Docker Commands

The Docker documentation is very good and has plenty of examples. You can find a detailed description for each command, e.g docker run

Your docker cli also has help built-in.

```
# Docker built-in help
$ docker --help

# You can get help for a specific command with
$ docker run --help
```

Docker Version

```
# To show the Docker client version
docker --version

# To show client and server version
docker version
```

If the `docker version` fails with `Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?`. Try to start your server again, pay attention to any error messages during startup.

Docker on windows

If Docker is used with git bash and you get the error message `the input device is not a TTY`, try to run it with `winpty`.

```
# So use winpty before docker (git bash on windows only)
winpty docker run -P nginx:alpine
```

Docker exit status

When your docker command fails with a non-zero exit status, it can originate from e.g the docker daemon, a wrongful command inside the container or the command inside the container.

If the Docker daemon is not running

```
$ docker run nginx:alpine; echo $?

docker: Cannot connect to the Docker daemon at
unix:///var/run/docker.sock. Is the docker daemon running?.
See 'docker run --help'.
125
```

If the command is not found within the container

```
$ docker run nginx:alpine do_something; echo $?
/docker-entrypoint.sh: exec: line 38: do_something: not found
127
```

If the command inside the container failed with a exit status

```
$ docker run alpine /bin/sh -c 'exit 3'; echo $?
3
```

Docker ps

Usage: docker ps [OPTIONS]

List containers (old style)

Options	Description
--all , -a	Show all containers (default shows just running)
--latest , -l	Show the latest created container (includes all states)
--quiet , -q	Only display container IDs

Docker ps examples

Docker ps is a good start to get information about containers. By default it prints the columns `container id`, `image`, `command`, `created`, `status`, `ports`, `names`. Container id or name can be used by other docker commands, e.g. `docker logs cool_hermann` (or container id) would print the logs of a container with the name `cool_hermann`.

The Ports column shows information about ports, e.g. open ports from the host to the container. `0.0.0.0:55001->80/tcp` means that all possible interfaces on the host has a tcp bind to port 55001 mapping to port 80 inside the container. So in this example you would open http://127.0.0.1:55001 in your browser, docker will forward your request to port 80 inside the container and return the result.

```
# List all running containers
$ docker ps

CONTAINER ID   IMAGE          COMMAND                  CREATED
STATUS         PORTS                    NAMES
2184230b4834   nginx:alpine   "/docker-entrypoint.…"   2 seconds ago   Up
2 seconds   0.0.0.0:55001->80/tcp   cool_hermann

# list all containers
$ docker ps --all

CONTAINER ID   IMAGE          COMMAND                  CREATED
STATUS                       PORTS                    NAMES
e5937e902c2f   nginx:alpine   "/docker-entrypoint.…"   16 seconds ago
Exited (0) 14 seconds ago                             great_satoshi
2184230b4834   nginx:alpine   "/docker-entrypoint.…"   3 minutes ago    Up
3 minutes                0.0.0.0:55001->80/tcp   cool_hermann
```

Docker container

Docker container run

Usage: docker container run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

Options	Description
-d, --detach	Run container in background and print container ID
-i, --interactive	Keep STDIN open even if not attached
--name string	Assign a name to the container
-p, --publish	Publish a container's port(s) to the host
-P, --publish-all	Publish all exposed ports to random ports
--rm	Automatically remove the container when it exits
-t, --tty	Allocate a pseudo-TTY

Docker alpine linux examples

Read more about the alpine linux image at Docker Hub alpine

```
# Run a alpine linux container, that does nothing
$ docker container run alpine

# Run a alpine linux container, and list current working directory
$ docker container run alpine ls
```

Docker nginx examples

Read more about the nginx image at Docker Hub nginx

```
# Run a nginx container
$ docker container run nginx:alpine

# Run a nginx container and open a port 7777 on the host
$ docker container run -p 7777:80 nginx:alpine

# You can try to access the server in the browser with url:
http://127.0.0.1:7777/
```

By default the container start in attached mode, the container is then attached to the terminal, e.g. logs are printed in the terminal. If you close or interrupts the process, your container will exit. Another solution is to

run the container in detached mode, in the background. Then you can start multiple container the will be handled by the docker daemon. Attached mode is good for testing scripts, debugging, learning. Detached mode is good for long running services, production usage, a local development database or anything you want to process in the background.

```
# Run a container from Docker Hub detached (background) and with random
ports
$ docker container run -d -P nginx:alpine

# Run a container from Docker Hub detached and with the specific port
published
$ docker container run -d -p 2222:80 nginx:alpine

# You test that nginx page is accessible with
$ curl localhost:2222

# Run a container and remove it at exit
$ docker container run --rm IMAGE

# Run a container with input and tty, remove the container when it exits
$ docker container run --rm -it alpine sh
```

Docker container list

Usage: docker container ls [OPTIONS]

List containers (new style)

Options	Description
-a, --all	Show all containers (default shows just running)
-l, --latest	Show the latest created container
-q, --quiet	Only display container IDs

```
# List running containers
$ docker container ls

# List all containers (including stopped)
$ docker container ls --all

# List latest container id (including stopped)
$ docker container ls -lq
```

Docker container stop

Usage: docker container stop [OPTIONS] CONTAINER [CONTAINER...]

Stop one or more running containers

```
# Stop a running docker container
$ docker stop CONTAINER

# Stop all running containers (linux/mac)
$ docker stop $(docker ps -q)
```

Docker container start

Usage: docker container start [OPTIONS] CONTAINER [CONTAINER...]

Start one or more stopped containers

```
# Start a stopped container
$ docker container start CONTAINER
```

Docker container exec

Usage: docker container exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container

Options	Description
-i, --interactive	Keep STDIN open even if not attached
-t, --tty	Allocate a pseudo-TTY

```
# Open a shell on a running container
$ docker exec -it CONTAINER sh

# List a folder on a running container
$ docker exec -it CONTAINER ls /usr/bin
```

Docker build

Usage: docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile

Options	Description
-f, --file string	Name of the Dockerfile (Default is 'PATH/Dockerfile')
--no-cache	Do not use cache when building the image
--pull	Always attempt to pull a newer version of the image
-t, --tag list	Name and optionally a tag in the 'name:tag' format

```
# Build a image used locally
$ docker built -t NAME .

# Build a image that can be pushed to remote registry
$ docker build -t registry.gitlab.com/username/project-name .
```

Docker push

Usage: docker push [OPTIONS] NAME[:TAG]

Push an image or a repository to a registry

```
# Before pushing to a remote registry you must first log in (once).

# Enter your username when prompted and then your password or personal
access token
$ docker login registry.gitlab.com

# Push build image to gitlab container registry
$ docker push registry.gitlab.com/username/project-name
```

Docker Cleanup

```
# Basic cleanup
$ docker system prune

# To remove all unused images add -a
$ docker system prune -a

# To remove everything unused
$ docker system prune -a --volumes
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all volumes not used by at least one container
  - all images without at least one container associated to them
  - all build cache
```