.geeksforgeeks.org/introduction-to-docker/#dockerengine)
* [How to Install Docker On Ubuntu? A Step-By-Step Guide](https://www.geeksforgeeks.org/introduction-to-docker/#install-docker-on-ubuntu)
* [Docker Security](https://www.geeksforgeeks.org/introduction-to-docker/#docker-security)
* [Docker Vs Kubernetes](https://www.geeksforgeeks.org/introduction-to-docker/#docker-vs-kubernetes)

**What is Docker?**

[Docker](https://www.geeksforgeeks.org/introduction-to-docker/)is an open-source [containerization](https://www.geeksforgeeks.org/containerization-using-docker/)platform by which you can pack your application and all its dependencies into a standardized unit called a container. Containers are light in weight which makes them portable and they are isolated from the underlying infrastructure and from each other container. You can run the [docker image](https://www.geeksforgeeks.org/what-is-docker-image/)as a [docker container](https://www.geeksforgeeks.org/virtualisation-with-docker-containers/)in any machine where docker is installed without depending on the [operating system.](https://www.geeksforgeeks.org/what-is-an-operating-system/)

**Why Docker is popular?**

Docker gained its popularity due to its impact on the software development and deployment. The following are the some of the main reasons for docker becoming popular:

1. **Portability:**Docker facilitates the developers in packaging their applications with all dependencies into a single lightweight containers. It facilities in ensuring the consistent performance across the different computing environments.
2. **Reproducibility:**Through encapsulating the applications with their dependencies within a container it ensures in software setups remaining consistent across the development, testing and production environments.
3. **Efficiency:**Docker through its container based architecture it optimizes the resource utilization. It allows the developers to run the multiple isolated applications on a single host system.
4. **Scalability:**Docker’s scalability features facilitated the developers in making easier of their applications handling at time of workloads increment.

*Understanding Docker’s core concepts is essential, but practical experience is what truly sets you apart. Platforms like* [***Hostinger***](https://www.hostg.xyz/aff_c?offer_id=545&aff_id=115172&url_id=4598) *make it easy to deploy Docker containers, allowing you to focus on developing and testing your applications. Hostinger’s* ***scalable infrastructure*** *provides an ideal environment for learning and experimenting with Docker in a real-world setting. Their* ***seamless integration******with Docker containers*** *ensures that whether you’re running simple apps or complex multi-container setups, you can deploy with ease.*

**Key Components of Docker**

The following are the some of the key components of Docker:

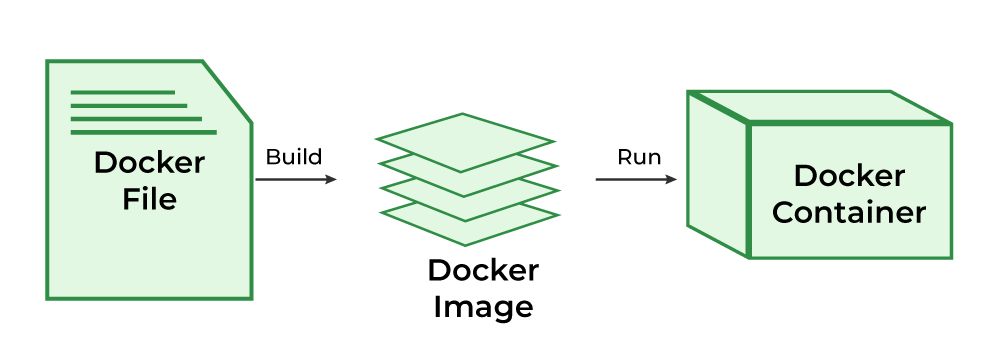
* **Docker Engine:**It is a core part of docker, that handles the creation and management of containers.
* **Docker Image:**It is a read-only template that is used for creating containers, containing the application code and dependencies.
* **Docker Hub:**It is a cloud based repository that is used for finding and sharing the container images.
* **Dockerfile:**It is a script that containing instructions to build a docker image.
* **Docker Registry**: It is a storage distribution system for docker images, where you can store the images in both public and private modes.

**What is a Dockerfile?**

The [Dockerfile](https://www.geeksforgeeks.org/what-is-dockerfile/" \t "_blank)uses DSL (Domain Specific Language) and contains instructions for generating a Docker image. Dockerfile will define the processes to quickly produce an image. While creating your application, you should create a Dockerfile in order since the **Docker daemon**runs all of the instructions from top to bottom.

(The Docker daemon, often referred to simply as “Docker,” is a background service that manages Docker containers on a system.)

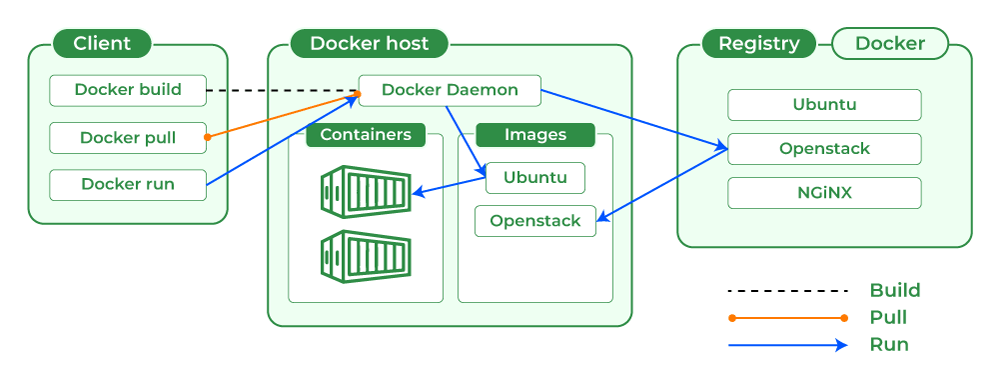
* It is a text document that contains necessary commands which on execution help assemble a Docker Image.
* Docker image is created using a Dockerfile.



To Know more about the Dockerfile refer to the [Docker – Concept of Dockerfile](https://www.geeksforgeeks.org/docker-concept-of-dockerfile).

**What is Docker Architecture and How Docker Works?**

Docker makes use of a client-server architecture. The Docker client talks with the docker daemon which helps in building, running, and distributing the docker containers. The Docker client runs with the daemon on the same system or we can connect the Docker client with the Docker daemon remotely. With the help of REST API over a UNIX socket or a network, the docker client and daemon interact with each other. To know more about working of docker refer to the [Architecture of Docker](https://www.geeksforgeeks.org/architecture-of-docker).



**What is Docker Image?**

It is a file, comprised of multiple layers, used to execute code in a Docker container. They are a set of instructions used to create docker containers. Docker Image is an executable package of software that includes everything needed to run an application. This image informs how a container should instantiate, determining which software components will run and how. Docker Container is a virtual environment that bundles application code with all the dependencies required to run the application. The application runs quickly and reliably from one computing environment to another.

**What is Docker Container?**

Docker container is a runtime instance of an image. Allows developers to package applications with all parts needed such as libraries and other dependencies. Docker Containers are runtime instances of Docker images. Containers contain the whole kit required for an application, so the application can be run in an isolated way. For eg.- Suppose there is an image of Ubuntu OS with NGINX SERVER when this image is run with the docker run command, then a container will be created and NGINX SERVER will be running on Ubuntu OS.

**What is Docker Hub?**

Docker Hub is a repository service and it is a cloud-based service where people push their Docker Container Images and also pull the Docker Container Images from the Docker Hub anytime or anywhere via the internet. Generally it makes it easy to find and reuse images. It provides features such as you can push your images as private or public registry where you can store and share Docker images.

Mainly DevOps team uses the Docker Hub. It is an open-source tool and freely available for all operating systems. It is like storage where we store the images and pull the images when it is required. When a person wants to push/pull images from the Docker Hub they must have a basic knowledge of Docker. Let us discuss the requirements of the Docker tool.

**What is Docker Compose?**

Docker Compose will execute a YAML-based multi-container application. The YAML file consists of all configurations needed to deploy containers [Docker Compose](https://www.geeksforgeeks.org/installation-of-docker-compose/), which is integrated with [Docker Swarm](https://www.geeksforgeeks.org/introduction-to-docker-swarm-mode/), and provides directions for building and deploying containers. With Docker Compose, each container is constructed to run on a single host.

**How to Download Docker Desktop?**

Docker Desktop provides GUI to work on docker containers, docker images and docker networks. Docker desktop provides and separate environment which contains Docker Engine, Docker CLI, Docker Compose, [Kubernetes,](https://www.geeksforgeeks.org/introduction-to-kubernetes-k8s/)and other tools which are needed to build, ship and run the applications in the form of containers which makes it more user friendly. To know more how to install docker desktop refer to [Docker Desktop Sample Image.](https://www.geeksforgeeks.org/docker-desktop-sample-image)

**Docker Commands**

Through introducing the essential docker commands, docker became a powerful software in streamlining the container management process. It helps in ensuring a seamless development and deployment workflows. The following are the some of docker commands that are used commonly:

* **Docker Run:**It used for launching the containers from images, with specifying the runtime options and commands.
* **Docker Pull:**It fetches the container images from the container registry like Docker Hub to the local machine.
* **Docker ps**: It helps in displaying the running containers along with their important information like container ID, image used and status.
* **Docker Stop**: It helps in halting the running containers gracefully shutting down the processes within them.
* **Docker Start:**It helps in restarting the stopped containers, resuming their operations from the previous state.
* **Docker Login:**It helps to login in to the docker registry enabling the access to private repositories.

To Know more about the docker commands refer tot the [Docker – Instruction Commands](https://www.geeksforgeeks.org/docker-instruction-commands).

**Docker Engine**

The software that hosts the containers is named Docker Engine. Docker Engine is a client-server based application. The docker engine has 3 main components:

1. **Server:**It is responsible for creating and managing Docker images, containers, networks, and volumes on the Docker. It is referred to as a daemon process.
2. [**REST API**](https://www.geeksforgeeks.org/rest-api-introduction/)**:**It specifies how the applications can interact with the Server and instructs it what to do.
3. **Client:**The Client is a docker command-line interface (CLI), that allows us to interact with Docker using the docker commands.

**Why to use Docker?**

Docker can be used to pack the application and its dependencies which makes it lightweight and easy to ship the code faster with more reliability. Docker make its very simple to run the application in the production environment docker container can be platform independent if the docker engine is installed in the machine.

* **Resource Efficiency**: Docker helps in maximizing the resource utilization by running the multiple containers on a single host. It helps in reducing the infrastructure costs and improves the efficiency.
* **Version Control:**It simples the versioning for the applications and their dependencies ensuring the consistency and making easier of collaboration across the teams.
* **Microservices Agility:**It enables the adoption of microservices architecture, promoting the scalability, flexibility and fault isolation agile application development.

**What is Docker For AWS?**

Docker is the most powerful tool to run the application in the form of containers. Docker container are light in weight and can be run on any operating system.

AWS provides the Amazon Elastic Container Service ( [Amazon ECS](https://www.geeksforgeeks.org/introduction-to-amazon-elastic-container-service-ecs/)) it is an fully managed container service by which you can deploy, scale and manage the docker containers. Amazon ECS is the most reliable platform according to the performance and also it can be integrated with the other AWS Service like load balancing, service discovery, and container health monitoring. To know more about [Amazon Elastic Container Service (Amazon ECS)](https://www.geeksforgeeks.org/introduction-to-amazon-elastic-container-service-ecs).

**Difference Between Docker Containers and Virtual Machines**

The following are the differences between docker containers and Virtual Machines:

| **Docker Containers** | **Virtual Machines** |
| --- | --- |
| Docker Containers contain binaries, libraries, and configuration files along with the application itself. | Virtual Machines (VMs) run on Hypervisors, which allow multiple Virtual Machines to run on a single machine along with its own operating system. |
| They don’t contain a guest OS for each container and rely on the underlying OS kernel, which makes the containers lightweight. | Each VM has its own copy of an operating system along with the application and necessary binaries, which makes it significantly larger and it requires more resources. |
| Containers share resources with other containers in the same host OS and provide OS-level process isolation. | They provide Hardware-level process isolation and are slow to boot. |

**How to Install Docker On Ubuntu? A Step-By-Step Guide**

The following steps guide you in installation of docker on ubuntu:

**Step 1: Remove old version of Docker**

* Execute the following command to remove the old versioned docker software:

$ sudo apt-get remove docker docker-engine docker.io containerd runc

**Step 2: Installing Docker Engine**

* The following command is used for installation of docker engine:

$ sudo apt-get update  
   
$ sudo apt-get install \  
 ca-certificates \  
 curl \  
 gnupg \  
 lsb-release  
$ sudo mkdir -p /etc/apt/keyrings  
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
$ echo \  
 "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \  
 $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
   
$ sudo apt-get update  
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin  
$ sudo groupadd docker  
$ sudo usermod -aG docker $USER

**Step 3: Verify Docker Installation**

* Check if docker is successfully installed in your system by trying to run a container with the following command:

$ sudo docker run hello-world

To more detailing on the installation of docker on ubuntu, refer this – [Article](https://www.geeksforgeeks.org/how-to-install-and-configure-docker-in-ubuntu/)

**Sample Example: Containerizing Application Using Docker**

The following steps guides in containerizing the application using Docker:

**Step 1: Create Dokcerfile and Python Application**

* Create a folder with 2 files (Dockerfile and main.py file) in it

**Step 2: Develop Python Code**

* Edit main.py with the below code, or else you can try on developing your own python code.

*#!/usr/bin/env python3*

print("Docker and GFG rock!")

**Step 3: Develop a Dockerfile**

* Edit Dockerfile with the below instructions for developing the Dockerfile:

FROM python:latest  
COPY main.py /  
CMD [ "python", "./main.py" ]

**Step 4: Create a Docker Image**

* Once you have created and edited the main.py file and the Dockerfile, create your image contain your application by running the following command:

$ sudo docker build -t python-test .

* The ‘-t’ option allows to define the name of your image. ‘python-test’ is the name we have chosen for the image.

**Step 5: Run Docker Container**

* Once the image is created, your code is ready to launch.

$ sudo docker run python-test

To know more about this containerization of your application, refer this – [Article](https://www.geeksforgeeks.org/containerization-using-docker/)

**Sample Example to Push an image to Docker Hub**

The following steps guides in pushing an image to the Dockerhub:

**Step 1:**Create an account on Docker Hub or use an existing one if you already have one.

**Step 2:**Click on the “Create Repository” button, put the name of the file, and click on “Create”.

**Step 3:**Now will “tag our image” and “push it to the Docker Hub repository” which we just created.

* Now, run the below command to list docker images:

$ docker images

* The above will give us this result

REPOSITORY TAG IMAGE\_ID CREATED SIZE afrozchakure/python-test latest c7857f97ebbd 2 hours ago 933MB

* Image ID is used to tag the image. The syntax to tag the image is:

docker tag <image-id> <your dockerhub username>/python-test:latest  
$ docker tag c7857f97ebbd afrozchakure/python-test:latest

**Step 4:**Push image to Docker Hub repository

$ docker push afrozchakure/python-test

To know more about publishing the images to dockerhub, refer this – [Article](https://www.geeksforgeeks.org/docker-publishing-images-to-docker-hub/)

**Fetch and run the image from Docker Hub**

**1.**To remove all versions of a particular image from our local system, we use the Image ID for it.

$ docker rmi -f af939ee31fdc

**2.**Now run the image, it will fetch the image from the docker hub if it doesn’t exist on your local machine.

$ docker run afrozchakure/python-test

**Importance of Docker**

The following are the some of the insights that discusses on the importance of docker:

* **Efficiency and Speed**: It facilitates with providing the streamlined development and deployment by packaging applications with dependencies into consistent containers.
* **Resource Optimization**: It helps in sharing host system resources efficiently, allowing for higher application density and cost savings.
* **Scalability and Portability**: It is easily able to scale the applications and ensures seamless movement across different environments.
* **Isolation and Security**: it provides high isolation, reducing conflicts and enhancing security.

**Benefits of Docker**

The following are the some of the benefits of Docker:

* **Portability:**Docker facilities with creation of lightweight portable containers that can be unable on any machine regardless of the underlying operating systems.
* **Isolation:**Docker through containers provides a high level of isolation with enabling the applications to run independently of each other addressing the issues that one container doesn’t impact on other.
* **Reproducibility:**With, Docker developers can easily package their applications and their dependencies into a reusable images. It allows for consistent and repoduciable builds across the development, testing and production environments.
* **DevOps Integration**: It promotes the collaboration and automation across the software development life cycle in handing the increasing workloads.

**Alternatives of Docker**

The following are the alternatives of Docker:

* **Podman**: Offers a Docker-compatible container engine with a focus on security and compatibility, ideal for environments where Docker is not preferred or available.
* **rkt:**A lightweight container runtime developed by CoreOS, designed for simplicity, security, and composability, offering an alternative to Docker’s container runtime.
* **LXC (Linux Containers):**Provides operating-system-level virtualization for running multiple isolated Linux systems (containers) on a single host, offering a lightweight alternative to Docker for certain use cases.
* **containerd:**An industry-standard core container runtime developed by Docker, Inc., offering a minimal and stable platform for building containerized applications, often used as a lower-level alternative to Docker for more advanced container orchestration systems like [Kubernetes.](https://www.geeksforgeeks.org/kubernetes-tutorial/)

**Docker Security**

The following are the some of the insights on docker security:

* **Isolation**: Docker containers provides the strong isolation ensuring the applications and processing its running.
* **Immutable Infrastructure**: It promotes the use of immutable infrastructure, where containers are build from the immutable images that are version controlled.
* **Resource Constraints:**It allows you to define the resource constraints for containers such as CPU and memory limits.
* **Security Scanning:**It provides the built-in security scanning tools that allows you to scan the container images for known vulnerabilities and malware before deployment.

**Use Cases of Docker**

The following are the some of the use cases of Docker:

* **Continuous Integration and Continuous Deployment (CI/CD)**: It helps in streamlining and automating the software delivery process with ensuring faster and more reliable releases.
* **Microservices Architecture:**It facilitates the development, deployment, and management of microservices with enabling independent scaling and maintenance.
* **Development Environment Consistency:**It will ensures with consistent development, testing, and production environments, reducing “it works on my machine” issues.
* **Multi-Cloud and Hybrid Cloud Deployments:**It helps in simplifying the application deployment across various cloud platforms, enhancing flexibility and reducing vendor lock-in.

**Best Practices and Tips for using Docker Effectively**

The following are the best practices and tips of using docker effectively:

* **Use Official Images:**Try to always start with official images from Docker Hub to ensure security and reliability.
* **Minimize Image Size**: Keep your Docker images as small as possible by using multi-stage builds and removing unnecessary files.
* **Leverage Docker Compose:**Use Docker Compose for managing multi-container applications, simplifying setup and deployment.
* **Implement Proper Security Measures:**Regularly update images, avoid running containers as root, and use network segmentation to enhance security.

**Docker Vs Kubernetes**

The following are the difference between docker and kubernetes:

| **Feature** | **Docker** | **Kubernetes** |
| --- | --- | --- |
| **Primary Purpose** | Containerization platform | Container orchestration platform |
| **Functionality** | Creates and manages containers | Manages and scales containerized applications |
| **Setup Complexity** | Simple to set up and use | More complex setup and configuration |
| **Scalability** | Limited to single-node scaling | Designed for large-scale, multi-node environments |
| **Networking** | Basic networking capabilities | Advanced networking with service discovery and load balancing |
| **State Management** | Stateless; manages individual containers | Stateful; manages container clusters and services |
| **Use Case** | Development and testing environments | Production environments with high scalability and reliability requirements |

To known more the difference between docker and kubernetes, refer this – [Article](https://www.geeksforgeeks.org/kubernetes-vs-docker/)

**Conclusion**

So you have learned about the basics of Docker, the difference between Virtual Machines and Docker Containers along some common terminologies in Docker. Also, we went through the installation of Docker on our systems. We created an application using Docker and pushed our image to Docker Hub. Lastly, we learned how we could remove a particular image from our local system and later pull the image from Docker Hub if it doesn’t exist locally.