**Management**

# Stop all containers.

docker stop $(docker ps -a -q -f "status=running")

**Cleanup**

# Remove all containers.

docker rm $(docker ps -a -q -f)

# Remove stopped containers.

docker rm $(docker ps -a -q -f "status=exited")

# Remove all images.

docker rm $(docker images -q -f)

# Remove orphaned images.

docker rmi $(docker images -q -f "dangling=true")

# Remove orphaned volumes.

docker volume rm $(docker volume ls -q -f "dangling=true")

## Set up

Pull a base image.

docker pull ubuntu

It's annoy to restore Container ID, you may forget to restore. You can set below alias. With this, you can get the ID of the last-run Container ([15 Docker tips in 5 minutes](http://sssslide.com/speakerdeck.com/bmorearty/15-docker-tips-in-5-minutes))

alias dl='docker ps -l -q'

## Container

To create a Container.

docker run -d ubuntu /bin/sh -c "while true; do echo hello world; sleep 1; done"

To stop a Container.

docker stop `dl`

To start a Container.

docker start `dl`

To restart a Container.

docker restart `dl`

To Connect to a running Container.

docker attach `dl`

To copy file in a Container to the host.

docker cp `dl`:/etc/passwd .

To mount the directory in host to a Container.

docker run -v /home/vagrant/test:/root/test ubuntu echo yo

To delete a Container.

dockr rm `dl`

## Info of Container

To show running Containers. With -a option, it shows running and stopped Containers.

docker ps

To show Container information like IP adress.

docker inspect `dl`

To show log of a Container.

docker logs `dl`

To show running process in a Container.

docker top `dl`

## Image

To create a image from a Container. For tag name, <username>/<imagename> is [recommended](http://docs.docker.io/en/latest/use/workingwithrepository/#committing-a-container-to-a-named-image).

docker run -d ubuntu /bin/sh -c "apt-get install -y hello"

docker commit -m "My first container" `dl` tcnksm/hello

To create a image with Dockerfile.

echo -e "FROM base\nRUN apt-get install hello\nCMD hello" > Dockerfile

docker build tcnksm/hello .

To login to a image.

docker run -rm -t -i tcnksm/hello /bin/bash

To push a imges to remote repository. You need to sign up to [Docker index](https://index.docker.io/) in advance. [Exmple uploaded image](https://index.docker.io/u/tcnksm/hello).

docker login

docker push tcnksm/hello

To delete a image

docker rmi tcnkms/hello

## Info of Image

To show all images

docker images

To show image information like IP adress.

docker inspect tcnksm/hello

To show command history of a image.

docker history tcnksm/hello

# Docker Command Essentials

The idea of docker essentials is to summarize all the most use full docker commands.

## Docker

Docker status related command

### docker info

To check that the docker is working correctly. You need at least to have server detected.

$ docker info

## Container

Container related command

### docker run

To create and run a container. You need to give at least a docker image as a parameter.

$ docker run hello-world

**Interactive shell options**

* -i, --interactive : Keep STDIN open
* -t, --tty : Allocate pseudo-tty (terminal)

$ docker run -i -t ubuntu /bin/bash

...

root@9b5e9657f47b:/#

Run ubuntu container, and launch the bin/bash command.

The container will run, as long as the application keep run.

**Container naming**

* --name : define name.

$ docker run --name container\_name -i -t ubuntu /bin/bash

By default docker will give a container a name.

**Detached mode**

* -d, --detach : Run container in background and print container ID

**Port Publish**

* -p, --publish : Publish a container's port(s) to the host

Two ways of publish a port

* Let docker to define the host port
* $ docker run -d -p 80 image\_name
* Define a specific port in the host. Here map the port 80 of the container to the port 80 of the host.
* $ docker run -d -p 80:80 image\_name

### docker ps

List containers.

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

8841e75b49a3 ubuntu "/bin/bash" 2 minutes ago Up 7 seconds container\_name

**List all container**

* -a, --all : Show all containers (default shows just running)

$ docker ps -a

### docker start

Start docker by using its name or id.

Using name

$ docker start container\_name

Using id

$ docker start 8841e75b49a3

### docker stop

Stop docker by using its name or id.

Using name

$ docker stop container\_name

Using id

$ docker stop 8841e75b49a3

### docker attach

Attach to the container session.

Using name

$ docker attach container\_name

Using id

$ docker attach 8841e75b49a3

In this case is the /bin/bash command launched.

### docker logs

Fetch the log of a container

$ docker logs container\_name

Use Ctrl-C to exit.

### docker top

Inspect the processes inside the container

$ docker top container\_name

### docker stats

Show statistics for one or more running containers.

$ docker top container\_a container\_b

### docker exec

Execute a command inside a container.

$ docker exec -t -i container\_name /bin/bash

See the **docker run** for the -t -i options

### docker inspect

To gather more information.

$ docker inspect container\_name

### docker rm

To delete a container

$ docker rm container\_name

## Images

Images related command.

### docker images

Lists all images in a docker hosts.

$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

hello-world latest 693bce725149 11 days ago 967 B

ubuntu latest 2fa927b5cdd3 3 weeks ago 122 MB

### docker pull

Pull an image from docker registry.

$ docker pull ubuntu

We can specify a tag for a docker image

$ docker pull ubuntu:12.04

Where 12.04 is a tag.

### docker search

Search an image from docker hub

$ docker search ubuntu

### docker commit

Commit an state of a container to an image, this method is not recommended, instead use Dockerfile

$ docker commit 8841e75b49a3 container\_new\_name

**commit information**

* -m : commit message
* -a : commit author

We can also add tag to a commit

$ docker commit -m "message" -a "bowie brotosumpeno" 8841e75b49a3 container\_new\_name:new

### docker build

Build a docker image from a Dockerfile

**Define repository/name**

* -t : define a repository/name of an image

$ docker build -t="keraton/container\_name:tag\_v1 .

The . to reference the repository of a Dockerfile.

### docker history

Show history of an image.

+

$ docker history 2fa927b5cdd3

### docker port

To get an information about container to host mapping.

$ docker port 2fa927b5cdd3 80

0.0.0.0:49154

In this example the port 80 of contianer 2fa927b5cdd3 is mapped to port 49154 of host

# Dockerfile Essentials

By default docker will find a file with a name Dockerfile. This file is used to defines series of instructions to build an image.

Example :

# Usefull comment

FROM ubuntu: 14.04

MAINTAINER Bowie Brotosumpeno "keratonjava@gmail.com"

RUN apt-get update && install -y nginx

RUN echo 'Hello world' \

> /user/share/nginx/html/index.html

EXPOSE 80

### Comment (#)

All the lines starts with a # is considered as a comments

# Usefull comment

### FROM

The first instruction of a Dockerfile to specifies an existing image to be used as a base image.

FROM ubuntu: 14.04

### MAINTAINER

Define the author of the image

MAINTAINER Bowie Brotosumpeno "keratonjava@gmail.com"

### RUN

Run instruction inside the container

RUN apt-get update && install -y nginx

We can run the instruction in exec format

RUN ["apt-get", "update", "&&", "install", "-y", "nginx"]

### ENV

Set an environment variable in the image

ENV REFRESHED\_AT 2016-06-19

### EXPOSE

Specify a port that the container used.

EXPOSE 80

Can be use with mapping too.

+

EXPOSE 80:80

### CMD

Specify the command to run when a container launched.

CMD ["/bin/bash", "-l"]

You can override CMD in the docker run

### ENTRYPOINT

An Entrypoint of an instruction, like CMD it run when a container launched. But it should not be override when docker run have some arguments. Instead the arguments in the docker run used as the arguments of the command that specified by ENTRYPOINT.

ENTRYPOINT ["/bin/bash"]

### WORKDIR

Set the working directory for the container RUN, ENTRYPOINT and CMD

WORKDIR /opt/webapp/db

RUN bundle install

WORKDIR /opt/webapp

ENTRYPOINT ["/bin/bash"]

### USER

Specify which user that image should run.

USER nginx

By default the user is root

### VOLUME

Add Volumes to any container created from the image. Advantages of volumes are :

* Volumes can be shared and reused between containers.
* A container doesn't have to be running to share its volumes.
* Changes to a volume are made directly.
* Volumes persist (until no containers use them).

To mount a volume

VOLUME ["/opt/project"]

### ADD

Add a file or directory from the host or from URL to the container

ADD software.lic /opt/application/software.lic

ADD http://wordpress.org/latest.zip /root/wordpress.zip

ADD can untar a compressed tar file

ADD latest.tar /var/www/wordpress/

### COPY

Same as ADD but focusing to copy without untar.

COPY latest.tar /var/www/wordpress/

### LABEL

Add a metadata label to docker image in the form of key values.

LABEL version="1.0"

LABEL beta="true"

These labels shown with the docker inspect command.

### STOPSIGNAL

Send a signal to the container.

### ARG

Defines variables that can be passed at build time

ARG version

ARG beta="true"

This is how we use it with docker build by using option --build-arg

docker build --build-arg version="123"

### ONBUILD

Add triggers to images.

### List All Docker Images

docker images -a  
List All Running Docker Containers

docker ps  
List All Docker Containers

docker ps -a  
Start a Docker Container

docker start <container name>  
Stop a Docker Container

docker stop <container name>

### View the Logs of a Running Docker Container

docker logs <container name>

### Delete All Docker Containers

Use -f option to nuke the running containers too.

docker rm $(docker ps -a -q)

### Remove a Docker Image

docker rmi <image name>

### Delete All Docker Images

docker rmi $(docker images -q)

### SSH Into a Running Docker Container

Okay not technically SSH, but this will give you a bash shell in the container.

sudo docker exec -it <container name> bash

### Use Docker Compose to Build Containers

Run from directory of your docker-compose.yml file.

docker-compose build

### Use Docker Compose to Start a Group of Containers

Use this command from directory of your docker-compose.yml file.

docker-compose up -d

This will tell Docker to fetch the latest version of the container from the repo, and not use the local cache.

docker-compose up -d --force-recreate

This can be problematic if you’re doing CI builds with Jenkins and pushing Docker images to another host, or using for CI testing. I was deploying a Spring Boot Web Application from Jekins, and found the Docker container was not getting refreshed with the latest Spring Boot artifact.

#stop docker containers, and rebuild

docker-compose stop -t 1

docker-compose rm -f

docker-compose pull

docker-compose build

docker-compose up -d

### Follow the Logs of Running Docker Containers With Docker Compose

docker-compose logs -f

### Save a Running Docker Container as an Image

docker commit <image name> <name for image>

### Follow the Logs of One Container Running Under Docker Compose

docker-compose logs pump <name>

## Dockerfile Hints for Spring Boot Developers

### Add Oracle Java to an Image

For CentOS/ RHEL

ENV JAVA\_VERSION 8u31

ENV BUILD\_VERSION b13

# Upgrading system

RUN yum -y upgrade

RUN yum -y install wget

# Downloading & Config Java 8

RUN wget --no-cookies --no-check-certificate --header "Cookie: oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/$JAVA\_VERSION-$BUILD\_VERSION/jdk-$JAVA\_VERSION-linux-x64.rpm" -O /tmp/jdk-8-linux-x64.rpm

RUN yum -y install /tmp/jdk-8-linux-x64.rpm

RUN alternatives --install /usr/bin/java jar /usr/java/latest/bin/java 200000

RUN alternatives --install /usr/bin/javaws javaws /usr/java/latest/bin/javaws 200000

RUN alternatives --install /usr/bin/javac javac /usr/java/latest/bin/javac 200000

### Add/Run a Spring Boot Executable Jar to a Docker Image

VOLUME /tmp

ADD /maven/myapp-0.0.1-SNAPSHOT.jar myapp.jar

RUN sh -c 'touch /myapp.jar'

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/myapp.jar"