Docker Cheatsheet

This cheatsheet provides a collection of commonly used docker commands.

Getting Started

The getting started guide on Docker has detailed instructions for setting up Docker.

After setup is complete, run the following commands to verify the success of installation:

PLEASE NOTE POST INSTALLATION STEPS BELOW IF YOU HAVE TO PREPEND SUDO TO EVERY COMMAND

- **docker version** - provides full description of docker version
- **docker info** - display system wide information
- **docker -v** - provides a short description of docker version
- **docker run hello-world** - pull hello-world container from registry and run it

Have a look at the free training offered by Docker.

Have a look at the repository of images offered by Docker.

Optional Post Installation Steps

To create the docker group and add your user:

- Create the docker group

  ```
  sudo groupadd docker
  ```

- Add your user to the docker group

  ```
  sudo usermod -aG docker $USER
  ```

- Log out and log back in so that your group membership is re-evaluated.
- Verify that you can run docker commands without sudo.

Docker Commands

Get docker info

General

Command	Description
docker version	provides full description of docker version
docker -v	provides a short description of docker version
docker info	display system wide information
docker info –format '{{.DriverStatus}}'	display 'DriverStatus' fragment from docker information
docker info –format '{{json .DriverStatus}}'	display 'DriverStatus' fragment from docker information in JSON format

Manage Images

Command	Description
docker image ls	shows all local images
docker image ls –filter 'reference=ubuntu:16.04'	show images filtered by name and tag
docker image pull [image-name]	pull specified image from registry
docker image rm [image-name]	remove image for specified *image-name*
docker image rm [image-id]	remove image for specified *image-id*
docker image prune	remove unused images

Search Images

Command	Description
docker search [image-name] –filter "is-official=true"	find only official images having *[image-name]*
docker search [image-name] – filter "stars=1000"	find only images having specified *[image-name]* and 1000 or more stars

Manage Containers

Display Container Information

Command	Description
docker container ls	show all running containers
docker container ls -a	show all containers regardless of state
docker container ls –filter "status=exited" –filter "ancestor=ubuntu"	show all container instances of the ubuntu image that have exited

Command	Description
docker container inspect [container-name]	display detailed information about specified container
docker container inspect –format ‘{{.NetworkSettings.IPAddress}}’ [container-name]	display detailed information about specified container using specified format
docker container inspect –format ‘{{json .NetworkSettings}}’ [container-name]	display detailed information about specified container using specified format

Run Container

Command	Description
docker container run [image-name]	run container based on specified image
docker container run –rm [image-name]	run container based on specified imaged and immediately remove it once it stops
docker container run –name fuzzy-box [image-name]	assign name and run container based on specified image

Remove Container

Command	Description
docker container rm [container-name]	remove specified container

docker container rm ($docker container ls --filter "status = exited" --filter "ancestor = ubuntu" -q) | remove all containers whose id's are returned from *'(...)'* list

Manage Volumes

Display Volume Information

Command	Description
docker volume ls	show all volumes
docker volume ls –filter “dangling=true”	display all volumes not referenced by any containers
docker volume inspect [volume-name]	display detailed information on *[volume-name]*

Remove Volumes

Command	Description
docker volume rm [volume-name]	remove specified volume

docker volume rm ($docker volume ls --filter "dangling = true" -q) | remove all volumes having an id equal to any of the list

---

Running containers

Run hello-world container

```
docker run hello-world
```

Run an Alpine Linux container (a lightweight linux distribution)

1. Find image and display brief summary

   ```
   docker search alpine --filter=stars=1000 --no-trunc
   ```

2. Fetch alpine image from Docker registry and save it

   ```
   docker pull alpine
   ```

3. Display list of local images

   ```
   docker image ls
   ```

4. List container contents using *listing* format

   ```
   docker run alpine ls -l
   ```

5. Print message from container

   ```
   docker run alpine echo "Hello from Alpine!"
   ```

6. Running the run command with the -it flags attaches container to an interactive tty

   ```
   docker run -it alpine bin/sh
   ```

Run MongoDB

Run MongoDB Using Named Volume To run a new MongoDB container, execute the following command from the CLI:

```
docker run --rm --name mongo-dev -v mongo-dev-db:/data/db -d mongo
```

CLI Command	Description
–rm	remove container when stopped
–name mongo-dev	give container a custom name
-v mongo-dev-db/data/db	map the container volume 'data/db' to a custom name 'mongo-dev-db'
-d mongo	run mongo container as a daemon in the background

Run MongoDB Using Bind Mount

```
cd
mkdir -p mongodb/data/db
docker run --rm --name mongo-dev -v ~/mongodb/data/db:/data/db -d mongo
```

CLI Command	Description
–rm	remove container when stopped
–name mongo-dev	give container a custom name
-v ~/mongodb/data/db/data/db	map the container volume 'data/db' to a bind mount '~/mongodb/data/db'
-d mongo	run mongo container as a daemon in the background

Access MongoDB

Connect to MongoDb There are 2 steps to accessing the MongoDB shell.

1. Firstly, access the MongoDB container shell by executing the following command:

   ```
   docker exec -it mongo-dev bash
   ```

   This will open an interactive shell (bash) on the MongoDB container.

2. Secondly, once inside the container shell, access the MongoDB shell by executing the following command:

   ```
   mongo localhost
   ```

Show Databases Once connected to MongoDB shell, run the following command to show a list of databases:

```
show dbs
```

Create New Database Create a new database and collection

```
use test
db.messages.insert({"message": "Hello World!"})
db.messages.find()
```

Creating Images

Create custom Alpine Linux image with GIT setup

1. Create project folder with empty Dockerfile

   ```
   cd ~
   mkdir -p projects/docker/alpine-git
   cd !$
   touch Dockerfile
   ```

2. Create Dockerfile

   ```
   FROM alpine:latest

   LABEL author="codesaucerer" \
         description="Official Alpine image with GIT and VIM installed"

   ENV WORKING_DIRECTORY=/projects

   RUN apk update && apk add git vim

   RUN mkdir $WORKING_DIRECTORY

   WORKDIR $WORKING_DIRECTORY
   ```

3. Build Dockerfile

   ```
   docker image build -t [docker-username]/alpine-git
   docker run --rm -it [docker-username]/alpine-git /bin/sh
   ```

4. Push Dockerfile

   ```
   docker login
   docker push [docker-username]/alpine-git:latest
   ```
