Lab: Introduction to Docker - Part 1

What is Docker?

Docker is the industry standard in container technology. Docker containers resemble virtual machines in providing portability of applications regardless of the host machine operating system. However, a container is not a virtual machine, it is much more lightweight. A container is a standardized unit of software that includes everything that is needed to run an application on any host OS: code, any dependencies, and any system tools and libraries and settings required. Any host OS that runs Docker engine will be able to run any Docker containers. Containers do not contain an OS (as in the case of virtual machines), instead, they all share the underlying machine's OS system kernel — this makes them much more lightweight than virtual machines, while still retaining the property of portability.

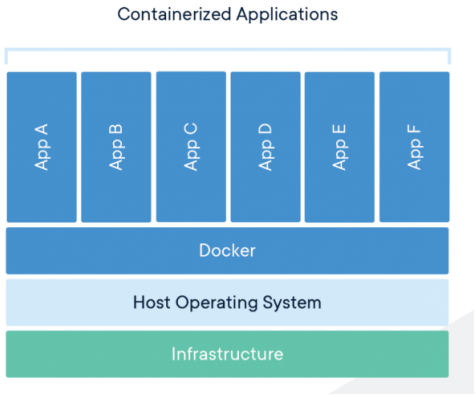

Figure 1: `https://www.docker.com/resources/what-container`

Install and set-up

Download and install Docker for Windows:

- Go to `https://docs.docker.com/docker-for-windows/install/`

- Click "Download from Docker Hub"
- On the right, it says "Get Docker Desktop (Windows)". Under that it should say it requires Windows 10 Pro or Enterprise.
- Click the link to sign up and create a Docker ID (if you haven't already).
- Once you've signed up and signed in, you can download Docker.
- Run the executable to install.
- When installing, in "Configuration", **do not** click "Use Windows containers instead of Linux containers".

Now that Docker is installed:

- Click on the "Docker for Windows" icon on your desktop. A box will pop up to say that Hyper-V features are not enabled. Click OK. Computer will restart. If the box doesn't pop up, manually restart computer.
- After the computer restarts, Docker Desktop will automatically start. Log in with your Docker ID and password. Now Docker is running.
- Type in shell "docker version" to see the version installed. Make sure you can see Client and Server.
- There is a Docker whale icon in your taskbar. Right click, go to "Settings" to see the current settings, including memory and CPU allocated. Go to "Shared Drives". Check "C" since you'll most likely have code stored in C: drive that runs inside containers. Go to "Advanced". You shouldn't change anything here. But if in the future you find you'll need more memory or CPU, you can always change these if your hardware allows.

Some basic concepts and try it out

There are *images* and *containers*. An image is an application including all the executable binaries and source code and libraries. A container is an *instance* of the image running as a process. You can have many containers from the same image.

Docker Hub stores images that you can download and run containers from them.

Let's try it out.

1. Start Cmder or Powershell.
2. (The symbol '>' indicates whatever command prompt you may be using, do not type it in.) Run a docker container by typing:
   `> docker container run --publish 80:80 --detach nginx`

   The image `nginx` is downloaded from Docker Hub (first Docker will look for this image in the local cache, if it's not found, then go to Hub to download). Then a container is run from it.

   The `--publish 80:80` means, open port 80 (the one on the left) on local host machine, all traffic to this port will be directed to any code in the container that accepts traffic from port 80 (the one on the right).

   The `--detach` means, run the container in the background.

   Interact with this running container by opening a browser, type in `localhost:80`, and you should see the welcome page for nginx.
3. You can list all the currently running containers:
   `> docker container ls`

   Notice there is a container ID for each container. For example, say the container ID is 0080a3825c76.
4. You can stop a container with ID 0080a3825c76:
   `> docker container stop 00`

   You only need to enter the first few digits of the ID that distinguishes it from other container IDs.
5. Notice containers also are assigned names. You can name your containers. Let's run a container from `nginx` image and name it `mynginx`
   `> docker container run --publish 80:80 --detach --name mynginx nginx`

Task

- Run an `nginx` container and a `ubuntu` container, in the background.

- Give a name to each container.
- List all the running containers.
- Stop both containers.