Docker Commands — The Ultimate Cheat Sheet

August 21st 2018

If you don't already know, Docker is an open-source platform for building distributed software using "containerization," which packages applications together with their environments to make them more portable and easier to deploy.

Thanks to its power and productivity, Docker has become an incredibly popular technology for software development teams. However, this same power can sometimes make it tricky for new users to jump into the Docker ecosystem, or even for experienced users to remember the right command.

Fortunately, with the right learning tools you'll have a much easier time getting started with Docker. This article will be your one-stop shop for Docker, going over some of the best practices and must-know commands that any user should know.

Docker Commands and Best Practices

Before we get into the best practices for using Docker, here's a quick overview of the vocabulary you should know:

- **Layer**: a set of read-only files or commands that describe how to set up the underlying system beneath the container. Layers are

built on top of each other, and each one represents a change to the filesystem.

- **Image**: an immutable layer that forms the base of the container.

- **Container**: an instance of the image that can be executed as an independent application. The container has a mutable layer that lies on top of the image and that is separate from the underlying layers.

- **Registry**: a storage and content delivery system used for distributing Docker images.

- **Repository**: a collection of related Docker images, often different versions of the same application.

With that refresher in mind, here are some quick tips for building applications with Docker:

- Try to keep your images as small as possible. This will make them easier to transfer and faster to load into memory when starting a new container. Don't include libraries and dependencies unless they're an absolute requirement for the application to run.

- If your application needs to be scalable, consider using Docker Swarm, a tool for managing a cluster of nodes as a single virtual system.

- For maximum efficiency, use Docker in combination with continuous integration and continuous deployment practices. You

can use services such as Docker Cloud to automatically build images from source code and push them to a Docker repository.

Below, you'll find all of the basic Docker commands that you need to start working with containers:

Developing with Docker Containers:

- **docker create [image]**: Create a new container from a particular image.
- **docker login**: Log into the Docker Hub repository.
- **docker pull [image]**: Pull an image from the Docker Hub repository.
- **docker push [username/image]**: Push an image to the Docker Hub repository.
- **docker search [term]**: Search the Docker Hub repository for a particular term.
- **docker tag [source] [target]**: Create a target tag or alias that refers to a source image.

Running Docker Containers

- **docker start [container]**: Start a particular container.
- **docker stop [container]**: Stop a particular container.

- **docker exec -ti [container] [command]**: Run a shell command inside a particular container.

- **docker run -ti—image [image] [container] [command]**: Create and start a container at the same time, and then run a command inside it.

- **docker run -ti—rm—image [image] [container] [command]**: Create and start a container at the same time, run a command inside it, and then remove the container after executing the command.

- docker pause [container]: Pause all processes running within a particular container.

Using Docker Utilities:

- **docker history [image]**: Display the history of a particular image.

- **docker images**: List all of the images that are currently stored on the system.

- **docker inspect [object]:** Display low-level information about a particular Docker object.

- **docker ps**: List all of the containers that are currently running.

- **docker version**: Display the version of Docker that is currently installed on the system.

Cleaning Up Your Docker Environment:

- **docker kill [container]**: Kill a particular container.

- **docker kill $(docker ps -q)**: Kill all containers that are currently running.

- **docker rm [container]**: Delete a particular container that is not currently running.

- **docker rm $(docker ps -a -q)**: Delete all containers that are not currently running.