Docker CheatSheet:

Docker Basics

Terms

Containers
Dockerfiles
Docker daemon (engine)
Docker registries
Hypervisors
Images
Kernel
Process
Virtual machines

Summary

•Docker is a platform for consistently building, running, and shipping applications.
•A virtual machine is an abstraction of hardware resources. Using hypervisors we can create and manage virtual machines. The most popular hypervisors are VirtualBox, VMware and Hyper-v (Windows-only).
•A container is an isolated environment for running an application. It's essentially an operating-system process with its own file system.
•Virtual machines are very resource intensive and slow to start. Containers are very lightweight and start quickly because they share the kernel of the host (which is already started).
•A kernel is the core of an operating system. It's the part that manages applications and hardware resources. Different operating system kernels have different APIs. That's why we cannot run a Windows application on Linux because under the hood, that application needs to talk to a Windows kernel.
•Windows 10 now includes a Linux kernel in addition to the Windows kernel. So we can run Linux applications natively on Windows.
•Docker uses client/server architecture. It has a client component that talks to the server using a RESTful API. The server is also called the Docker engine (or daemon) runs in the background and is responsible for doing the actual work.
•Using Docker, we can bundle an application into an image. Once we have an image, we can run it on any machine that runs Docker.
•An image is a bundle of everything needed to run an application. That includes a cutdown OS, a runtime environment (eg Node, Python, etc), application files, third-party libraries, environment variables, etc.
•To bundle an application into an image, we need to create a Dockerfile. A Dockerfile contains all the instructions needed to package up an application into an image.
•We can share our images by publishing them on Docker registries. The most popular Docker registry is Docker Hub.

Images

```
FROM        # to specify the base image
WORKDIR     # to set the working directory
COPY        # to copy files/directories
ADD         # to copy files/directories
RUN         # to run commands
ENV         # to set environment variables
EXPOSE      # to document the port the container is listening on
USER        # to set the user running the app
CMD         # to set the default command/program
ENTRYPOINT  # to set the default command/program
```

Dockerfile instructions

```
docker build -t <name> .
docker images
docker image ls
docker run -it <image> sh
```

Image commands

```
docker stop <containerID>
docker start <containerID>
```

Starting and stopping containers

```
docker container rm <containerID>
docker rm <containerID>
docker rm -f <containerID>        # to force the removal
docker container prune            # to remove stopped containers
```

Removing containers

```
docker volume ls
docker volume create app-data
docker volume inspect app-data
docker run -v app-data:/app/data <image>
```

Volumes

```
docker cp <containerID>:/app/log.txt .
docker cp secret.txt <containerID>:/app
```

Copying files between the host and containers

```
docker run -v $(pwd):/app <image>
```

Containers

```
docker run <image>
docker run -d <image>             # run in the background
docker run —name <name> <image>   # to give a custom name
docker run —p 3000:3000 <image>   # to publish a port HOST:CONTAINER
```

Running containers

```
docker logs <containerID>
docker logs -f <containerID>      # to follow the log
docker logs —t <containerID>      # to add timestamps
docker logs —n 10 <containerID>   # to view the last 10 lines
```

Viewing the logs

```
docker exec <containerID> <cmd>
docker exec -it <containerID> sh  # to start a shell
```

Executing commands in running containers

```
docker ps            # to list running containers
docker ps -a         # to list all containers
```

Listing containers

```
docker stop <containerID>
docker start <containerID>
```

Starting and stopping containers

```
docker container rm <containerID>
docker rm <containerID>
docker rm -f <containerID>      # to force the removal
docker container prune          # to remove stopped containers
```

Removing containers

```
docker volume ls
docker volume create app-data
docker volume inspect app-data
docker run -v app-data:/app/data <image>
```

Volumes

```
docker cp <containerID>:/app/log.txt .
docker cp secret.txt <containerID>:/app
```