DOCKER COMMANDS: A CHEAT SHEET



Containerization technologies have revolutionized the software development landscape. By allowing developers to package applications and their dependencies into isolated environments known as containers, they have simplified the development process, enabling applications to run consistently across various environments. Docker is one such example of a leading platform in the containerization space.

What does Docker do?

<u>Docker's purpose</u> is to build and manage compute images and to launch them in a container. Docker enables managing microservices, facilitating efficient resource utilization, and accelerating application delivery, making it a vital tool for modern DevOps practices.

Here's a cheat sheet on the top Docker commands to know and use.



(This is part of our *Docker Guide*. Use the right-hand menu to navigate.)

Images and containers

The docker command line interface follows this pattern: docker <COMMAND>

docker images docker container

The docker images and container commands grant access to the images and containers. From here, you are permitted to do something with them, hence:

docker images <COMMAND> Docker container <COMMAND>

There are some basic docker commands:

- is lists the resources.
- **cp** copies files/folders between the container and the local file system.
- **create** creates new container.
- diff inspects changes to files or directories in a running container.
- logs fetches the logs of a container.
- **pause** pauses all processes within one or more containers.
- **rename** renames a container.
- run runs a new command in a container.
- start starts one or more stopped containers.
- stop stops one or more running containers.
- stats displays a livestream of containers resource usage statistics.
- **top** displays the running processes of a container.

View resources with Is

docker images ls
docker container ls

From the **container ls** command, the container id can be accessed (first column).

		2. rah	ul@Lethalbrains: ~ (zsł	n)		
<pre>> \$ docker containe fc6b0cf9bba77b62576</pre>	r rundetach nginx 848048c07e143715be92	2b5a3781008e02893fd9eba16				node-6.10.0
lethalbrains ~ <rub > S docker containe CONTAINER ID fc6b@cf9bba7</rub 	y-2.1.3> r ls IMAGE nginx	COMMAND "nginx -g 'daemon"	CREATED 6 seconds ago	STATUS Up 5 seconds	PORTS 80/tcp	[21:01:44] node-6.10.0 NAMES blissful_benz
lethalbrains ~ <rub > \$ docker containe 660ed872630c6d32bcd</rub 	y-2.1.3> r rundetachpub 2c6e6184f57726a9293f	lish 7777:80 nginx 1b1d9bf2ba0519d16cb2388a0				[21:01:49] node-6.10.0
lethalbrains – <rub > 5 docker containe CONTAINER ID 660ed872630c fc6b0cf9bba7</rub 	y-2.1.3> r ls IMAGE nginx nginx	COMMAND "nginx -g 'daemon" "nginx -g 'daemon"	(REATED 2 seconds ago 14 minutes ago	STATUS Up 2 seconds Up 14 minutes	PORTS 0.0.0.0:7777->80/tcj 80/tcp	[21:15:45] node-6.10.0 NAMES youthful_jones blissful_benz
lethalbrains ∼ ⟨rub > S	y-2.1.3>					[21:15:48] node-6.10.0

Control timing with start, stop, restart, prune

- start starts one or more stopped containers.
- stop stops one or more running containers.
- restart restarts one or more containers.
- prune (the best one!) removes all stopped containers.

```
docker container stop <container id>
docker container start <container id>
docker container restart <container id>
docker container prune <container id>
```

Name a container

docker run -d -name myfirstcontainer

View vital information: Inspect, stats, top

docker container inspect <container id>

docker container top <container id>

docker container stats <container id>

• stats displays a live stream of container(s) resource usage statistics

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS	
a736f0f5dcdb	label-studio	0.06%	61.34MiB / 1.943GiB	3.08%	2.43kB / ØB	0B / 0B	1	

• top displays the running processes of a container:

i	~ home\$ docker container top	a736f0f5dcdb	
PID	USER	TIME	COMMAND
1699	root	0:05	{label-studio} /usr/local/bin/python /usr/local/bin/label-studio start my_project

• **inspect** displays detailed information on one or more containers. With inspect, a JSON is returned detailing the name and states and more of a container.

```
:~ home$ docker container inspect a736f0f5dcdb
    {
        "Id": "a736f0f5dcdbf943d381426dc515a5a767b14bf6f2c2cb476598cf06670eae74",
        "Created": "2020-05-10T22:36:14.6469835Z",
        "Path": "label-studio",
        "Args": [
            "start",
            "my_project"
        ],
"State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "00MKilled": false,
            "Dead": false,
            "Pid": 1699,
            "ExitCode": 0,
            "Error": "",
"StartedAt": "2020-07-17T22:16:13.991687584Z",
"FinishedAt": "2020-07-17T22:16:12.161403085Z"
        },
        "Image": "sha256:ee47e34c82db06fbd4e65c583721baec99d5c16054c446ee716d790e5cb
        "ResolvConfPath": "/var/lib/docker/containers/a736f0f5dcdbf943d381426dc515a5
        "HostnamePath": "/var/lib/docker/containers/a736f0f5dcdbf943d381426dc515a5a7
        "HostsPath": "/var/lib/docker/containers/a736f0f5dcdbf943d381426dc515a5a767b
        "LogPath": "/var/lib/docker/containers/a736f0f5dcdbf943d381426dc515a5a767b14
06670eae74-json.log"
        "Name": "/label-studio",
        "RestartCount": 0,
        "Driver": "overlay2",
        "Platform": "linux",
        "MountLabel": "",
```

Additional resources

For more on this topic, there's always the <u>Docker documentation</u>, the <u>BMC DevOps Blog</u>, and these articles:

- <u>Getting Started with Containers and Microservices for Enterprise Leaders</u>
- How To Introduce Docker Containers in The Enterprise
- Docker Management Tips
- Docker Monitoring: How to Monitor Containers and Microservices
- <u>Containers Aren't Always the Solution</u>