The Interactive and TTY Options in docker run

Last modified: December 23, 2022

by Chin Ming Jun (https://www.baeldung.com/linux/author/chinmingjun)

1. Overview

While running Docker containers, we've always come across the *-i* and *-t* options whenever we want to drop into the shell of a running container. But, what do the two actually do? Furthermore, what does it mean when Docker errors with the message "input device is not a TTY"?

In this tutorial, we'll look at the *-i* and *-t* options of the *docker run* command in depth. Specifically, we'll learn how these two different options enable an interactive mode of the process in a Docker container.

Because both *docker run* and *docker exec* share these options, we'll be referring only to the *docker run* command for brevity.

2. Enabling Input Using the *-i* Option

By default, the *docker run* command does not attach the standard input stream (STDIN) of the process within the container to the host terminal. However, it does connect the standard output (STDOUT) and standard error (STDERR) streams. In other words, we'll only see the output being printed without a way to send any input to it.

If the main process of the container is expecting input, running it without attaching the STDIN will cause it to exit immediately.

2.1. Running a Process With Input Prompt Without Attaching STDIN

To demonstrate the effect, let's run the *passwd* command using an Ubuntu container:

```
$ docker run ubuntu passwd root
New password: Password change has been aborted.
passwd: Authentication token manipulation error
passwd: password unchanged
```

We choose the *passwd* command for this demonstration because it's interactive and requires a user's manual input. Therefore, when we start it without attaching an active STDIN, the command immediately exits with an error.

To attach the STDIN from the host terminal to the main process within the container, we pass the *-i* option when invoking *docker run*:

```
$ docker run -i ubuntu passwd root
New password:
```

In this case, we see that the command now waits for our input.

When we pass the *-i* option, the *docker run* command attaches the input device to the main process within the container. Specifically, the input device it takes is the input device to the *docker run* command. **Note the term "input device" here because the input device is not necessarily the terminal.**

2.2. Attaching Output Pipe to STDIN

We can also attach the output pipe to the STDIN of the process within the Docker container. For instance, let's pipe the output of the *echo* command to a *cat* process in the container:

```
$ echo "This is a piped input" | docker run -i ubuntu cat
This is a piped input
```

The example above starts the *cat* process inside the container without any argument. When invoked, *cat* echoes any input coming from STDIN indefinitely until it reaches the end of the stream. When we pipe the output of the *echo* command to the *cat* process, it terminates after it echoes a single line. This is because the STDIN has reached the end of the stream.

Contrast this with the behavior of the *cat* command when we attach the terminal to the STDIN:

```
$ docker run -i ubuntu cat
indefinite stream
indefinite stream
continue to read
continue to read
```

In this example, the first line is the input we've keyed, and when we press enter, the *cat* process simply echoes the input string. Note that in this case, the *cat* process continues indefinitely until its STDIN is closed.

3. Allocating the Pseudo-Terminal With the *-t* Option

From the official documentation, Docker states that the *-t* option will "allocate a pseudo-TTY" to the process inside the container. TTY stands for Teletype and can be interpreted as a device that offers basic input-output. **The reason it's a pseudo-TTY is that there's no physical teletype needed, and it's emulated using a combination of display driver and keyboard driver.**

For the sake of this article, it's sufficient to think of a pseudo-TTY as a terminal console we're using for running commands and reading output in Linux.

3.1. Missing Functionality Without the *-t* Option

In the previous section, we've seen how the *-i* option attaches the terminal to the main process within the container. The implication is that we can interact with the process as if we're running it on the host. However, **without passing the *-t* option to *docker run* for the interactive mode, not all of the terminal-specific functionality will be working**.

For example, most programs that prompt the user for a password will turn off the echo of the input. This allows the password to be hidden from the console.

Let's run the *passwd* command with the *-i* option and complete the prompt:

```
$ docker run -i ubuntu passwd root
New password: thisisanewpassword
Retype new password: thisisanewpassword
passwd: password updated successfully
```

Under typical usage, we know that *passwd*'s password prompt does not display our password input. The way *passwd* achieves this is by turning off the echo option of the terminal. Since we started this process without allocating a pseudo-TTY, the echo-off functionality of the terminal is not taking effect here.

Additionally, some programs like *bash* display output differently if they detect the input is a TTY device. For instance, running *ls* in a TTY device will cause *bash* to color the file and directory names differently.

3.2. Enabling Complete Terminal Functionality Using the *-t* Option

To enable complete terminal functionality, including the echo-off option, we can pass the *-t* option along with *-i*:

```
$ docker run -i -t ubuntu passwd root
New password:
Retype new password:
passwd: password updated successfully
```

Now, the echo-off option on the *passwd* command is working correctly under the pseudo-TTY environment.

Although there's nothing stopping us from passing just the *-t* option, there seems to be not much use for it. Unless the output of the command requires a TTY device, there aren't many benefits to running a container with just the *-t* option.

3.3. "*Input device is not a TTY*" Error

If we specify *-t* and then attach an output pipe to the STDIN of the container's process, the *docker run* command will complain that the "input device is not a TTY". This is because when the *-t* option is present, *docker run* will check if the input device is a TTY-like device. When it sees the input device is a pipe file, it'll exit with an error:

```
$ echo "this is a pipe input" | docker run -i -t ubuntu cat
the input device is not a TTY
$ echo $?
1
```

4. Summary

In this article, we've learned that the *-i* option of the *docker run* command attaches the STDIN of the container to the host process. Then, we've also seen how the *-t* option enables complete terminal functionality such as echo-off for password masking. Then, we also simulated the "input device is not a TTY" error by piping the pipe files and specifying the *-t* option.