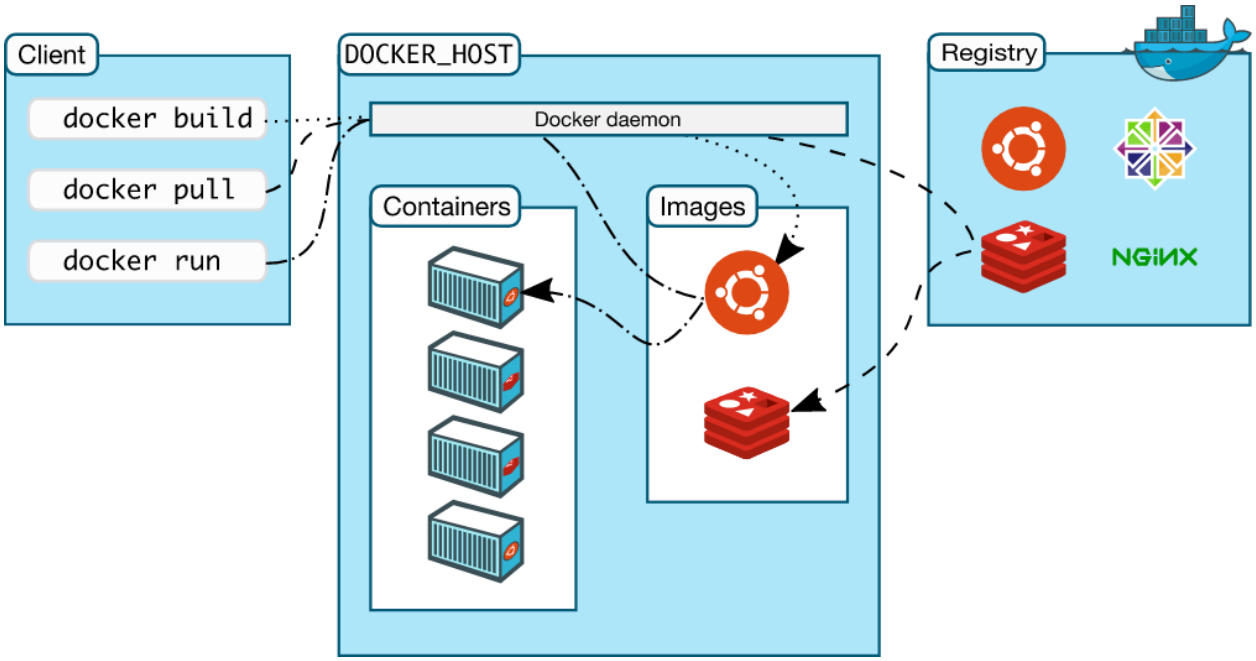
DOCKER

Docker is a platform for developing, shipping, and running applications in containers. Containers are lightweight, portable, and self-sufficient units that can run applications and their dependencies isolated from the host system and other containers.



1. **Containerization:**
    - Docker uses containerization technology to encapsulate applications and their dependencies into containers. This ensures that the application runs consistently across different environments.
2. **Docker Image:**
    - A Docker image is a lightweight, standalone, and executable package that includes the application code, runtime, libraries, and other settings required to run the application. Images are used to create containers.
3. **Docker Container:**
    - A Docker container is an instance of a Docker image. Containers run applications in isolated environments, preventing conflicts with other applications or the host system. They are portable and can run consistently across various environments.

4. **Docker Hub:**
    - Docker Hub is a cloud-based registry service provided by Docker. It serves as a central repository for Docker images. Users can publish and share their Docker images on Docker Hub, and others can pull and use those images.
5. **Dockerfile:**
    - A Dockerfile is a script that defines the steps to build a Docker image. It includes instructions for installing dependencies, configuring the environment, and setting up the application. Docker images are created based on Dockerfiles.
6. **Docker Compose:**
    - Docker Compose is a tool for defining and running multi-container Docker applications. It allows you to define a multi-container environment in a YAML file, specifying services, networks, and volumes, and then launch the entire environment with a single command.
7. **Orchestration:**
    - Docker provides orchestration tools, such as Docker Swarm and Kubernetes, to manage and scale containers in a production environment. These tools automate deployment, scaling, and management of containerized applications.
8. **Portability:**
    - Docker containers are portable and can run consistently across different environments, from a developer's laptop to a production server. This eliminates the "it works on my machine" problem and streamlines the deployment process.

Docker has become a popular technology in the world of DevOps and containerization, providing a standardized and efficient way to package, distribute, and deploy applications. It simplifies the development and deployment process, making it easier for teams to collaborate and ensure consistency in different environments.

What Is Docker Used For?

Docker is used for:

- Running multiple workloads on fewer resources.
- Isolating and segregating applications.
- Standardizing environments to ensure consistency across development and release cycles.

- Streamlining the development lifecycle and supporting CI/CD workflows.
- Developing highly portable workloads that can run on multi-cloud platforms.

Additionally, it is used as:

- A cost-effective alternative to virtual machines.
- A version control system for an application.

Creating a Docker Hub account

Creating a Docker Hub account is a straightforward process. Docker Hub is a cloud-based registry service provided by Docker, where you can store and share Docker images.

1. **Visit the Docker Hub Website:**
    - Open your web browser and go to the Docker Hub website: https://hub.docker.com/
2. **Click on "Sign Up":**
    - On the Docker Hub homepage, click on the "Sign Up" button. This is usually located in the upper right corner of the page.
3. **Complete the Registration Form:**
    - Fill in the required information in the registration form. This typically includes your desired Docker ID, email address, and a strong password.
4. **Verify Your Email:**
    - After submitting the registration form, Docker Hub will send a verification email to the email address you provided. Check your email and click on the verification link to confirm your account.
5. **Set Up Your Profile:**

- Once your email is verified, you can log in to Docker Hub using your Docker ID and password. After logging in, you have the option to set up your profile by adding a profile picture and other details.

6. **Explore Docker Hub:**

    - After creating your Docker Hub account, you can explore Docker Hub to find existing Docker images, contribute your own images, and collaborate with other developers.

That's it! You have successfully created a Docker Hub account. With your account, you can publish your Docker images, pull images from the public registry, and manage your repositories.

Difference between Virtualization and Containerization

Key Factor	**Virtualization**	**Containerization**
Technology	One physical machine has multiple OSs residing on it and appears as multiple machines.	The application developed in a host environment with same OS and the same machine executes flawlessly on multiple different environments.
Start-up Time	Higher than containers	Less
Speed of working	VMs being a virtual copy of the host server on its own operating system, VMs are resource-heavy, hence slower.	Containers are faster.
Size	Larger	Smaller

Component that Virtualizes and the one being virtualized	Hypervisors virtualize the underlying hardware (use of the same hardware).	Containers virtualize the operating system (use of the same OS).
Cost of implementation	Higher	Lower
Benefits for	IT enterprise businesses	Software developers and in turn IT businesses

Docker Commands

docker –version	To get the currently installed version of docker
docker images	Lists all the locally stored docker images
docker search	Searches for specific images through the Docker hub
docker pull Usage: docker pull <image name>	To pull images from the docker repository(hub.docker.com)
docker run Usage: docker run -it <imagename>	To create a container from an image
docker run Usage: docker run -it -d <imagename>	To create a container from an image in detach mode
docker ps	To list the running containers
docker ps -a	To show all the running and exited containers
ctrl + p + q	To exit from container without stopping the container
docker stop cid	Stops a container using the container name or its id
docker start cid	Starts a container using the container name or its id
docker kill cid	Stop the container immediately by killing its execution
docker pause cid	Pause a container using the container name or its id

docker unpause cid	Unpause a container using the container name or its id
docker info	Display system-wide information
docker rename <pre.name/cid> <new name>	Rename the container
docker rm cid	Remove one or more containers
docker top cid	Display the running processes of a container
docker login	Log in to a Docker registry
docker logout	Logout in to a Docker registry
docker attach cid	Attach to a running container/enter into a container
docker commit cid	Create a new image from a container
docker network	To know the details of the list of networks in the cluster
docker logs cid	To view the logs of a running container.
docker rmi iid docker rmi -f iid	To remove image To remove images forcefully if any containers are running in image
docker rm $(docker ps -aq)	Remove all containers at a time
docker rmi $(docker images -q)	Remove all images at a time
docker run -itd -h <hostname> i.name	To give hostname to the container

Dockerfile

The Dockerfile 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.

Dockerfile is the source code of the image

Steps To Create a Dockerfile

- Create a file named Dockerfile.
- Add instructions in Dockerfile.
- Build Dockerfile to create an image.
- Run the image to create a container.

Important Dockerfile Keywords

1. FROM: Represents the base image(OS), which is the command that is executed first before any other commands.

Syntax:

FROM <ImageName>

Example: The base image will be ubuntu:19.04 Operating System.

FROM ubuntu:19.04

2. COPY: The copy command is used to copy the file/folders to the image while building the image.

Syntax:

COPY <Source> <Destination>

Example: Copying the .war file to the Tomcat webapps directory

COPY target/java-web-app.war /usr/local/tomcat/webapps/java-web-app.war

3. ADD: While creating the image, we can download files from distant HTTP/HTTPS destinations using the ADD command.

Syntax:

ADD <URL>

Example: Try to download Jenkins using ADD command

ADD https://get.jenkins.io/war/2.397/jenkins.war

4. RUN: Scripts and commands are run with the RUN instruction. The execution of RUN commands or instructions will take place while you create an image on top of the prior layers (Image).

Syntax:

RUN < Command + ARGS>

Example:

RUN touch file

5. CMD: The main purpose of the CMD command is to start the process inside the container and it can be overridden.

Syntax:

CMD [command + args]

Example: Starting Jenkins

CMD ["java","-jar", "Jenkins.war"]

6. ENTRYPOINT: A container that will function as an executable is configured by ENTRYPOINT. When you start the Docker container, a command or script called ENTRYPOINT is executed. It can't be overridden.

Syntax:

ENTRYPOINT [command + args]

Example: Executing the **echo command.**

ENTRYPOINT ["echo","Welcome "]

7. MAINTAINER: By using the MAINTAINER command we can identify the author/owner of the Dockerfile and we can set our own author/owner for the image.

Syntax:

MAINTAINER <NAME>

Example: Setting the author for the image as a xyz author.

MAINTAINER xyz author

Simple Dockerfile:

FROM <to get base image>

RUN <to execute given commands>

COPY <to copy a file from docker host to docker container>

WORKDIR <To set a working DIR ex: cd >

ENTRYPOINT < to start a process while running a container >

CMD <to start a process while running a container>

EXPOSE <port number>

Tomcat Image creation using Dockerfile

```
FROM amazonlinux

RUN yum install java –y && yum install zip gzip tar -y

WORKDIR /opt/

ADD https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.84/bin/apache-tomcat-9.0.84.tar.gz  .

RUN tar -xvf apache-tomcat-9.0.84.tar.gz

RUN sed -i 's/"127\\.\\d+\\.\\d+\\.\\d+|::1|0:0:0:0:0:0:0:1"/".*"/g' /opt/apache-tomcat-9.0.84/webapps/manager/META-INF/context.xml

WORKDIR /opt/apache-tomcat-9.0.84/conf/

RUN rm -rf tomcat-users.xml

RUN  echo '<?xml version="1.0" encoding="utf-8"?> \
                <tomcat-users> \
                <role rolename="manager-gui"/> \
                <user username="tomcat" password="Tomcat" roles="manager-gui, manager-script, manager-status"/> \
                </tomcat-users>' > tomcat-users.xml

CMD ["/opt/apache-tomcat-9.0.84/bin/catalina.sh", "run"]

EXPOSE 8080
```

Commands To Execute Dockerfile:

docker build –t <imagename> <Dockerfile> .	To build image
docker run --itd --name <cont1> -p cont port:Hostport <imagename> eg: docker run –itd –name cont1 –p 8081:8080 imagename/id	To create and run a container in detach mode
docker exec –it <contname> bash	To enter into the container

Docker Advantages

- **Consistency**. Docker ensures reliability that your app runs the same across multiple environments. Developers working on different machines and operating systems can work together on the same application without environment issues.
- **Automation**. The platform allows you to automate tedious, repetitive tasks and schedule jobs without manual intervention.
- **Faster deployments**. Since containers virtualize the OS, there is no boot time when starting up containers instances. Therefore, you can do deployments in a matter of seconds. Additionally, you can share existing containers to create new applications.
- **Support of CI/CD**. Docker works well with CI/CD practices as it speeds up deployments, simplifies updates, and allows teammates to work efficiently together.
- **Rollbacks and image version control**. A container is based on a Docker image which can have multiple layers, each representing changes and updates on the base. Not only does this feature speed up the build process, but it also provides version control over the container. This allows developers to roll back to a previous version if the need arises.
- **Modularity**. Containers are independent and isolated virtual environments. In a multi-container application, each container has a specific function. By segregating the app, developers can easily work on a particular part without taking down the entire app.
- **Resource and cost-efficiency**. As containers do not include guest operating systems, they are much lighter and smaller than VMs. They take up less memory and reuse components thanks to data volumes and images. Also, containers don't require large physical servers as they can run entirely on the cloud.

Reference

Interview questions

- https://www.simplilearn.com/tutorials/docker-tutorial/docker-interview-questions
- https://www.interviewbit.com/docker-interview-questions/
- https://www.edureka.co/blog/interview-questions/docker-interview-questions/

Quiz:

https://docs.google.com/forms/d/e/1FAIpQLSfWKryaY58DUbf1vkOGywCf9YOsysncXp8_ZVnE-p8TnRHIIQ/viewform?usp=sf_link