



DOCKER

☰ Tags	Tools
👤 Created by	
📅 Date	@September 28, 2022

INTRODUCTION

What is Docker?

- Docker is an open platform for developing, shipping, and running applications.
- Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
- By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

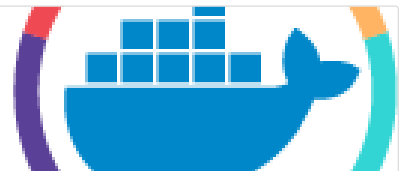
Where will Docker be useful?

- **For Fast, consistent delivery of your applications.**
- **Responsive deployment and scaling**
 - Docker containers can run on a developer's local laptop, on physical or virtual machines in a data center, on cloud providers, or in a mixture of environments.
 - Docker's portability and lightweight nature also make it easy to dynamically manage workloads.
- **Running more workloads on the same hardware**
 - It provides a viable, cost-effective alternative to hypervisor-based virtual machines, so you can use more of your compute capacity to achieve your business goals.
 - Docker is perfect for high-density environments and for small and medium deployments where you need to do more with fewer resources.

Docker overview

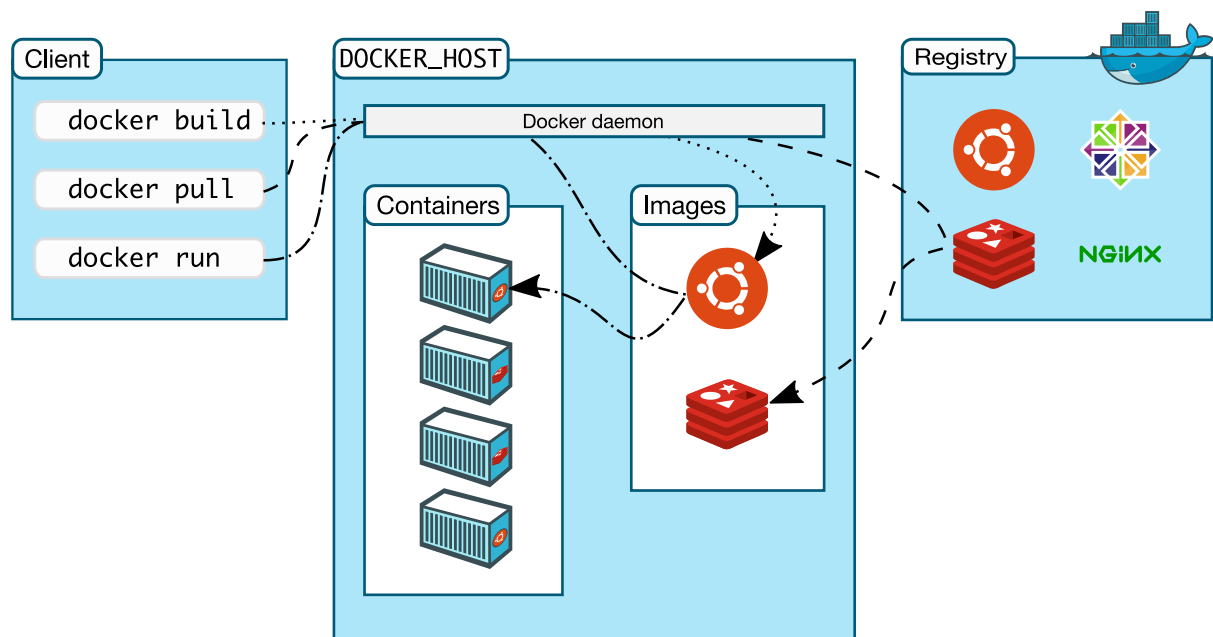
Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications.

 <https://docs.docker.com/get-started/overview/>



Docker Architecture

- This is the Docker Architecture which shows how individual components communicate with each other.
- The Docker *client* talks to the **Docker daemon**, which does the heavy lifting of building, running, and distributing your Docker containers.
- The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon (cloud).
- The Docker client and daemon communicate using a REST API.
- Another Docker client is **Docker Compose**, which lets you work with applications consisting of a set of containers (multiple containers).
- Now, let's have a look at some of the individual components that we have defined above.



The Docker daemon

The Docker daemon (`dockerd`) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

- The Docker client (`docker`) is the primary way that many Docker users interact with Docker.
- When you use commands such as `docker run` , the client sends these commands to `dockerd` , which carries them out.
- The Docker client can communicate with more than one daemon.

Docker Desktop

- Docker Desktop is an easy-to-install application for your Mac, Windows or Linux environment that enables you to build and share containerized applications and microservices.

Docker registries

- A Docker *registry* stores Docker images.
- **Docker Hub** is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub *by default*.
- There are different cloud platforms where we can store the docker image. For example, in Amazon Web Services we have [Amazon Elastic Container Registry](#), in Microsoft Azure, we have [Azure Container Registry](#), etc.

Docker Images

- A Docker image is a file used to execute code in a Docker container
- Docker images act as a set of instructions to build a Docker container, like a template. Docker images also act as the starting point when using Docker.
- Often, an image is *based on* another image, with some additional customization.
- To build your own image, you create a *Dockerfile* with a simple syntax for defining the steps needed to create the image and run it.

Simple Flask application example -:

```
FROM python:3.8-slim-buster
WORKDIR /app
COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt
```

```
COPY . .  
CMD [ "python3", "-m", "flask", "run", "--host=0.0.0.0"]
```

Docker Container

- A container is a runnable instance of an image.
- You can create, start, stop, move, or delete a container using the Docker API or CLI.
- You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.
- A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.

Docker Installation

Windows

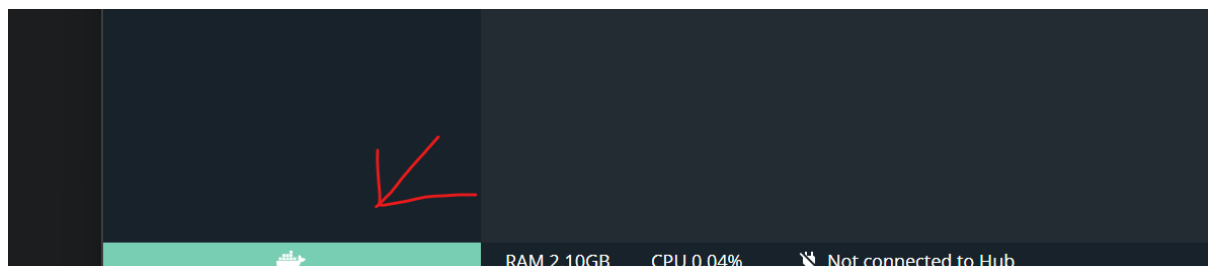
- This is the official blog to install Docker on windows.
- The first step is to enable **WSL (Windows Subsystem for Linux)**. Follow [this](#) video.

How to Install Ubuntu in Windows 10 with WSL2-Windows Subsystem for Linux

 <https://youtu.be/fWq6ZLRqTZc>



- The second step is to install **Docker Desktop**.
- Your Installation will be completed once you open Docker Desktop.
- Check if you can see this green symbol on bottom left.



Ubuntu

- To install on Ubuntu just follow this official documentation. [LINK](#)

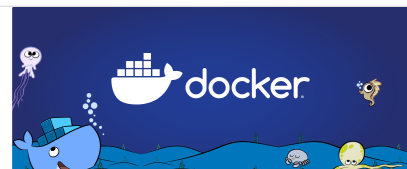
macOS

- To install on macOS just follow this official documentation. [LINK](#)

Docker Desktop - Docker

Install Docker Desktop - the fastest way to containerize applications. The Docker Subscription Service Agreement has been updated. Our Docker Subscription Service Agreement includes a change to the terms for Docker Desktop. It remains free for small businesses (fewer than 250 employees AND less than \$10

 <https://www.docker.com/products/docker-desktop/>

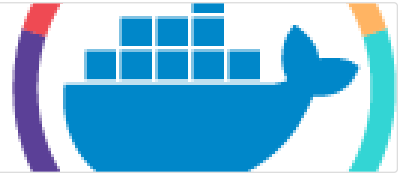


Install Docker Desktop on Windows

Install Docker Desktop on Windows

Estimated reading time: 9 minutes Docker Desktop terms Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than \$10 million USD in annual revenue) requires a paid subscription. Welcome to Docker Desktop for Windows. This page contains information about Docker

<https://docs.docker.com/desktop/install/windows-install/>



Manual installation steps for older versions of WSL

For simplicity, we generally recommend using the wsl --install to install Windows Subsystem for Linux, but if you're running an older build of Windows, that may not be supported. We have included the manual installation steps below. If you run into an issue during the install process, check the installation section of

<https://learn.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package>




Docker Cheat Sheet

The Ultimate Docker Cheat Sheet

Create an image from a Dockerfile. Run a command in an image. Example Create a container from an image. docker exec [options] CONTAINER COMMAND -d, --detach # run in background -i, --interactive # stdin -t, --tty # interactive Example Run commands in a container.

<https://dockerlabs.collabnix.com/docker/cheatsheet/>

Docker Cheat Sheet



Build

Build an image from the Dockerfile in the current directory and tag the image
`docker build -t myimage:1.0 .`

List all images that are locally stored with the Docker Engine
`docker image ls`

Delete an image from the local image store
`docker image rm alpine:3.4`

Share

Pull an image from a registry
`docker pull myimage:1.0`

Retag a local image with a new image name and tag
`docker tag myimage:1.0 myrepo/myimage:2.0`

Push an image to a registry
`docker push myrepo/myimage:2.0`

Run

Run a container from the Alpine version 3.9 image, name the running container "web" and expose port 5000 externally, mapped to port 80 inside the container.
`docker container run --name web -p 5000:80 alpine:3.9`

Stop a running container through SIGTERM
`docker container stop web`


Stop a running container through SIGKILL
`docker container kill web`

List the networks
`docker network ls`

List the running containers (add --all to include stopped containers)
`docker container ls`

Delete all running and stopped containers
`docker container rm -f $(docker ps -aq)`

Print the last 100 lines of a container's logs
`docker container logs --tail 100 web`



Docker Management

All commands below are called as options to the base **docker** command. Run `docker <command> --help` for more information on a particular command.

app*	Docker Application
assemble*	Framework-aware builds (Docker Enterprise)
builder	Manage builds
cluster	Manage Docker clusters (Docker Enterprise)
config	Manage Docker configs
context	Manage contexts
engine	Manage the docker Engine
image	Manage images
network	Manage networks
node	Manage Swarm nodes
plugin	Manage plugins
registry*	Manage Docker registries
secret	Manage Docker secrets
service	Manage services
stack	Manage Docker stacks
swarm	Manage swarm
system	Manage Docker
template*	Quickly scaffold services (Docker Enterprise)
trust	Manage trust on Docker images
volume	Manage volumes

*Experimental in Docker Enterprise 3.0.

```
docker pull nginx #Pull Nginx
docker run --name docker-nginx -p 80:80 nginx #Expose Nginx 80 Port
docker run --name docker-nginx -p 8080:80 -d nginx #Expose 8080
docker run -P nginx
docker run -d -P nginx
```

```
docker build -t friendlyname . # Create image using this directory's Dockerfile
docker run -p 4000:80 friendlyname # Run "friendlyname" mapping port 4000 to 80
docker run -d -p 4000:80 friendlyname # Same thing, but in detached mode
docker run --name test-ubuntu -it ubuntu:16.04 ./bin/bash
docker exec -it [container-id] bash # Enter a running container
docker ps # See a list of all running containers
```

```

docker stop <hash>                # Gracefully stop the specified container
docker ps -a                      # See a list of all containers, even the ones not running
docker kill <hash>                # Force shutdown of the specified container
docker rm <hash>                  # Remove the specified container from this machine
docker rm $(docker ps -a -q)      # Remove all containers from this machine
docker images -a                  # Show all images on this machine
docker rmi <imagename>            # Remove the specified image from this machine
docker rmi $(docker images -q)    # Remove all images from this machine
docker logs <container-id> -f     # Live tail a container's logs
docker login                      # Log in this CLI session using your Docker credentials
docker tag <image> username/repository:tag # Tag <image> for upload to registry
docker push username/repository:tag # Upload tagged image to registry
docker run username/repository:tag # Run image from a registry
docker system prune               # Remove all unused containers, networks, images (both dangling and unreferenced), and opt
docker system prune -a            # Remove all unused containers, networks, images not just dangling ones (Docker 17.06.1-ce
docker volume prune               # Remove all unused local volumes
docker network prune              # Remove all unused networks

cd usr/share/nginx/html/

docker volume create my_vol        # Create a volume
docker volume ls                   #
docker volume inspect my_vol       # troubleshooting
docker volume rm my_vol

##Setup Docker in EC2
Allows access to port 80 (HTTP) from anywhere
HTTP TCP 80 Anywhere
sudo yum update -y
sudo yum install -y docker
sudo service docker start
sudo usermod -aG docker ec2-user

```

Move Docker to Different Directory

```

# Step 1 : Stop Docker
# Commands
wsl --shutdown
wsl --export docker-desktop-data E:\docker-desktop\docker-desktop-data.tar
wsl --unregister docker-desktop-data
wsl --import docker-desktop-data E:\docker-desktop\data E:\docker-desktop\docker-desktop-data.tar --version 2

```

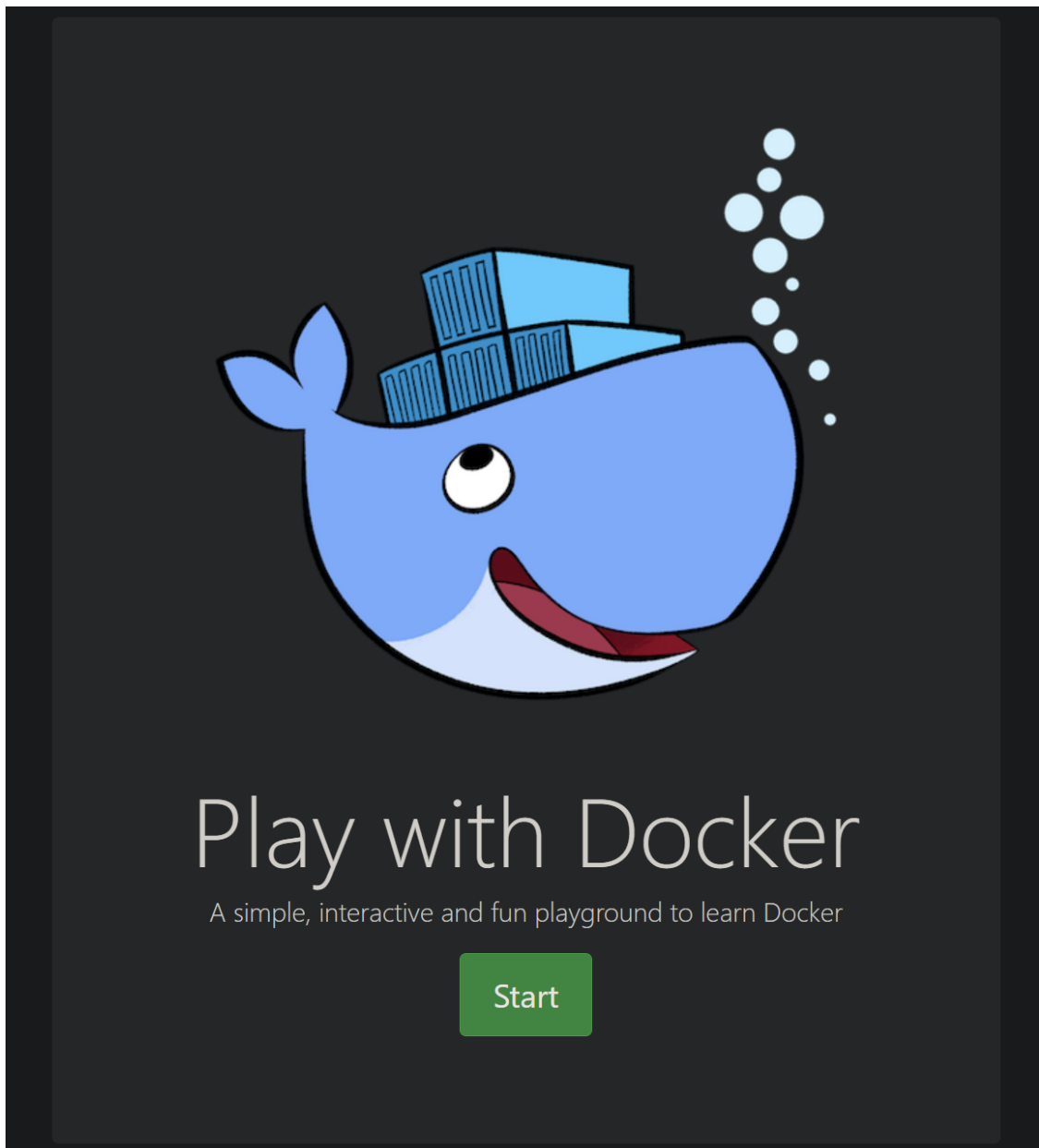
Create Docker Compose

Docker Compose is **a tool that was developed to help define and share multi-container applications**. With Compose, we can create a YAML file to define the services and with a single command, can spin everything up or tear it all down.

```
docker compose COMMAND
```

Docker Playground

- You can access the docker playground using this [LINK](#).
- Docker playground which **allows users to run Docker commands in a matter of seconds** . It gives the experience of having a free Alpine Linux Virtual Machine in browser, where you can build and run Docker containers and even create clusters in Docker Swarm Mode.



- Basic docker playground implementation.

```

[node1] (local) root@192.168.0.28 ~
$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:62af9efd515a25f84961b70f973a798d2eca956b1b2b026d0a4a63a3b0b6a3f2
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
[node1] (local) root@192.168.0.28 ~
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
[node1] (local) root@192.168.0.28 ~
$ docker images
REPOSITORY    TAG        IMAGE ID        CREATED        SIZE
hello-world   latest    feb5d9fea6a5    12 months ago  13.3kB
[node1] (local) root@192.168.0.28 ~
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

```