Docker : Docker is a software platform for building applications based on [*containers*](https://www.infoworld.com/article/3077875/containers-101-docker-fundamentals.html) small and lightweight execution environments that make shared use of the operating system kernel but otherwise run in isolation from one another. While containers have been used in Linux and Unix systems for some time, Docker, an open source project launched in 2013, helped popularize the technology by making it easier than ever for developers to package their software to “build once and run anywhere.

docker run alpine echo hello world

* run a command in a new container
* alpine is one of the smallest Linux systems
* echo hello world is the command which is executed

For interactive processes (like a shell), We must use docker run with -i -t together in order to allocate a tty for the container process. -i -t is often written -it.

To expose a container's internal port, We can start the container with the -P or -p flag. The exposed port is accessible on the host and the ports are available to any client that can reach the host.

docker ps -a – :

he container is stopped but it still exists on disk. We can take a look:

docker rm $(docker ps -aq):

To delete all running and stopped container execute

docker ps -a:

Check if the containers get deleted

docker pull loodse/demo-www:

docker pull command downloads image from docker hub (which serves as a default registry for docker installation).

docker run -d loodse/demo-www:

The -d flag instructs docker to create container and run it in background.

We can see list of running containers using docker ps. We can see an entry with loodse/demo-www string in IMAGE column.

docker ps -a:

The docker ps command only shows running containers by default. To see all containers, use the -a (or --all) flag:

docker run -t -p 8080:80 loodse/demo-www:

With docker ps you see now two running containers.

A Dockerfile copies the working directory into the Docker Image. As a result, this would include potentially sensitive information such as a passwords file which we'd want to manage outside the image. View the Dockerfile with

cd ~/scrapbook/tutorial cat Dockerfile

Build the image with docker build -t password . .

Look at the output using docker run password ls /app

This will include the passwords file.

the Docker Ignore file it shouldn't include the passwords file.

docker build -t nopassword . .

docker run nopassword ls /app:

The .dockerignore file can ensure that sensitive details are not included in a Docker Image. However they can also be used to improve the build time of images.

In the environment, a hypothetical 100M temporary file has been created. This file is never used by the Dockerfile. When you execute a build command, Docker sends the entire path contents to the Engine for it to calculate which files to include. As a result sending the 100M file is unrequired and creates a slower build.

echo somelargefile.img >> .dockerignore :

When we rebuild the image, it will be much faster as it doesn't have to copy the large file.

What is inspect in docker?

docker inspect is **a command that returns detailed, low-level information on Docker objects**. Those objects can be docker images, containers, networks, volumes, plugins, etc

What is naming in docker?

So far we have referenced containers only with their ID or with a prefix, but containers can also be referenced by their names also.

Docker Logs :

When we started a container then docker will track the standard out and standard error output from process them available via the client.

Using the Docker client, we can access the standard out and standard error outputs using docker logs redis-server

What is syslog Docker?

The syslog logging driver **routes logs to a syslog server**. The syslog protocol uses a raw string as the log message and supports a limited set of metadata. The syslog message must be formatted in a specific way to be valid.

What is Docker networking?

Docker networking is **primarily used to establish communication between Docker containers and the outside world via the host machine** where the Docker daemon is running. Docker supports different types of networks, each fit for certain use cases.

if we run a container nginx image. docker run -d -P --name myweb nginx then

we can find ports exposed by container in docker ps. Check PORTS column.

Port can be manually exposed using -p flag. docker run -d -p 8080:80 --name myweb1 nginx

Above command will start a container from image nginx with internal port 80 exposed to external port 8080.

To check network connectivity to container IP using this command ping $(docker inspect --format '{{ .NetworkSettings.IPAddress }}' myweb1)

**Bridge networking**

It is a default networking mode if you don't specify networking mode explicitly using --net option.

It creates MASQUERADE and DNAT iptables rules for outbound and inbound traffic respectively.

This mode creates network isolation for container. A container can't send and receive network traffic. It can only see lo interface.

We can used this Null driver mode like docker run -d --net none --name nullnetdemo nginx.

Even though it is a container running web server, We can't connect to it.

Check network details using docker inspect nullnetdemo -f '{{json .NetworkSettings.Networks}}'

It shows none in networks.

Start a Redis container below, and create a data volume using the -v parameter. This specifies that any data saved inside the container to the /data directory should be persisted on the host in the directory /docker/redis-data.

docker run -v /docker/redis-data:/data \ --name r1 -d redis \ redis-server --appendonly yes

# Shared Volumes

Data Volumes mapped to the host are great for persisting data. However, to gain access to them from another container we need to know the exact path which can make it error-prone.

An alternate approach is to use -volumes-from. The parameter maps the mapped volumes from the source container to the container being launched.