

Section 5 - Containers lifecycle

3 - Container Shell

Overview - Getting a Shell Inside Containers

1. `docker container run -it` => start new container interactively
2. `docker container exec -it` => run additional command in existing container
3. Default shell CMD of different Linux distributions in containers

Start new container interactively

- Use the `-it` options with the `docker run` command to get an interactive shell command prompt inside a container.
- From Docker CLI help documentation

```
$ docker run --help
```

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

```
Run a command in a new container
```

```
...  
-i, --interactive           Keep STDIN open even if not attached
```

```
...  
-t, --tty                   Allocate a pseudo-TTY
```

```
-t pseudo-TTY => simulates a real shell terminal like SSH does.
```

```
-i interactive => keep the STDIN channel open to input commands
```

Note that the `-it` are two separate options.



Example

- Run `bash` command in nginx container.

```
# docker container run -it --name proxy nginx bash  
root@27056e1170d7:/#
```

- The "-it" options are used to open an interactive terminal.
- The "-name" option is used to set the name of the container.
- After the image name, the `bash` argument is used to specify the COMMAND to execute inside the container. This will override the image default CMD.
- The `bash` command will provide a shell prompt from which we can execute linux commands inside the container.
- `nginx` default CMD [Dockerfile](#)

```
CMD ["nginx", "-g", "daemon off;"]
```

Example (2a)

From the container command line prompt `root@27056e1170d7:/#` we can see:

- The shell user is "root"
- The system hostname is "27056e1170d7", which is also the container ID.
- We can now execute commands as we could do from a regular system.

Example (2b)

- For example we can list all files inside the container:

```
root@27056e1170d7:/# ls -la
total 8
...
drwxr-xr-x    2 root root 4096 Mar 26 12:00 bin
drwxr-xr-x    2 root root    6 Feb  3 13:01 boot
drwxr-xr-x    5 root root  360 Apr  8 06:35 dev
drwxr-xr-x    1 root root   66 Apr  8 06:35 etc
drwxr-xr-x    2 root root    6 Feb  3 13:01 home
...
```

Example (3)

From here we can perform all kind of administrative task such as:

- change config files
- download and install packages from the internet etc...

- Use the `exit` command to exit from the container shell:

```
root@27056e1170d7:/# exit
exit
[root@<docker-host>] #
```

This will return back to the host shell prompt.

Example (4a)

- Verify if the nginx "proxy" is still running.

```
# docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	... STATUS	PORTS	NAMES
27056e1170d7	nginx	"bash"	... Exited (0)...		proxy
515d82f84f9e	mysql	"..."	... Up 2 minutes	3306/tcp...	mysql
1b76d91c5f77	nginx	"nginx..."	... Up 3 minutes	80/tcp	nginx

After exiting the bash shell, the container "proxy" is not running. Why ?

Example (4b)

- The default command for an nginx container is to run the nginx program itself. We changed that default program to actually be `bash`, giving us the shell prompt.
- When we exited the shell, the container stopped.
- Because the main application (PID=1 "default command") running inside the container is `bash`, exiting the bash shell will stop the main `bash` process ((PID=1)).

container shell

- One of the most important operations is to access the shell of the container and execute Linux commands.
- Note that there is no need to have an ssh server running inside the container to actually access the shell prompt of the container.
- Docker CLI will provide access to the container shell.

Full linux distribution containers

- In the following example we will use a full linux distribution such as Ubuntu to run a container.

```
# docker container run -it --name ubuntu ubuntu
...
Status: Downloaded newer image for ubuntu:latest
root@1da52b3057f6:/#
```

Note that the default CMD of the ubuntu image is bash, so we do not need to specify it.

Ref: [Dockerfile](#)

```
...
CMD ["/bin/bash"]
```

typical operations (1)

- We will go through some typical operations performed within an ubuntu system.
- We will use the package manager to install the `curl` tool.

```
root@1da52b3057f6:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
...

root@1da52b3057f6:/# apt-get install -y curl
...
Setting up curl (7.58.0-2ubuntu3.6) ...
...
```

Note that the `curl` tool is not available in the ubuntu image because to keep the image small in size a minimal number of tools have been installed

typical operations (2b)

- In this example, the ubuntu running container has a curl installed and we can use it as we would do on a local machine.

```
root@1da52b3057f6:/# curl google.com
<HTML>
...
</HTML>
root@1da52b3057f6:/#
```

typical operations (3)

- Exit the ubuntu shell:

```
root@1da52b3057f6:/# exit  
exit  
[root@<docker-host>] #
```

This will **stop** the ubuntu container.

typical operations (4)

- Verify which container is running and which is stopped.

```
# docker container ls
```

CONTAINER ID	IMAGE	COMMAND	...	STATUS	PORTS	NAMES
515d82f84f9e	mysql	"..."	...	Up 26 minutes	3306/tcp...	mysql
1b76d91c5f77	nginx	"nginx..."	...	Up 27 minutes	80/tcp	nginx

ubuntu container is not running since we exited from the bash shell **the default application.**

typical operations (5)

- Use the `docker container ls -a` command to list stopped containers.

```
# docker container ls -a
```

CONTAINER ID	IMAGE	COMMAND	...	STATUS	PORTS	NAMES
1da52b3057f6	ubuntu	<code>"/bin/bash"</code>	...	Exited (0)...		ubuntu
27056e1170d7	nginx	<code>"bash"</code>	...	Exited (0)...		proxy
515d82f84f9e	mysql	<code>"..."</code>	...	Up 29 minutes	3306/tcp...	mysql
1b76d91c5f77	nginx	<code>"nginx..."</code>	...	Up 30 minutes	80/tcp	nginx

If we start the specific ubuntu container (`ID="1da52b3057f6"`) again, then this container will have curl tool installed on it. But if we create a new container from the ubuntu image (`docker container run ubuntu`), that different container will not have the curl tool installed on.

docker container exec

- Use the `docker container exec` command to get shell prompt of a running container
- In the following examples we have two running containers mysql and nginx:

```
# docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
515d82f84f9e	mysql	"docker-entrypoint.s..."	About an hour ago	Up
1b76d91c5f77	nginx	"nginx -g 'daemon of..."	About an hour ago	Up

The "docker container exec" command can be used to execute any command inside a container.

docker container exec - example mysql (1)

- Use the "docker container exec" command to access the shell prompt of the mysql running container.

```
# docker container exec -it mysql bash  
root@515d82f84f9e:/#
```

- The `-it` options are used to open an interactive terminal.
- "mysql" is the name of the container (or the ID) on which we want to execute the command.
- `bash` is the command to execute in the container.
- This will create a new process inside the container related to the `bash` command.
- The shell user of the mysql container is the "root" user.
- The system hostname of the container is the "515d82f84f9e", which is also the container ID.

docker container exec - example mysql (2)

- Use the `ps aux` command to see the processes running inside the container.

```
root@515d82f84f9e:/# ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
mysql	1	1.0	10.2	1345576	395896	?	Ssl	08:01	0:45	mysqld
root	180	0.0	0.0	18172	2100	pts/0	Ss	09:13	0:00	bash
root	496	0.0	0.0	36624	1584	pts/0	R+	09:15	0:00	ps aux

Note that the "ps" command is not included any more in the official mysql image by default.

To install the ps command:

```
# apt-get update
# apt-get install procps
```

docker container exec - example mysql (3)

- Exit the bash shell and verify the container status.

```
# exit
```

```
exit
```

```
# docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
515d82f84f9e	mysql	"docker-entrypoint.s..."	About an hour ago	Up About
1b76d91c5f77	nginx	"nginx -g 'daemon of..."	About an hour ago	Up About

- The mysql container is still running because the exit command did not stop the default (main) application running.
- The default main application running inside a container has PID = 1.
- In this example the application with PID = 1 is actually the mysqld daemon.
- The `docker container exec` command actually runs an additional process on an existing running container.

docker container exec - example mysql (4)

- Docker use Linux namespaces to provide isolation for running processes
- A process might have the apparent PID 1 inside a container, but if we examine it from the host system, it would have an ordinary PID

```
# docker container top mysql
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
polkitd	3729	3713	1	12:53	?	00:00:05	mysqld

```
# ps aux | grep mysql
```

polkitd	3729	0.9	38.9	1366564	394964	?	Ssl	12:53	0:08	mysqld
---------	------	-----	------	---------	--------	---	-----	-------	------	--------

```
# docker container exec mysql ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
mysql	1	0.9	38.9	1366564	394964	?	Ssl	09:53	0:07	mysqld
root	493	0.0	0.1	36624	1528	?	Rs	10:06	0:00	ps aux

Alpine Linux (1)

- Alpine is a Linux distribution designed to be very small in size. It's actually only 5MB.

```
# docker pull alpine
# docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysql	latest	7bb2586065cd	12 days ago	477MB
nginx	latest	2bcb04bdb83f	12 days ago	109MB
ubuntu	latest	94e814e2efa8	3 weeks ago	88.9MB
alpine	latest	5cb3aa00f899	4 weeks ago	5.53MB

- We used the "docker pull" command to download the latest alpine image from the docker.hub registry.
- We used the "docker image ls" to list all images available in the local cache.
- Do not worry, we will cover Docker images in depth in the next section.

Alpine Linux (2)

- The Alpine Linux distribution comes with its own package manager **apk**.
- The **bash** shell is not available in the alpine image.

```
# docker container run -it alpine bash
docker: Error response from daemon: OCI runtime create failed: container_l
Result:
\"bash\": executable file not found
```

The alpine image is so minimal that does not contain the **bash** shell. Instead, it contains the **sh** shell, which is not fully featured as Bash is.

Alpine shell

- Use the `sh` command to get access to the alpine shell.

```
# docker container run -it alpine sh  
/ #
```


Exercise

- Ref:
- D_S5_L3_Container_Shell_ex.md