What can you containerize?

We can containerize almost all sorts of the application even simple ones, like browsers or utilities like curl, applications like spotify etc.

CMD VS ENTRYPOINT

Containers are not meant to host an operating system. Containers are meant to run a specific task or process, such as to host an instance of a web server or application server or a database or simply to carry out some kind of computation or analysis. Once the task is complete the container exits, a container only lives as long as the process inside it is alive.

There are different ways of specifying the command. Either the command simply as is in a shell form:

```
CMD command param 1
CMD ["command", "param1"]
```

or in a JSON array format like this:

```
CMD sleep 5
CMD ["sleep", "5"]
```

When you specify in a JSON array format, the first element in the array should be the executable.

The entrypoint instruction is like the command instruction. As in you can specify the program that will be run when the container starts. And whatever you specify on the command line will get appended to the entry point.

Difference:

In case of the CMD instruction, the command line parameters passed will get replaced entirely.
Whereas in case of entry point, the command line parameters will get appended.

Q. How do you configure a default value

for the command if one was not specified in the command line?

That's where you would both use both entry point as well as the command instruction. The command instruction will be appended to the entry point instruction.

DOCKER NETWORKING

When you install docker, it creates three network automatically. Bridge, none and host.

Bridge: is the default network a container gets attached to. If you would like to associate the container with any other network, specify the network information using the network command line parameter.

The bridge network is a private internal network created by Docker on the host. All containers attached to this network by default, and they get an internal IP address, usually in the range 172.17 series. The containers can access each other using this internal IP if required. To access any of these containers from the outside world, map the ports of these containers to port on the Docker host.

To access any of these containers from the outside world, map the ports of these containers to port on the Docker Host. Another way to access the containers externally is to associate the container to the host network. This takes out any network isolation b/w the Dockers host and the Dockes container.

Meaning if you were to run a web server on port 5000; in a web app container, it is automatically accessible on the same port externally without requiring any port mapping as the web container uses the host's network.
This will also mean that unlike before, you will now not be able to run multiple web containers on the same host on the same port, as the ports are now common to all containers in the host network.

With the none network, the containers are not attached to any network and doesn't have any access to the external network, or other containers. They run in an isolated network.

User Defined Networks:
By default, Docker only creates one internal

bridge network, we could create our own internal network using the command

```
docker network create \
  --driver bridge \
  --subnet 182.18.0.0/16
  custom-isolated-network.
```

Embedded DNS

All containers in a Docker host can resolve each other with the name of the container. Docker has a built in DNS server that helps the containers to resolve each other using the container name.
The built in DNS server always runs at address 127.0.0.11.
Docker uses network namespaces that creates a seperate namespace for each container. It then uses virtual Ethernet pairs to connect containers together.

DOCKER STORAGE

File System:
When you install Docker on a system, it creates this folder structure at /var/lib/docker.

Layered Architecture

Read only

Image Layers
- Layer 5. Update Entrypoint with 'flask' command
- Layer 4. Source code
- Layer 3. Changes in pip packages
- Layer 2. Changes in apt packages
- Layer 1. Base Ubuntu Layer

So all of these are the Docker image layers. Once the build is complete, you cannot modify the contents of these layers and so they are read only and you can modify them by initiating a new build.

Read Write

Container layer
- Layer 6. Container Layer

When you run a container based off of this image using the Docker run command, Docker creates a container based off of these layers and creates a new writable layer on top of the ~~base~~ image layers. The writable layer is used to store data created by the container, such as log files.

Q. Can I not modify the file inside the read only containers?

A. Yes, we can still modify this file. But

before I save the modified file, Docker automatically creates a copy of the file in the read write layer. This is called copy on write mechanism.

Volumes

docker volume create data-volume

/var/lib/docker

---volumes

-----data-volume

If we wish to persist data, like if we were working with a database, we could add a persistent volume to the containers. To do this, we create a volume.

Docker will automatically create a volume and mount it to the containes if not created manually.

New way:

docker run \

-- mount type=bind, source=/data/mysql, target=/var/lib/mysqul mysql.

Storage Drivers

Docker uses storage drivers to enable layered architecture. Some of the common storage drivers are

- AUFS
- ZFS
- BTRFS
- Device Mapper
- Overlay
- Overlay 2

The selection of the storage driver depends on the underlying OS being used. e.g. with Ubuntu, the default storage driver is AUFS.
Device Mapper may be a better option.

DOCKER COMPOSE

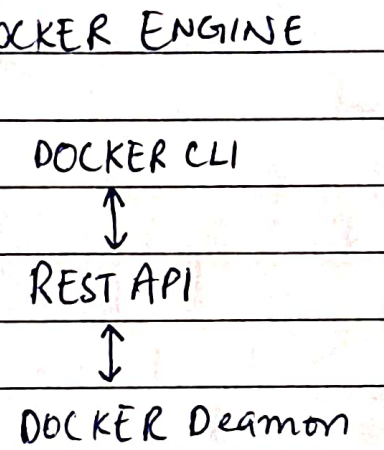
We can create a configuration file in yml format called "Docker-Compose.yml" and put together the different services and the options specific to this to running them in this file.
We could simply run a command to bring up the entire application stack.
"Docker compose up".

Link is a command line option, which can be used to link two containers together.

DOCKER REGISTRY

Its a central repository of all Docker images. The registary is where all the images are stored.

DOCKER ENGINE



Docker Deamon is a background process that manages Docker objects such as the images, containers, volumes and networks.

The Docker Rest API server is the API interface that programs can use to talk to the deamon and provide instructions.

The DOCKER CLI is the command line interface to perform actions such as running a container, stopping containers,

destroying images, etc.

The Docker CLI could be on another system like a laptop and can still work with a remote Docker engine.

```
docker -H=remote-docker-engine:2375
```

```
docker -H=10.123.2.1:2375 run nginx
```