Agenda

Difference between Management Modules and Command.

How to create Container?

Difference between Create and RUN command.

When we use background and foreground option with Run command.

In the Lab section, we will learn how to create container.

Create, Start and attach the container.

Image pull for different version.

How to list the running and shutdown containers?

How to delete the container?

Create a new Container

https://docs.docker.com/engine/reference/commandline/create/

Create a new container. (Only Create)

Create the container only ?

docker container create

Create the Container and Start it?

docker container run

Create the Container, Start and then execute some command?

docker container run

What options, we can use with run

docker container run [**OPTIONS**] **IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]**

Let's explore more [OPTIONS] :

docker container run **background** or **foreground**

When starting a Docker container, we should know that to run the container in the background in a "detached" mode or in the default foreground mode.

To start a container in detached mode, you use -d=true or just -d option.

When we execute the command "docker container run -d (detached) waits for the process being run to exit.

In simple words, we can say that with "–d" option, it first start container, then execute the command and after that shutdown/or keep running the container.

Further, If you use -d with --rm, the container is removed when it exits. "--rm" option will delete the newly created container. This option is useful when we do the testing if our container is working as expected.

We will cover in our upcoming slides.

In Linux, whenever, we execute a script we take the input for variable from terminal that we called STDIN or

When this script produce output, we called this STDOUT

and script gives some error we called at STDERR.

Foreground

In this mode, docker run can start the process in the container and attach the console to the process's standard input, output, and standard error.

All of that is configurable:

-a=[ ] : Attach to `STDIN`, `STDOUT` and/or `STDERR`

-t : Allocate a pseudo-tty (Terminal)

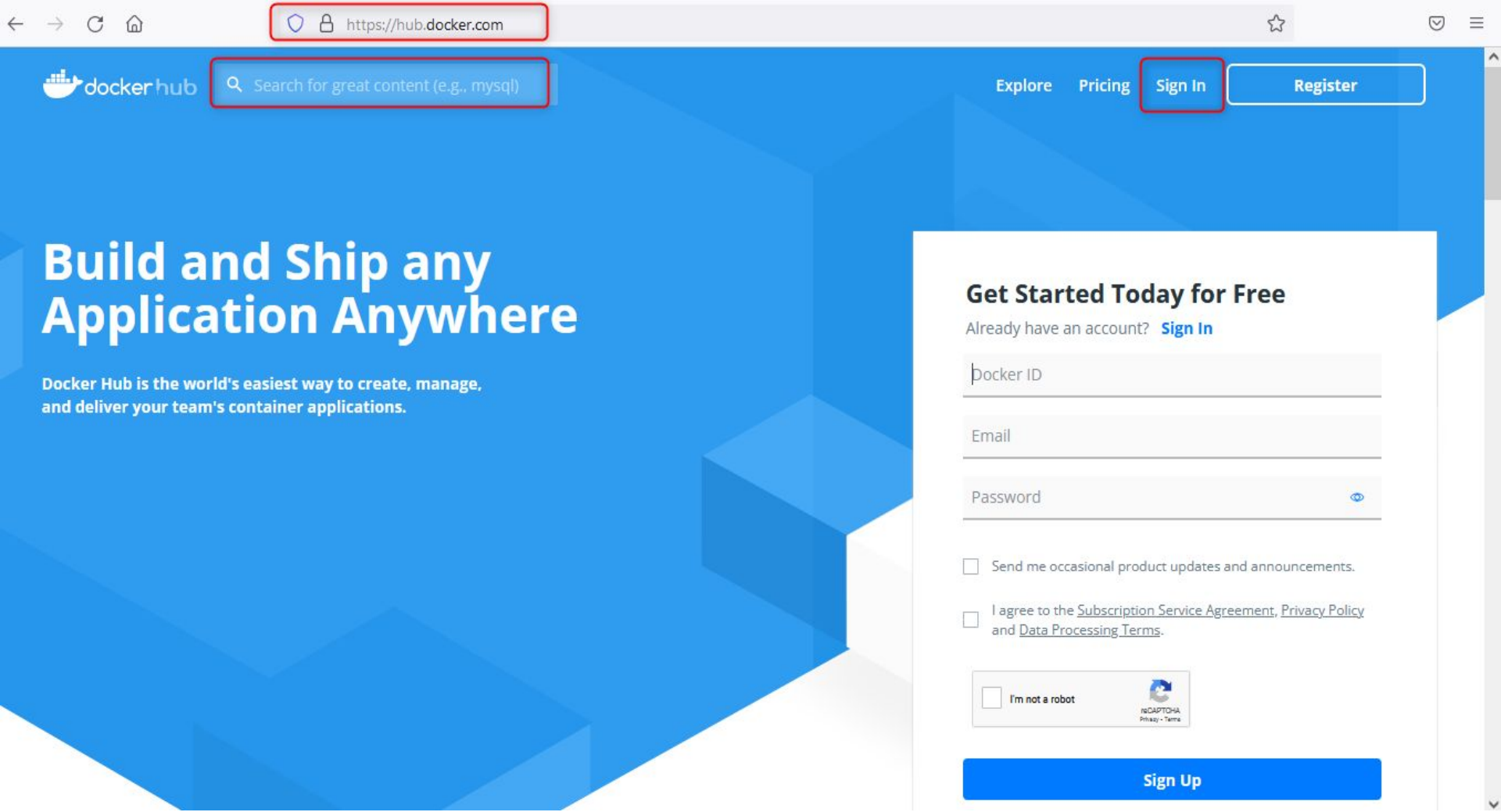
--sig-proxy=true : Proxy all received signals to the process (non-TTY mode only)

-i : Keep STDIN open even if not attached

If you do not specify **-a** then Docker will attach to both **stdout and stderr** . You can specify to which of the three standard streams (STDIN, STDOUT, STDERR) you'd like to connect instead, as in:

docker run -a stdin -a stdout -i -t ubuntu /bin/bash

For interactive processes (like a shell), you must use -i -t together in order to allocate a tty for the container process. -i -t is often written ***-it*** as you'll see in later examples


https://hub.docker.com
docker hub
Search for great content (e.g., mysql)
Explore
Pricing
Sign In
Register
Build and Ship any Application Anywhere
Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications.
Get Started Today for Free
Already have an account? Sign In
Docker ID
Email
Password
Send me occasional product updates and announcements.
I agree to the Subscription Service Agreement, Privacy Policy and Data Processing Terms.
I'm not a robot
reCAPTCHA
Privacy - Terms
Sign Up

Let’s do the lab

https://github.com/cfitechops/Docker/blob/main/01-docker.md

How to stop and Start the Container?

start : Start one or more stopped containers.

stop : Stop one or more running containers.

pause : Pause all processes within one or more containers.

unpause : Unpause all processes within one or more containers.

Difference between Pause and Stop option

https://docs.docker.com/engine/reference/commandline/pause/

The docker pause command suspends all processes in the specified containers.

On Linux, this uses the freezer cgroup.

SIGSTOP = SIG (Signal) STOP => STOP Signal. Its behaviour is to pause that process. So, When this signal send to process, it pause that process in its current state. And resume execution if it is sent the **SIGCONT** signal.

Container would be in **remain in memory** portion till it is in pause state.

It will use the memory portion again if we unpause this container.

Use case would be: pause the resource which do intensive tasks that can be resumed after some time.

Pause option could be used **in future** for live migration of container from one host to another.

https://docs.docker.com/engine/reference/commandline/stop/#options

The main process inside the container will receive SIGTERM, and after a grace period, SIGKILL.

--time,-t 20 Seconds to wait for stop before killing it

SIGTERM: SIG (SIGNAL) TERM (Termination) : Its behavior is to terminate the process. The SIGTERM gracefully kills the process whereas SIGKILL kills the process immediately.

However, with Stop option, it releases the memory used, once container is stopped.

Use case would be to stop the container if no use.

How to stop and Start the Container ?

start: Start one or more stopped containers

stop: Stop one or more running containers

pause: Pause all processes within one or more containers

unpause: Unpause all processes within one or more containers

docker container start containerID/container-name

docker container stop containerID/container-name

docker container pause containerID/container-name

docker container unpause containerID/container-name