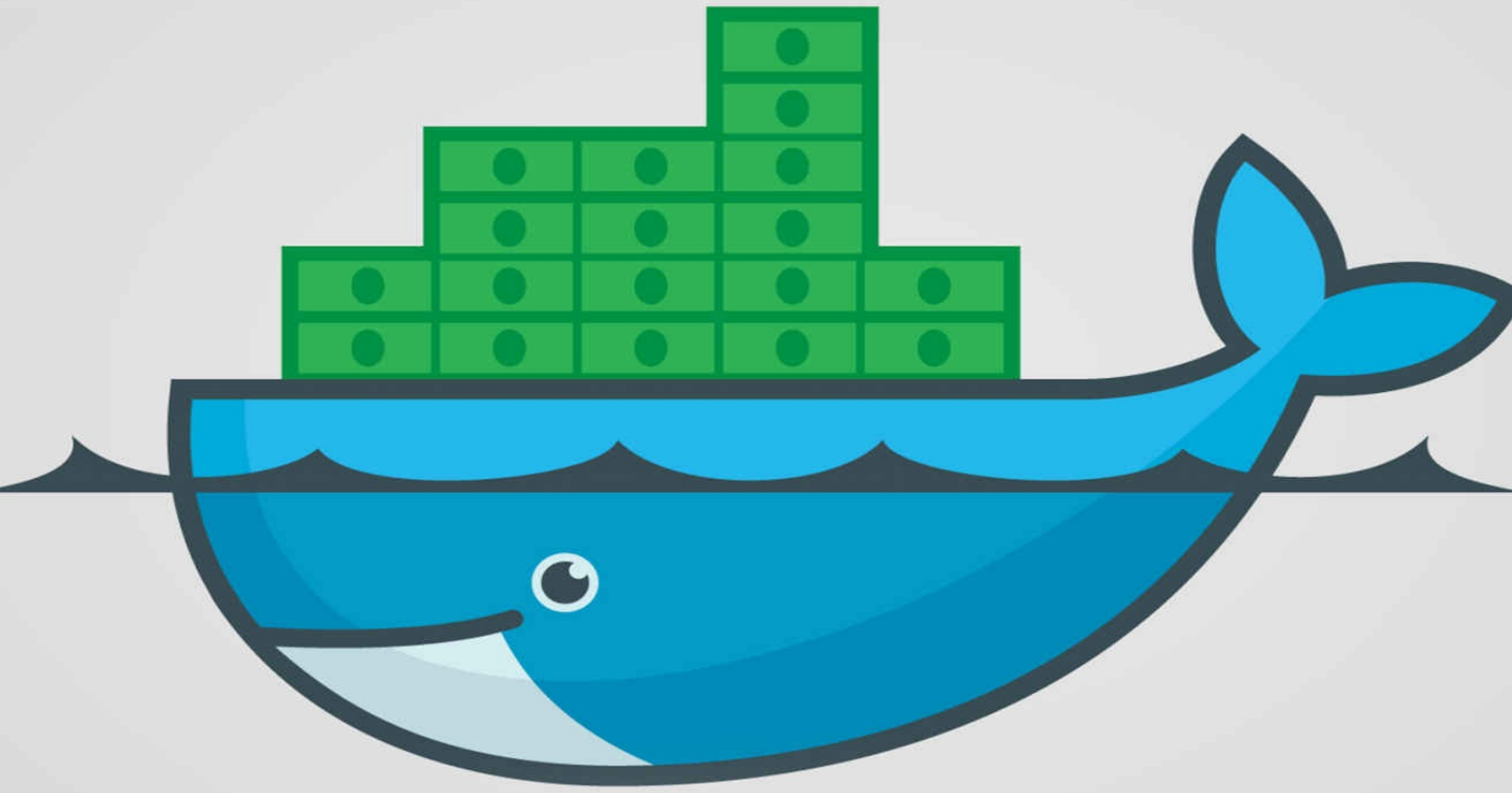
DOCKER

The Complete Introduction To Using Docker Containers Today



DAVID LARSON

Docker:

The Complete Introduction To Using Docker Containers Today

Bonus Gift For You!

Get **free** access to your complimentary book *"Amazon Book Bundle: Complete User Guides To 5 Amazon Products"* by clicking the link below.

>>>CLICK HERE TO DOWNLOAD<<<

(or go to: https://freebookpromo.leadpages.co/amazon-book-bundle/]

Table of Contents

Introduction

Docker is known as the world’s best software containerization destination. It is simply an open platform, through which you can build; ship as well as run distributed applications.

There are various things that Docker does, such as Docker Engine, Docker Hub, Docker Enterprise Support, and Docker Services & Training. There are different reasons that should make you use Docker, including: enormous changes in the software industry and the scaled out platform.

Chapter 1 – Learning Docker for the First Time

Docker Terminology

Terminology	Meaning
Docker	Refers to the world's best software containerization platform
Repository	Refers to a group of Docker images and you can publish it to a registry server so as to share it.
Registry	This is simply a hosted service that comprises of repositories of images.
Machine	Refers to a docker tool that simplifies your efforts of developing hosts on your PC.
Link	It offers you a legacy interface allowing you to connect Docker containers.
Image	Images are simply the foundation of any container.
Filesystem	This refers to a file naming and assigning of location method used by an operating system, such as Windows : NTFS
Dockerfile	Refers to a text document comprising commands that help to develop a docker image when executed manually.

Docker Hub	This is your online home/hub through which you can manage your Docker containers.
Docker Engine	This is your open-source container management.
Docker Enterprise Support	Refers to a commercial support for the platform.
Virtual machine	This refers to a program that virtualizes a complete PC and emulates dedicated hardware.
Namespace	This is simply an instrument for separating sets of repositories.
Image Layer	They are the building blocks of an image and they are linked in a parent-child connection.
Toolbox	This is simply the installer for Windows as well as Mac users.
Tag	Refers to a label that is applied to an image within a repository.
Swarm	This is a tool that pools a number of Docker hosts together and then it discloses them as one virtual docker host.
libnetwork	Provides an instinctive Go implementation for developing as well as handle network namespaces.

libcontainer	Offers an instinctive Go implementation for developing containers with filesystem access controls, capabilities, groups and namespaces.
Base Image	Refers to an image that does not have a parent layer.
Registry Server	This server is used to keep docker repositories.
Container Host	This is where an image ends up to when it is extracted from a registry server.
Graph Driver	This refers to a section of the application that maps the various image layers in the image set to the local storage.
Compose	Refers to a tool for describing as well as run multifaceted applications.
Container	Refers to a runtime instance, which is made up of a standard set of guidelines, execution environment and a docker image.
Data volume	This refers to a particular directory inside one/more containers that sidesteps the Union File System.
Build	This refers to the creation of docker images by making use of a dockerfile.
Btrfs	Also known as B-tree file system is a Linux filesystem that is accepted by Docker as storage backend.

Boot2docker	This is a specified lightweight Linux distribution for running docker containers.
aufs	This is an advanced multi-layered unification filesystem that is a Linux-based supported as a storage backend by Docker.

Docker is an open platform, through which you can build; ship as well as run distributed applications. Docker does the following: Docker Engine, Docker Hub, Docker Enterprise Support and Docker Services & Training. Docker Services & Training provides you with professional services along with the training you need to be an expert. Docker Enterprise Support is just a commercial support for the platform. Docker Hub is your online home/hub through which you can manage your Docker containers, and Docker Engine is your open-source container management.

Other things that Docker offers and does for you are: Docker Compose, Docker for Windows, and Docker for Mac, Docker for Windows, CS Docker Engine, Docker Trusted Engine, Universal Control Plane, Docker Cloud, Docker Store, Docker Machine, Docker Hub, and Docker Toolbox. Docker for Mac uses OS X sandbox security that provides your Mac with the necessary Docker tools. Docker for Windows is a native Windows application that provides your Windows PC with all Docker tools. Docker for Linux is installed on a PC with a preinstalled Linux distribution.

Docker Cloud is a hosted service for developing, testing as well as distributing Docker images to your hosts. Universal Control Plane allows you to manage a collection of on-premises Docker hosts. Docker Machine is an automate container purveying in the Cloud or on your network. Docker Machine is usually available for Linux, OS X and Windows. Docker Compose simply defines applications that are developed from multiple containers.

Why Docker?

Docker does bundle your application into a uniform element for software development. What this platform does is bind a piece of application in a complete filesystem that comprises of system libraries, system tools, runtime and code along with anything that you can install on a server.

With this, it is failsafe that the application will run in the same manner, irrespective of the environment it is in, always. Docker containers are open-based standards, making it possible for containers to run on Windows and Linux distribution.

Docker is lightweight as far as its operation is concerned. This is mainly so because containers share the same OS kernel and run on the same machine. These containers start instantaneously and they make use of less random access memory.

Disk usage as well as image downloads is efficient because images share common files and they are developed from layered filesystems. Docker is secure by default, unlike any other platform; containers offer an added protection layer because of how they isolate applications from each other.

When compared to virtual machines, containers are much better. Virtual machines are made up of: an entire guest OS, libraries, necessary binaries and the application, amounting to 10+ GBs. On the other hand, containers are made up of all of its dependencies along with the application. However, containers run as isolated processes on the host OS and they share the kernel. A feature unique to containers is the fact that they can run in any cloud, on any infrastructure, and on any PC.

Docker does it all for you by helping you build better software. This is what you need to empower creativity, because as a developer, you will be free of any constraints thanks to the isolation nature of Docker containers.

With Docker, you no longer have to waste precious time setting up developers' new instances, spinning up, and environments. All you have to do is obtain your live environment copies and then run them on a desired new endpoint that does run a Docker engine.

It is only Docker that does eradicate any environmental discrepancies. This is achievable because wrapping an application in a Docker container along with its dependencies and configurations ensures that the specified application will be operational in any environment.

Therefore, you do not have to install similar configurations into diverse environments. Docker allows you to share as well as collaborate by creating a collective framework for sysadmins and developers to work as a unit on distributed applications.

You can distribute and share content by storing, distributing and managing Docker images with your team in Docker Hub. The best part is that history, changes, and image updates are shared automatically across the organization.

You can easily share your applications with your associates by shipping them without environment dependencies developing any issues. Other teams do not have to know how

your app operates; all they have to do is test against or link to your application.

Docker containers ship application faster and at a convenient scale by allowing you to enthusiastically change your app from changing problem areas, scaling services to adding new capabilities. Docker ship seven times more applications after distributing containers in their environment. In addition, to frequent application updates, Docker offers added value to the clientele-base. Docker containers do quick scale by spinning up as well as down in seconds. This makes it easier for you to scale software services to gratify peak client demand.

It also reduces operating containers when demand gradually declines. Docker does easily amend issues by straightforwardly identify problems. Docker rapidly roll back to initiate the needed modifications and then drive the updated container into construction. The isolation that exists between containers allows the modifications less troublesome when compared to traditional application models.

Chapter 2 – Getting Started with Docker

In order to go ahead with Docker, you need to know how to download as well as install the various types of Docker. You can download and install Docker for Windows, Docker for Mac or Docker for Linux.

Docker for Windows

There are two channels you can use to download Docker for Windows: Stable channel and Beta channel. The stable channel is a fully developed and tested and it is fitted equipped with the contemporary form of Docker Engine. As far as a reliable and workable platform is concerned, this is considered to be the best. It is synched with Docker engine hotfixes and releases.

On the other hand, the Beta channel provides you with state-of-the-art features along with untried version of Docker Engine. It is regarded as the best channel to work with if you are much interested with experimental features. It is simply a continuance of the beta program, which allows you to leave feedback as the application grows. The releases for this channel take place more often that for the stable channel, which may be more than one every month.

You can install any of the two; all you have to do is click on the *Get Docker for Windows (stable)* or *Get Docker for Windows (beta)* from the following link: https://docs.docker.com/docker-for-windows/. However, Docker for Windows needs your PC to be running Windows 10 Pro, Education and Enterprise along with Microsoft Hyper-V. Docker installation requires readme first for Docker machine and toolbox users. Once Hyper-V is activated, Virtual Box seize to work.

Nevertheless, any particular virtual box VM images will be maintained, but virtual box VMs developed with Docker Machine will not start and they cannot be used in conjunction with Docker for Windows. Yet you can manage remote virtual VMs by using Docker Machine.

Through the *Settings* menu found in the System Tray, you can conveniently import a default VM after finishing the Docker for Windows installation. Currently, Docker platform for Windows only supports Windows 10 Pro, Education and Enterprise. However, the developers of Docker are promising to support additional versions of Windows in future.

Docker containers as well as images developed with Docker for Windows are normally shared among all user accounts on PC in which Docker is installed. This is mainly so since all Windows accounts utilize the same VM to create as well as run containers. Docker developers promise that Windows will improve its isolation of user account in future. With that said, the most important thing that you must do in order for Windows to work is enable the Hyper-V. Usually, Docker may enable Hyper-V by itself if necessary.

How to Install & Explore Docker for Windows for the First Time

1. Install Docker for Windows

Click on the stable or beta versions of Docker for Windows in order to access the *InstallDocker.msi* box. Then double click on the *InstallDocker.msi* box so as you can run the installer. Make sure you accept the license, then authorize the installer and go through with the installation by following the installation wizard. Docker will require you to use your system password to authorize Docker.app. In order to be able to manage Hyper-V VMs, install links to the Docker apps and install networking components, you will to use privileged access. Complete the installation by clicking *Finish*.

2. Launch Docker

Normally, Docker will start automatically the moment the installation is complete. You can check if Docker is running and if it is accessible from a terminal by observing the whale situated in the status bar. You will receive a pop message that indicates the installation is a success along with recommended next steps as well as a link to the following page, "*Docker is now up and running!*" you should verify if you have installed the latest version of Docker by selecting *About Docker*.

3. Verify the Installation

You can check your Docker as well as Docker-compose and then verify your installation by running PowerShell or any other favorite shell.

4. Explore Docker

Once you have completed and verified the installation, you should go ahead and explore Docker for the first time. Make certain that Docker commands are functioning by checking version info. Go ahead and open a shell, such as PowerShell. When the shell opens, try out some Docker commands, including *docker info, docker version or docker ps*. You will receive the desired outcome; you can thereafter run as many docker commands as possible.

Docker for Mac

This is the latest provision for Mac by Docker developers. The most amazing thing about it is that it runs as an intrinsic Mac app and it utilizes xhyve to emulate the Docker Engine atmosphere as well as Linux features for the app's daemon. You can get Docker for Mac by clicking on the blue bar from this page https://docs.docker.com/engine/getstarted/step_one/. The link will provide you with installer.

There are minimal requirements needed for Docker for Mac, and they are as follows. Your Mac should be a newer model or 2010 and it should have Intel hardware support for MMU virtualization, such as extended page tables.

The PC should have at least 4 GB of random access memory and OS X 10.10.3 Yosemite or newfangled version. Usually, Docker for Mac will automatically produce an error if you make attempts of installing VirtualBox before 4.3.30. If the error is displayed, you should install the existing VirtualBox and then retry the installation.

Docker for Linux

Docker Engine does run indigenously on Linux distributions. There are different Docker for Linux distributions, including: Ubuntu, openSUSE & SUSE Linux Enterprise, Red Hat Enterprise Linux, Oracle Linux, Gentoo, FrugalWare, Fedora, Debian, CRUX Linux, CentOS, and Arch Linux. Each of these distributions has its own instructions for installing Docker. You can find the various Linux distributions and the appropriate links from the following URL https://docs.docker.com/engine/installation/.

Docker for Ubuntu

Docker is supported on the following Ubuntu OS: Ubuntu Wily 15.10, Ubuntu Precise 12.04 [LTS], Ubuntu Xenial 16.04 [LTS], and Ubuntu Trusty 14.04 [LTS]. You are required to use installation mechanism and Docker-managed release packages. The aforementioned packages guarantee that you will get the newest release of docker. You should consult Ubuntu documentation, if you want to install Docker via Ubuntu –managed packages.

There are minimal prerequisites for installing Docker for Ubuntu. To begin with, Docker installation needs 64-bit installation irrespective of the Ubuntu version you are using. The second most important consideration is the kernel, which can be 3.10 minor or newer. You can check the version of your Ubuntu kernel by opening a terminal and then display the kernel version by using *uname –r*. You can update all APT sources to the latest Docker repository if you beforehand installed the platform using APT.

The APT repository compromises of Docker 1.7.1 and newer. There are a number of steps that you have to follow in order to set up APT to make use of packages from the newfangled repository. The first step requires you to log into your PC as a user with *root* privileges or *sudo*. The second step requires you to access a terminal window and then open it. The third step requires you to update package info, but you must make certain that CA certificates are properly installed and APT supports https.

You can access further details on how to install Docker on Ubuntu among other information from the following link:

https://docs.docker.com/engine/installation/linux/ubuntulinux/.

Chapter 3 – Exploring Docker & Learning the Basic Commands

Test Your First Image (Docker for Linux)

You can test if you have successfully installed Docker by running the following command *$ docker run debian echo "Hello World"*. This can consume a considerable amount of time with respect to your internet speed. You should be presented with the details presented in figure 1. The result is an indication that the run command has been called; a command that launches containers. Debian refers to the image you wish to use.

```
Unable to find image 'debian' locally
debian:latest: The image you are pulling has been verified
511136ea3c5a: Pull complete
638fd9704285: Pull complete
61f7f4f722fb: Pull complete
Status: Downloaded newer image for debian:latest
Hello World
```

Figure 1: Test Results

The very first line of the result indicates that the local copy of the image is not there. What Docker will do to correct this fault is use the Docker Hub to check and then download the latest version of the image. The image will be converted into a running container the moment the download is complete, and then the specified command is executed. The outcome of running the command is exhibited in very last line of the results. Running the command for the second time will automatically start the container.

Running a similar command using VM would definitely take several minutes when compared to Docker. The best thing about Docker is that you can ask it to provide you with a shell inside a specific container. To do this, you will have to run the command displayed in figure 2. This will provide you with a command prompt that exists inside the container.

```
$ docker run -i -t debian /bin/bash
root@622ac5689680:/# echo "Hello from Container-land!"
Hello from Container-land!
root@622ac5689680:/# exit
exit
```

Figure 2: Test Results

Basic Docker Commands

There are two programs that you must be aware of as far as Docker is concerned. These programs are: docker daemon (a server process that does manage containers) and docker client (which is a remote control for the first program). To list docker client commands, you will have to key in *docker* without arguments. You can look for images as well as utilize images that have been developed by other users.

Supported Environmental Variables

The following list contains the environmental variables that are supported by Docker.

- DOCKER_API_VERSION, which is simply the API version you should use, such as 1.19.
- DOCKER_TMPDIR refers to a site for provisional Docker files.
- DOCKER_CONFIG, this is the site of client configuration files
- DOCKER_CONTENT_TRUST_SERVER, this is the notary server URL for you to use.
- DOCKER_CERT_PATH, this is where you will find the authentication keys.
- DOCKER_CONTENT_TRUST, this refers to a situation where notary is used to sign as well as verify images by set Docker.
- DOCKER_DRIVER, this is the graph driver that you are supposed to use.
- DOCKER_TLS_VERIFY, this is when set docker verifies the remote and makes use of TLS.
- DOCKER_HOST, this refers to the daemon socket you can connect to.
- DOCKER_RAMDISK, it normally disables *pivot_ root* when set.
- DOCKER_NOWARN_KERNEL_VERSION, this does usually prevent any warnings that indicate that the Linux kernel you are using is unfitting.

Configuration Files

Normally, the docker command line keeps its configuration files in *.docker* directory, which is inside the *$HOME directory*. You can use the DOCKER_CONFIG variable to set a different site. You should know that Docker is responsible for a great number of files in the configuration directory. You are not supposed to changes the files, but you can change the *config.json* file to manage specific features with respect to how *docker* command acts.

You can use command-line options or environmental variables to change how the *docker* command behaves. You can change similar behavior by using options inside *config.json*, but you must observe the precedence order of mechanisms. Interestingly, command line options rule against environment variables while environmental variables rule against properties that have been specified in a *config.json* file. Usually, this file keeps a JSON encoding of different properties.

You should be aware of the *HttpHeaders* property, which stipulates a group of headers that can be included in all messages sent from the platform. What docker does is place the headers into the messages without concerning itself with interpretation or understanding of the headers. As far as the headers are concerned, docker prevents the headers from modifying headers. The *docker ps* output format is identified by property *psFormat*.

Docker Command	**Function/ Explanation**
docker pull image_name:tag	It downloads the image as well as the image's parents required to come up with containers with the designated image.
docker create —name container_name image_name:tag	This particular command is useful for detecting the container from the image.
docker build image_name.	Allows you to develop a dockerfile inside the directory
docker-machine ip default	This command helps you determine your docker-machine IP.
docker pull image_name	Allows you to download along with all the image's parents
docker start container_name	Allows you to start a container

docker stop container_name	Allows you to stop a container
docker run -ti —name container_name image_name /command	Allows you to run a shell command within a newly developed as well as launched container.
docker run —rm -ti image_name /command	Allows you to run a command within a container and it also allows you to eradicate the container when the command is through.
docker exec -ti container_name "cmd"	Allows you to run a shell within the container.
docker search [keywords]	Allows you to search for images
docker ps -1	Helps you locate container ID.
docker commit [id] [new image name]	Enables you to save the container with a specified image name.
docker run learn/ping ping www.google.com	Allows you to ping command on your container.
docker ps **docker inspect [container id]**	Help you mug up about accessible containers.
docker help run	unearths a complete list of the available Docker run flags

Build Your First Container

To build a container, you will have to run the *docker run* command, which offers Docker launch capabilities. You will be using this command multiple times to develop new containers. We have tried to state and explain some of the basic Docker commands in the above table; however, you can always get more commands by typing *docker help*. When you run the command, Docker will create your first container similar to the one displayed in the image below.

Figure 3: Creating a Container

According to the results displayed in figure 3, you started by instructing Docker to execute *docker run* command. You then passed the run command a couple of line flags, such as i and t. Each of the flags that you have passed has specific functions to perform. The i flag ensures STDIN remains open from the created container and this is regardless of whether or not you are attached. On the other hand, the t flag instructs Docker to allocate a pseudo-tty to a container that you are about to create.

The allocation of pseudo-tty offers you an interactive shell in the newly created container. Instead of running as a daemon service in order to interact on the command line, you use the t flag which makes things easier. You can unearth a complete list of the available Docker run flags by keyboarding *docker help run*.

Chapter 4 – Docker Hub

Docker Hub is your online home/hub through which you can manage your Docker containers. Docker Hub operates like GitHub for images and it is a public registry on which you can host images both private and public. You can share as well as collaborate with other Docker users. You can easily create a free Docker Hub account from the Docker Hub login page.

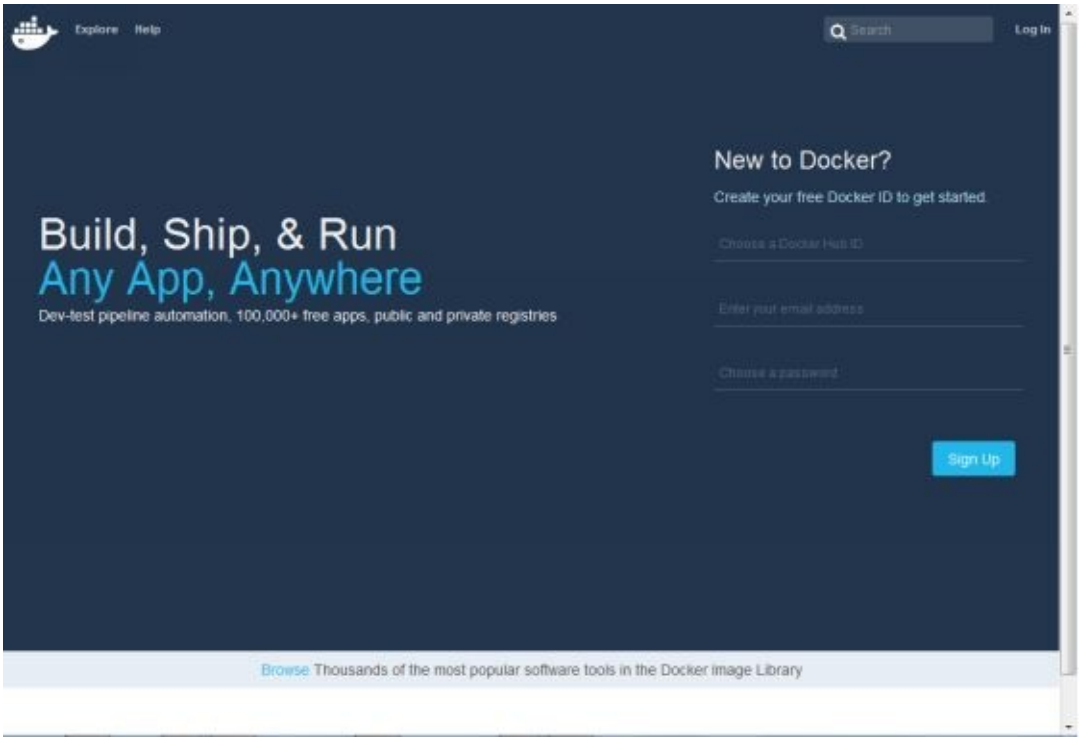


Figure 4: Docker Hub

To create a free account, you will need to choose a Docker Hub ID, provide your email address and choose an appropriate password. Once you sign up, you will receive a confirmation email, which you are supposed to click to confirm your registration. Once you have confirmed your registration, you will have to enter your Docker Hub ID and password to access your account.

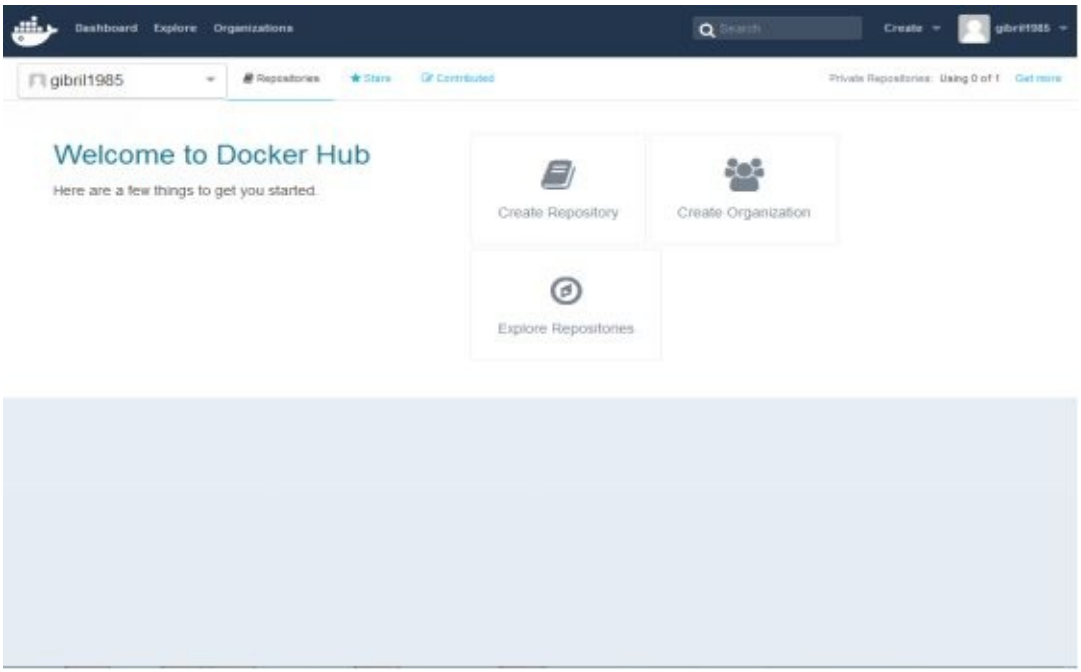


Figure 5: Docker Hub

Once you sign in to Docker Hub, you will be presented with your account as illustrated in figure 2. The Page is simple to maneuver – the top bar with the whale sign on the left top end presents you with Dashboard, Explore and Organizations icons. On the right side of the same bar you will find the dropdown create menu for creating repository, organization and automated build.

Create Repository

To create a repository, you will have to click on the *Create Repository* link. When you are in the *Create Repository* page, you will be prompted to do the following: choose a namespace, which is mandatory; add a repository name, which is mandatory; add a short description, with respect to what you are creating; add markdown; and set your repository to be either public or private. You will observe that there are very minimal requirements that you will have to provide to create repository.

Once you have finished filling the required information, you should go ahead and click on the create button. Your repository will be created and you will be presented with your first repository. The repository status bar presents you with the following button: Repo Info, Tags, Collaborators, Webhooks, & Settings. The Repo Info button provides you with the repository information. The collaborators button provides you with a section of adding collaborators.

When you click the Webhooks button, you will be presented with your workflows. When you publish an image to the repo, you should expect your workflows to kick off with respect to your itemized Webhooks. To create a Webhooks, you will have to provide a Webhook name and Webhook URL. Then you will have to set the Visibility under Settings. A private repository is only accessible by you or a member of your organization. While a public repository is available to all members. You are also presented with a *Delete* button for deleting repositories, this action will not only delete the repository, but also abolish all images and the process is non-reversible.

Create Automated Build and Organizations

To create automated build, you first have to link account to Github or Bitbucket. You can achieve this by clicking the *Link Accounts* button. Linked accounts are operated with automated builds, which allow Docker to gain right of entry to your project lists as well as assist you set up your automated builds. However, you should know that you can connect a Github or Bitbucket account to one Docker hub account.

When you connect your account to Github, Docker let you decide how much access they can have to your linked account. There are two options that you can choose: public and private choice, which is recommended by Docker and limited access.

Public and private choice is defined by the following: read & write access to private as well as public repositories; the choice is convenient if you want to configure an automated build from a private Github; it is expedient when using a private Github organization; and Docker Hub will spontaneously set up the hooks as well as deploy keys on your behalf.

The limited access option is defined by the following. It is based on public read only access. It does only work with public organizations and repositories. To be able to use automated build, you will have to manually make modifications to the repositories.

Once you have made up your mind, you should go ahead and click on the *Select* button beneath each option. You can create organizations and team by clicking on the *Create Organizations* button.

Docker Hub Account Settings

You can make necessary adjustments in the account settings page. The top section of the Account Settings page presents you with your default repository visibility, which can be public or private depending with the visibility settings. The second item on the page is your email address, which is utilized for all correspondence and notifications from Docker. You can change the primary email by first adding a new email and then select that new email as primary.

You can change your password from this page under the *Change Password* section. You can also provide your account information, and click on the save button; the information will be visible to all users. You can also transform your account to an organization, which will allow you to use organization features. All you have to do is click on the *Convert to Organization* link button.

There are some few things that you must know before you convert your account to an organization. To begin with, your account will be converted into an organization account and all administrative responsibilities are left to group of users or another user. Therefore, you will not be able to access the account thereafter. The email or email accounts that are linked to your user account will be freed, and you can use them for another account.

When you transform your account to an organization, all your linked accounts are removed, such as Bitbucket or GitHub. This allows you to connect your external accounts to a different Docker Hub user.

You should know that billing details along with private repository plans will continue to be attached to your account even after your account is transformed to an organization. The repositories names as well as namespaces will continue to be the same.

All collaborators that are set up for repositories will be eradicated then they must be rearranged using group collaborators. The automated builds for your account are brought up to date to look as if they were initially set up by the original organization holder. However, you are a great risk of losing your account for good once the conversion is complete. To make the necessary transition, you will be required to provide your Docker ID. Once you are done reading the warning, you can go ahead and click on the *Save and Continue* button.

Conclusion

Docker is an open-source platform that does automate the distribution of applications within containers. It does this by providing you with an additional layer of abstraction and automation of OS. This publication has provided you with the most important information about Docker.

This includes an in-depth definition of Docker, how to install Docker in different operating systems, and a well-crafted introduction into Docker containers. You will observe that Docker is the easiest platform to work with once you are through reading each and every piece of information provided.