

### Start

<code>docker version</code>	get the docker version
<code>docker run hello-world</code>	check if everething works
<code>docker run -d redis</code>	-d for run in background (detach mode)
<code>docker ps</code>	show all runs. <code>ps</code> stands for 'process status'
<code>docker ps -a</code>	show all run / paused / stoped
<code>docker stop CONTNAME</code>	stop a running container with his name
<code>docker rm CONTNAME</code>	remove the instance
<code>docker rm \$(docker ps -a -q)</code>	remove all listed instances
<code>docker system prune</code>	Erase all /!\ use for big clean
<code>docker kill CONTNAME</code>	stop close nicely with maybe some save, message etc Kill will kill now

### DOCKER IMAGE

<code>docker image ls</code>	List all your images
------------------------------	----------------------

### Useful flags

<code>ls</code>	List running
<code>ls --all</code>	List all
<code>ls -aq</code>	List all in quiet mode

### REDIS

<code>docker run redis</code>	will start the redis image
<code>docker exec -it 0c04f252f8d2 redis-cli</code>	In an other terminal run this command to access the redis instance with redis-cli cli

`exec` : execute multi command  
`-it` : interactive mode == `-i -t` : (`-i` attach to stdin) (`-t` is use to format nicely what you see)

### USEFUL COMMANDS

<code>docker cp folder/test.txt CONTNAME:/path</code>	CP - Copy file to/from docker container:
<code>docker image inspect IMAGENAME</code>	INSPECT - use to inspect info about an image
<code>docker logs</code>	Access to logs useful when build, run fail

### DELETE IMAGES & CONTAINERS

<code>docker rmi &lt;your-image-id&gt;</code>	delete docker image (no container linked)
<code>docker container rm &lt;container_id_1&gt;</code>	Delete container
<code>docker image prune -a</code>	delete all images
<code>docker container prune</code>	delete all stopped containers

### Working with volume

<code>VOLUME ["app/tempfile"]</code>	Add anonymous volume for temp files in Dockerfile
<code>docker volume ls</code>	List all volumes available
<code>docker run -p 3000:3000 --rm --name test-app -v tosave:app/saveme imageN-ame:info</code>	Create a named volume to get data persistant
<code>docker volume rm VOL_NAME</code>	To delete a specific volume
<code>docker volume prune</code>	To clear al volumes

### ENV & ARG

<code>ENV name value</code>	is use on docker file
<code>--env PORT=3000 or -e PORT=3000</code>	is use on <code>docker run</code> (overwrite dockerfile)
<code>--env-file .env</code>	ondocker <code>run</code> use a file to define env variable
<code>ARG DEFAULT_PORT=80</code>	is use on dockerfile for the image build
<code>ENV PORT \$DEFAULT_PORT</code>	ex of use ARG in dockerfile
<code>docker build -app:1 --build-arg DEFAULT_PORT=3000</code>	Change ARG in dockerfile on build with <code>--build-arg</code>

### TIPS 1 - CREATE START LOGS

<code>docker run = docker create + docker start</code>	
<code>docker create hello-world</code>	return a container id
<code>docker start -a 456789123dj..</code>	-a mean get output to my terminal
<code>docker logs 456789123dj..</code>	print the logs when not use the -a 456789123dj.. is the container id



### TIPS 2 - EXEC -it SH

<code>docker exec -it 0c04f252f8d2 sh</code>	open a shell in a running container
<code>cmd + d</code> or type <code>exit</code>	exit the shell
<code>docker run -it busybox sh</code>	attach stdin stdout & run shell

### DOCKERFILE

Create a <code>Dockerfile</code>	FROM WORKDIR COPY RUN EXPOSE CMD
<code>docker build .</code>	run this command to build your image
<code>=&gt; Successfully built b3cc42a9ef20</code>	Return you the image id
<code>docker build -t myredis .</code>	Add a tag to the image
<code>mcz/redis:latest</code>	tag convention: <code>dockerid/projectname:version</code>
<code>FROM node:alpine</code>	take the node base image
<code>COPY ./ ./</code>	copy all file inside our project
<code>WORKDIR /user/app</code>	Set a default safe directory to pull files
<code>RUN npm install</code>	node classic action
<code>ENV PORT 3000</code>	Use env variable in your docker file <code>EXPOSE \$PORT`</code>
<code>CMD ["npm", "start"]</code>	Script start in package.json
<code>docker build -t john/node-server .</code>	Build image with a tag name
<code>docker run -p 8080:8080 john/nodeserver</code>	Set port mapping to enable incoming traffic
<code>docker ps</code>	list running container with id
<code>docker exec -it c66323-38e690 sh</code>	Get a shell inside your running container
<code>docker build -f Dockerfile.dev</code>	To run a dev version called <code>Dockerfile.dev</code>

### DOCKER COMPOSE

<code>docker-compose</code>	List all cmd available
<code>docker-compose up</code>	like <code>docker run</code>
<code>docker-compose up --build</code>	build image + run
<code>docker-compose up -d</code>	With <code>-d</code> run in background
<code>docker-compose down</code>	Stop the containers
<code>restart: on-failure</code>	LIST OF RESART CMD
<code>"no"</code>	never restart
<code>always</code>	restart anyway
<code>on-failure</code>	restart on <code>process.exit(34)</code> value > 0
<code>unless-stoped</code>	always unless the dev stop
<code>docker-compose ps</code>	Only works in the current directory

### Terminology

<code>SIGTERM</code>	terminate signal
<code>SIGKILL</code>	Kill signal

