About Us Write For Us Contact Us

Programming

Double Your Efficiency With

These Docker Commands- Become Unstoppable!

May 20, 2020 0 Comments

In this article, we have included some lesser-known & advanced Docker commands that you can use to increase your productivity. All these commands can be of great help while executing a little difficult task without any hassle.

If you are reading this article, then probably you are already intrigued about Docker. But for the starters, it is a standalone, lightweight, & executable package that includes everything any developer requires to run an application. Considering all the benefits of Docker, indeed, it is essential, and the demand for the experts has also grown significantly.

In this article, the idea is to inform all our readers regarding some of the lesser-used Docker commands that can help you in developing on Docker efficiently!

But before going any further, you can check out some of the ***basic Docker commands*** that can be useful for you if you are a newbie.

Basic Docker Commands

- ***docker run*** – Runs a command in a new container.
- ***docker start*** – Starts one or more stopped containers
- ***docker stop*** – Stops one or more running containers
- ***docker build*** – Builds an image form a Docker file
- ***docker pull*** – Pulls an image or a repository from a registry
- ***docker push*** – Pushes an image or a repository to a registry
- ***docker export*** – Exports a container's filesystem as a tar archive
- ***docker exec*** – Runs a command in a run-time container
- ***docker search*** – Searches the Docker Hub for images
- ***docker attach*** – Attaches to a running container
- ***docker commit*** – Creates a new image from a container's changes

Read More: ***A Cheat Sheet to Basic Docker Commands for Software Developers***

So, these are the basic commands that you can use for Docker. However, this article is all about the more advanced versions of these commands that can help developers by increasing their productivity. All of the listed below advanced Docker commands are suitable for both intermediate as well as experienced developers.

Your Cheat Sheet To Advanced Docker Commands!

1. Executing Commands Within the Container

Though running containers allow you to run the commands, you can’t be attached to containers every time. This case becomes more crucial when you already have numerous containers.

This type of case often arises, and this is why docker runtime comes with a command that can execute container-specific commands. It does so from the terminal instance of the admin.

```
$ docker exec CONTAINER_ID command
```

With the above command, you can execute the command inside the particular container.

2. Add Persistent Volumes for Containers

Till now, you must have knowledge of the fact that all data for a container is lost when you delete the container. However. In Docker, you can create volumes that allow persistent data storage. These volumes bind a container with a space that is pre-defined on your file system.

```
$ docker create -it -v $(pwd):/var/www ubuntu:xenial
```

By using the above command, it won't be difficult for you to bind your current directory to the **/var/www** location. After this, you can easily able to modify any files in the current directory with changes being shown inside Docker.

3. Using Docker Run for Creating & Running Containers

Usually, developers initially create the Docker container and then start it, but with Docker, you can do both by using just 1 command.

```
$ docker run -it -d ubuntu:xenial bash
```

After executing, this command fetches the Xenial image, creates a container, and runs it with bash. But the catch is that you won’t be able to check directly in your terminal because this command gives instructions to Docker to run the container in the background with the help ***-d*** flag. And upon elimination, you will go to the container once it’s created.

4. How to Create a Container With a Name?

You can use a below command having ***-name*** option that will create containers with a specific name. It can also be used as a replacement to ***CONTAINER_ID*** for a majority of the Docker commands.

```
$ docker create --name TEST nginx
```

The above command will help you in pulling the Nginx image from Docker Hub. Afterward, it will create a container called ***TEST*** in this case.

5. Search Any Changes in a Container

Every experienced programmer knows the fact that often small misconfigurations can lead to big problems that can lead hrs to debug. One command called

Docker diff can rescue you from these types of situations. You can use the below command that will assist you in looking for any change in your container's filesystem.

```
$ docker diff CONTAINER_ID
```

Even if you are confident that any particular container is healthy, then also you can use the above command to monitor and check all the changes from the filesystem of the container.

6. Docker Command for Creating a New Image From a Modified Container

In Docker, you can also create a completely new image based upon all the changes of an already present container. It is beneficial when you are seeking to create many containers dedicated to the various phases of your app.

```
$ docker commit CONTAINER_ID REPOSITORY:TAG
```

Read More: ***Learn How To Use Docker Images?***

7. Docker Command That Can Monitor Servers for Containers

There is an even command in Docker that you can use for monitoring servers for real-time info on images, containers, plugins, and others. You may try this using two command-line shells of your system.

```
$ docker events
```

You can use the above command in the 1st shell and then spawn a Docker container in 2nd shell. After running a few Docker commands within the container, you'll start getting notifications in real-time in the 1st terminal.

8. Docker Command for Inspecting Image Histories

In Docker, images are the building blocks for containers & to easily view image histories, you can use the below command.

```
$ docker history IMAGE_ID
```

It helps you in detecting any issues in a container so that you can resolve them. You just have to get an image id of the respective container using docker ps and then you can use it as mentioned in the above command.

9. Using Tars to Load Docker Images

There is a Docker command that not only lets you load images from tar but also load images from the standard input. Below is the command that you can use for this job.

```
$ docker load < IMAGE.tar.gz
```

Apart from this, you can also use another command mentioned below.

```
$ docker load --input IMAGE.tar
```

Now you can list the images for checking all the effects of the commands.

10. Displaying Information (System-Wide) About Docker

You can monitor your containers easily after viewing the system-wide information about the runtime of Docker. It also provides essential information that can be useful for debuggers because of its revealing nature.

```
$ docker info
```

Though, as simple this commands looks, it provides a myriad of info not only for Docker containers but also for the host machine.

Read More: ***Learn How To Use Docker The Command Line Tools To Extract Info?***

11. Docker Command That Can Rename Containers

Renaming Docker commands is one of the most useful commands for many developers. It might look simple to you but definitely, it has a great significance for all the newbies who have entered the world of Docker.

```
$ docker rename CONTAINER NEW_NAME
```

For renaming, you can use ***CONTAINER_ID*** or even the name parameter that represents your container. In the above commands, the "***NEW_NAME***" is used for container name & you can get both the name or id of a container that uses the Docker ***ps -a*** command.

Read More: ***9 Best Orchestration Open Source Docker Tools***

Summing Up!

So, these were the Docker commands that can be useful for all our developers! If you'll ask me, it will be difficult to explain why Docker has become one of the most essential elements for the developers. Today, from big organizations to budding start-ups, almost everyone has adopted this technology and for many reasons. This has led to increased demands for skilled developers with high-paying annual salaries.

Considering every aspect of Docker, definitely getting used to with all the above-listed commands can be useful for your next project. I will hope that all of these Docker commands were enough to give you sufficient knowledge which you were looking for. And if you feel that we have missed out on anything, then do mention in the comment section below!

More Useful Resources:

- ***13 Best IDEs for C and C++ Developers- 2020 & Beyond***
- ***Best Books That Can Help You Learn Docker In No-Time!***
- ***Learn How To Stop, Kill And Clean Up Docker Containers?***
- ***Learn How To Debug Docker Containers?***

Share Article:

https://codecondo.com/double-your-efficiency

Aniket Anand

May 19, 2020

How Your Keyword Choice Affects Your

Website's Rank?

May 20, 2020

Will Paul Tudor Jones' Bitcoin Investment Paves Road for Institutional Investment?

Leave a Reply

Name

Email

Website

□ Save my name, email, and website in this browser for the next time I comment.

Comment

Post Comment

Search in resources...

Social

Twitter

FOLLOW ME!

Facebook

FOLLOW ME!

Recently published

July 26, 2020

7 Brilliant Video Marketing Ideas For Your Bank!

July 25, 2020

Is Your Website Racist? Signs To Look For!

July 24, 2020

What Is the Instagram Explore Page, and How Can You Get There?

July 23, 2020

The Ultimate Guide to Setting Up Your Business Computer Network and Ensuring It Runs Smoothly

July 22, 2020

Why Cybersecurity Matters in the Age of the Internet of Things?

Featured posts

July 2, 2020

18 Best C & C++ IDEs & Code Editors For Efficient Programming- Infographics Added!

May 13, 2020

10 Tips to Hire the Best Web Design Agency for Your Business

April 27, 2020

Top 10 Remote Work Tools to Boost Productivity in Web Design!

April 25, 2020

Branding Your Website To Generate Leads

April 1, 2020

3 Elements You Must Include When Creating a Website Design Contract

Social

Twitter
FOLLOW ME!

Facebook
FOLLOW ME!