THE KNOWLEDGE HOUSE

# MODULE 4

## DOCKER FUNDAMENTALS

### OBJECTIVE

To read and understand the basics of Docker Fundamentals.

### ANSWER THE FOLLOWING QUESTIONS IN A .TXT FILE:

WHAT_IS_DOCKER

- In your own words explain at its core what a computer really is doing whenever you type a letter on a keyboard (250 words)?
- Explain in your own words what are the uses of Docker? (300 Words)
- Explain why is docker recommended (200 Words)
- Explain What are the main features of Dicker: ( 250 Words )

# MODULE 4
## DOCKER FUNDAMENTALS

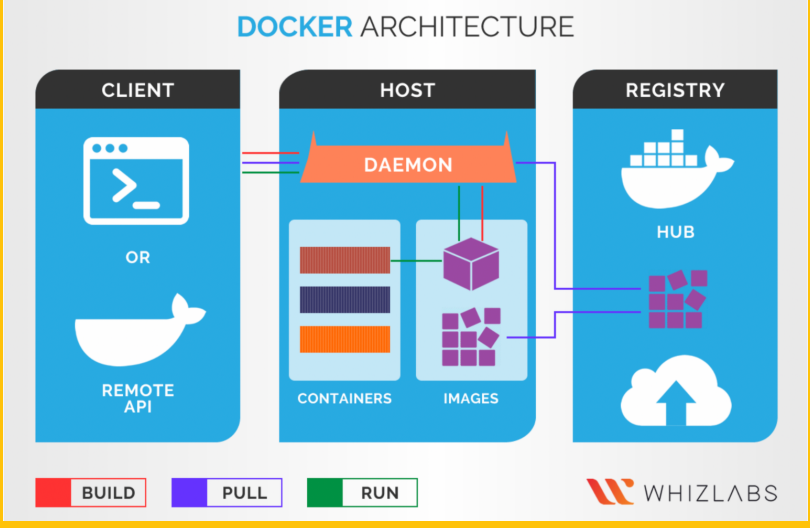
## WHAT IS DOCKER?

To define in simple terms, Docker is a developer tool to assist developers and system admins in deploying applications in containers to effectively run them in host operating systems like Linux. It's an overall platform used to develop, ship, and running applications. It is a significant tool to separate your applications from the applications software structure to deliver the software quickly. The docker's infrastructure can be managed just like software management, and the interface is also easy. Containerization is another popular term tagged with Docker. Containerization is the term coined for using containers to deploy applications in a Linux platform. It was nascent in the tech world in the year 2013, and from then it is one stable content for modern developments in technology.

**Uses of Docker are:**

- It helps to use the tooling to create and use containers for developing your required applications.
- The container becomes an entire unit that can be used for distributing and testing the developed applications.
- The system can use deployed and tested locally, on cloud services or even tested on a hybrid mix of the two.

## UNDERSTANDING THE DOCKER ARCHITECTURE



# MODULE 4
## DOCKER FUNDAMENTALS

## UNDERSTANDING THE DOCKER ARCHITECTURE CONT.

Docker architecture follows the client-server model. The Docker client communicates with the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers.

**Docker architecture has several components like:**

- **Docker Client (CLI):** a facility used to trigger docker commands
- **Docker Host:** command for running docker daemon
- **Dockery Registry:** or also termed as docker hub is a registry for storing docker images

A docker daemon, when allowed to execute with a docker host, is responsible for the formation and running of images and containers.

**You need to know the following points to understand how docker architecture works.**

- A build command that is issued from a suitable client will trigger the docker daemon to build a Docker image. The docker daemon runs on the docker host which runs the images based on the inputs given by the clients. Once the image is built it is saved in the registry of a docker hub and can be retrieved when required. The docker hub can also be a local repository or cloud-based storage.
- If you do not wish to proceed with the process of creating an image, you can just pull out an image already present in the registry. A different user of the community may have uploaded this image.
- Run command from the client will run the docker image, and for the same, a docker container is created and used.

If you are aspiring to become a **Docker Certified Associate**, start your preparation now with the Docker Certified Associate Training Course and Practice Tests.

## WHY IS IT RECOMMENDED TO GO FOR DOCKER?

Unlike virtual machines, docker does not induce version mismatches when running in different setups. There is no loss in the time and the efforts spent by the developer in creating and running, unlike virtual machines.

# MODULE 4
## DOCKER FUNDAMENTALS

## WHY IS IT RECOMMENDED TO GO FOR DOCKER? CONT.

**On a comparative tone between a virtual machine and a docker container, there are three parameters that decide its functionality:**

- **Size:** the base of this comparison falls on the amount of resource that is utilized by Docker or virtual machines while running. Dockers comparatively uses lesser storage space, which can be reutilized for creating more containers while that is not the case with virtual machines.
- **Startup:** On the amount of boot time utilized, this comparison is made. Since the operation of the guest server starts from scratch, boot time is higher in the case of virtual machines.
- **Integration:** The basis of this comparison is the ability to integrate with other tools easily. DevOps tools in VM are very limited, and so is its functionality. However, in docker, several instances can be set, making its functionality easier.

## WHY ARE DOCKER CONTAINERS POPULAR?

Containerization and Docker containers are very popular terms used invariably by techies around the globe.

**And it is identified to be equally useful with the number of features it has to offer which includes:**

- It offers high flexibility in usage. Any type of complex applications can be containerized and that too easily.
- Updates and upgrades on the deploy can be interchanged easily.
- The docker can be built locally, upgraded to be deployed to the cloud and can be run anywhere. The portability feature of docker is very high.
- Container replicas can be created and scaled easily and increased with efficiency.
- On the fly- the docker services can be stacked vertically.

# MODULE 4
## DOCKER FUNDAMENTALS

## FEATURES OF DOCKER

The docker facility allows users to condense the developed application size and assists in producing a smaller trail of the operating system when containers are used. Containers are a system that helps in interacting across the different sectors and borders of the company. With docker, different teams of an organization can correlate and work together
easily.
Docker are featured with an option to run on any platform. It can be used locally, deployed, and tested on cloud platforms also. Further, even a hybrid environment can be used for running and testing the application. Further, scalability is not at all an issue with docker and deploying containers. Docker are lightweight files, making them very cost-effective and space-friendly. Taking up lesser space, you can use the saved capacity for other work-related goals. It's suitable for smaller to medium size deployments in work-related issues.

## WHY ARE DOCKER CONTAINERS POPULAR?

Docker fundamentals are easier to handle, and you don't have to be a pro for it. For getting started with docker, there are no prerequisites or skills required. A piece of basic knowledge with cloud services and web applications development will help you through the process. However, it is not a mandatory thing.

**The entire docker tutorial for installation is listed out here.**

- Docker can be supported across different OS platforms. Setting up of docker in your computer with required tooling can be easy. Initially, there were laid back problems faced with OSX and windows which was them testified to make it work like never before. So start with the docker install. A prompt "hello world" proves you that the installation is done correctly and your docker application is running fine.
- Before the actual installation, its good to ensure that the Linux kernel 3.8 above is installed in your personal computer. Docker are supported in such versions or higher. Install the OS with latest versions helps you work better.
- The next step would be to add certifications needed for working of the docker by installing them. Installing necessary packages can help in running easily and smoothly.
- 4. Adding a GPG key will be the next step in the Docket tutorials for encrypting all the data.

# MODULE 4
## DOCKER FUNDAMENTALS

### DOCKER FILES

The file which contains the sequence of commands for creating a docker image is termed as docker files. These files are provided with a suitable functional name and are executed with a docker command; this results in the docker images. This setup after installation forms a part of the docker fundamentals and a quick start to learning docker. When the docker image is set to run using the “docker run” command, the application may start execution by itself locally or on the cloud as per requirement.

A docker hub, on the other hand, is a location to store all the docker images. A docker hub acts like a cloud registry that holds data of all uploaded docker images uploaded by several users of a community onto the cloud file. Even you can develop your own docker image that can be uploaded to the docker hub.

A docker composes lets you run several docker container files on a single server system efficiently.

### DOCKER ENGINE

To put it very simple, a docker engine acts as a heart of any docker system.

**A host system that is installed with a docker application is technically termed as a docker engine.**

- A long type of running process in docker termed is defined as a daemon process.
- The actual meaning of a client (CLI) is a command-line interface.
- For virtual communication between CLI client and Docker daemon, a REST API is used.

# MODULE 4
## DOCKER FUNDAMENTALS

### WHY DO WE NEED DOCKER?

Talk to any software developer who has been around for a while, and you'll be sure to hear stories of having to support the same software across multiple environments. Indeed, cross-platform support has been a major challenge for developers ever since the early days of computing.
A common trope heard in software development teams used to be, "Well, it works on my machine! I don't know what's wrong with yours!"
The lack of consistency across environments made software support a nightmare. While smart developers created dependency management systems and virtual environments, it was still difficult to guarantee that everyone had a consistent operating system and environment settings.
Docker solves that problem with virtualization. Now, instead of running the application locally, you run the container locally and the container runs the application. Docker takes care of all the work involved in creating a consistent virtualized, containerized environment, no matter what operating system your computer uses.

### IS IT A VIRTUAL MACHINE?

One misconception is that Docker is basically a virtual machine. It's not. Docker is different. Virtual machines are an older concept that has been around for a while. In effect, running a virtual machine involves installing an entire operating system that runs dual-booted, in parallel, via a hypervisor, or instead of your native operating system. In contrast, Docker containers are not full installations of operating systems. They just require access to your native OS's kernel, memory, and other functionality. The Docker container engine works in tandem with your native OS without needing a full install of the container's OS.

### HOW DO I USE DOCKER?

You can get started by following the download instructions for your operating system on the Docker website. Docker is an open-source technology, so anybody can use it and play around with it at no cost. Once you've installed Docker, you can check your installation by running $ docker run hello-world. Docker will first look to see if you have a container image (the blueprint for how to spin up a Docker container) called "hello-world" locally. If it doesn't find it on your local machine (and it shouldn't because you just freshly installed Docker), then it will go to Docker Hub online to pull a container image from the web. Docker Hub has all kinds of useful pre-designed container images for projects. They're worth checking out if you're new. But for now, let's see what goes inside a container image.

# MODULE 4
## DOCKER FUNDAMENTALS

## WHAT DO I DO TO MAKE DOCKER WORK?

First, you'll need an application to run. Most commonly, Docker is used with web applications that will run on a server or via a cloud provider like AWS, GCP, or Azure. I'll assume you have a web application already created and that it lives in a file called App.py, App.js, App.java, or something like that.

**There are three key steps to running your application inside a Docker container:**

- **Create a Dockerfile**

  Now, we need to tell Docker about that application and the dependencies it requires. So, we create a Dockerfile in the same directory as our application. The Dockerfile is the blueprint for how Docker should construct the container and run your application. If we were running a Python application, the file would look something like this:

```
# Inherit from the Python Docker image
FROM python:3.7-slim
# Install the web framework “Flask” via the package manager, pip
RUN pip install flask==1.0.2
# Copy the source code to app folder
COPY ./app.py /app/
# Change the working directory
WORKDIR /app/
# Set "python" as the entry point
ENTRYPOINT ["python"]
# Set the command as the script name
CMD ["app.py"]
```

  Docker will build the application using these instructions. In other languages, the process is similar. Inherit from the language's existing Docker image hosted on Docker Hub, install any dependencies, copy the source code into the Docker app, then set an entry point and a command to start the application.

# MODULE 4
## DOCKER FUNDAMENTALS

## WHAT DO I DO TO MAKE DOCKER WORK? CONT.

- **Build the Container**

  Now that we have our Docker file ready, we need to give Docker the time to pull together all the resources and dependencies we specified. This is the part where Docker creates the containerized environment to run our application. Since it involves downloading and setting up dependencies, it can sometimes take a while depending on the complexity of your application.
  To build a new container, type: docker build -t my_app:0.1
  Docker will look for a Docker file in the current directory, then it will start to build the container based on those instructions. The -t flag tells Docker what we want to name the new container with the syntax <name>:<tag> – the tag is usually a version number.

- **Run the Container**

  Now that Docker has the environment and dependencies ready, we can run our application inside the new container: docker run -d my_app:0.1
  Docker will try to start up your application using the command you specified in the Dockerfile. The -d flag tells Docker to run the application in "detached" mode so that it doesn't take over your terminal session. You can always see what applications Docker is running by typing: docker ps hat

## WHAT DID THAT DO?

Well, it didn't change your application in any way. Docker doesn't place limits on the types of applications you can build. If you can build it on your local machine, you can build it as a Docker container. What changed is the environment in which your application is running. If you get your application working correctly on a Docker container locally, you can be sure it will work on any computer in the world that has Docker installed. That peace of mind is priceless when you consider the variety of operating systems and deployment options there are for your application.

## CAN I MAKE IT MORE COMPLEX?

It's nice that Docker can reliably run a single file, App.py or App.js or whatever. But can it reliably run a complex web application that has an app, database, cache, worker tasks, and more? The answer is yes! Docker includes a set of tools called docker-compose that allow you to work with multiple containers at once, running different services alongside one another and allowing them to talk. Implementing docker-compose is beyond the scope of this article, but it's an important next step in learning to use Docker effectively.