# Docker - Quick Command Reference

**Manage images**

**docker image pull <image name>**
*download an image from DockerHub*

**docker image ls**
*list all local images*

**docker image build -t <image name> .**
*build an image with a tag (note the dot!)*

**docker image push <image name>**
*publish an image to dockerhub*

**docker image tag <image id> <tag name>**
*tag an image - either alias an exisiting image or apply a :tag to one*

**Manage Containers**

**docker container run -p <public port>:<container port> <image name>**
*run a container from an image, publishing the specified ports*

**docker container ls -a**
*list all containers, even the stopped ones*

**docker container stop <container id>**
*stop a running container*

**docker container start <container id>**
*restart a stopped container*

**docker container rm <container id>**
*remove a stopped container*

# Docker - Quick Command Reference

**Manage Containers (ctd)**

**docker container prune**
*remove all stopped containers*

**docker container run -it <image name>**
*run a container with interactive terminal*

**docker container run -d <image name>**
*run a container detached (or in a daemon like way)*

**docker container exec -it <container id> <command>**
*run a command in a container*

**docker container exec -it <container id> bash**
*special form of the above, runs a bash shell, connected to your local terminal (your distro needs to have bash, alpine will require /bin/sh)*

**docker container logs -f <container id>**
*Follow the log (STDIN/System.out) of the container*

**docker container commit -a "author" <container id> <image name>**
*Take a snapshot image of a container*

**Manage your (local) Virtual Machine**

**docker-machine ip**
*Find the IP address of your VirtualMachine, required for Docker Toolbox users only*

# Docker - Quick Command Reference

**Manage Networks**

**docker network ls**
*list all networks*

**docker network create <network name>**
*create a network using the bridge driver*

**Manage Volumes**

**docker volume ls**
*list all volumes*

**docker volume prune**
*delete all volumes that are not currently mounted to a container*

**docker volume inspect <volume name>**
*inspect a volume (can find out the mount point, the location of the volume on the host system)*

**docker volume rm <volume name>**
*remove a volume*

**Docker Compose**

**docker-compose up**
*process the default docker-compose.yaml file, starting any containers as required. If containers are already running they are ignored, meaning this command also serves as a "redeploy".*

**docker-compose up -d**
*run containers in the detached state. Note the order of the command line arguments!*

**docker-compose logs -f <service name>**
*follow the log for the specified service. Omit the -f to tail the log.*

**docker-compose down**
*stop all the containers (services) listed in the default compose file.*

# Docker - Quick Command Reference

**Manage a Swarm**

**docker swarm init (--advertise-addr <ip address>)**
*Switch the machine into Swarm mode. We didn't cover how to stop swarm mode:* ***docker swarm leave --force***

**docker service create <args>**
*Start a service in the swarm. The args are largely the same as those you will have used in docker container run.*

**docker network create --driver overlay <name>**
*Create a network suitable for using in a swarm.*

**docker service ls**
*List all services*

**docker node ls**
*List all nodes in the swarm*

**docker service logs -f <service name>**
*Follow the log for the service. This feature is a new feature in Docker and may not be available on your version (especially if using Linux Repository Packages).*

**docker service ps <service name>**
*List full details of the service - in particular the node on which it is running and any previous failed containers from the service.*

**docker swarm join-token <worker|manager>**
*Get a join token to enable a new node to connect to the swarm, either as a worker or manager.*

**Manage Stacks**

**docker stack ls**
*list all stacks on this swarm.*

**docker stack deploy -c <compose file> <stack name>**
*deploy (or re-deploy) a stack based on a standard compose file.*

**docker stack rm <stack name>**
*delete a stack and its corresponding services/networks/etc.*