



Install Docker on your local machine

Download the correct installer for your operating system and run the installation:

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

 **Docker Desktop for Mac**
A native application using the macOS sandbox security model that delivers all Docker tools to your Mac.

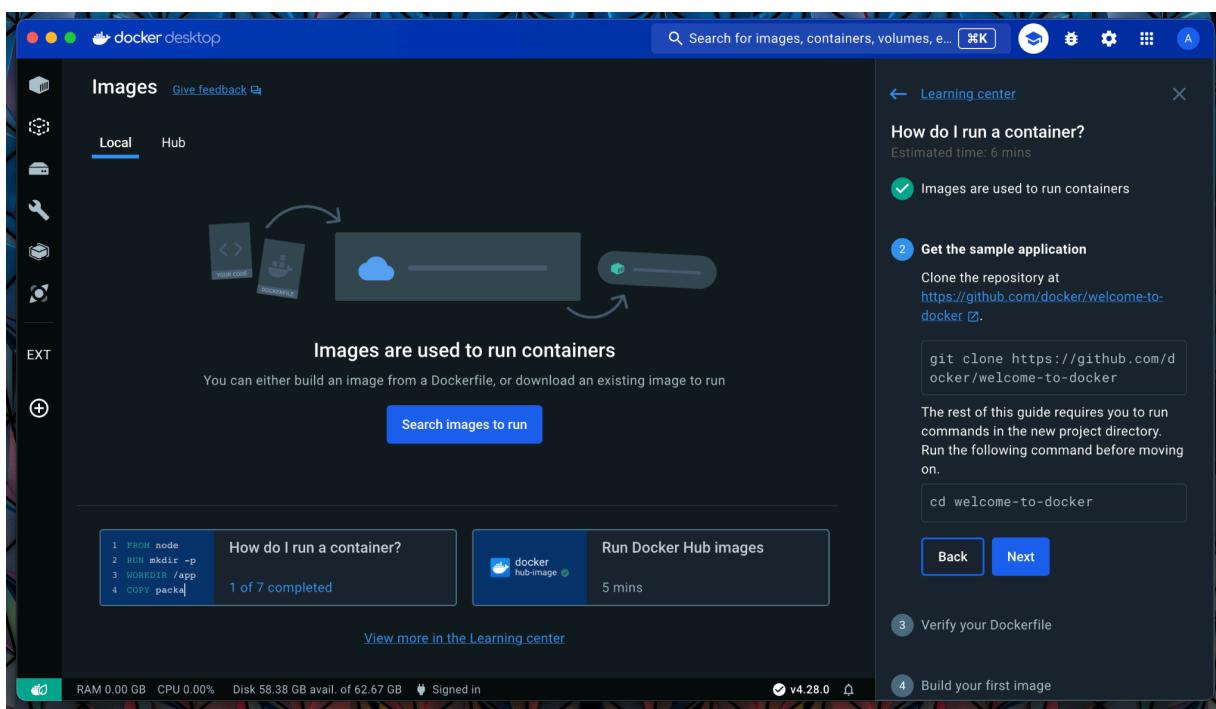
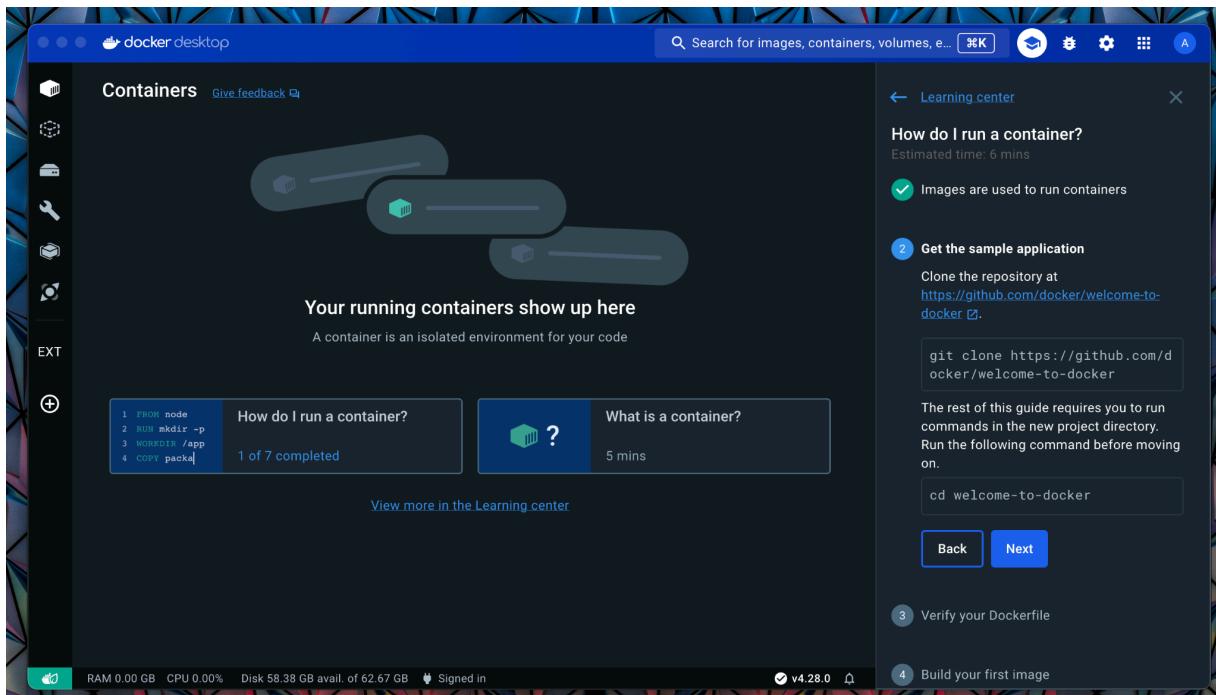
 **Docker Desktop for Windows**
A native Windows application that delivers all Docker tools to your Windows computer.

 **Docker Desktop for Linux**
A native Linux application that delivers all Docker tools to your Linux computer.

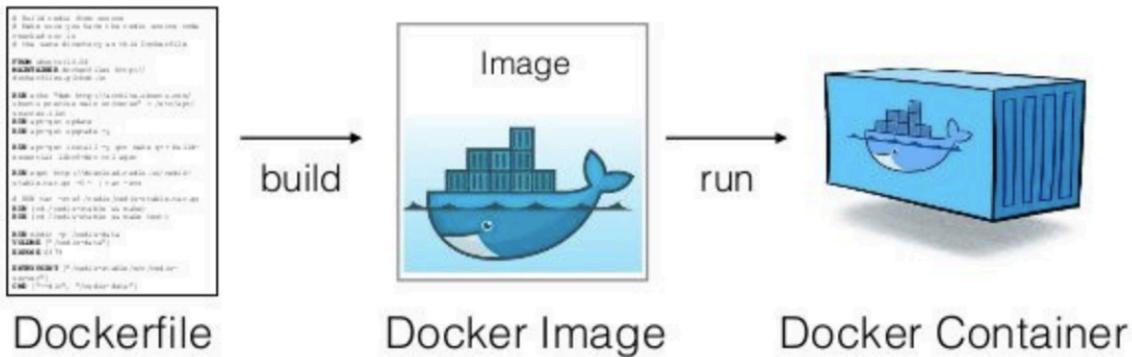
- Go to the official Docker website
- Follow the installation step

Always best to **refer to the latest installation guide** in the official documentation.

Open Docker Engine



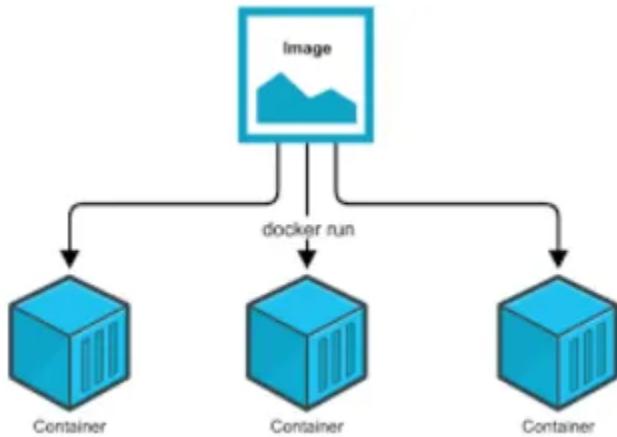
Docker Images vs Docker Containers



Docker images are like blueprints, containing all the instructions and ingredients needed to build and run software.

Docker containers are the actual running instances of these images, providing isolated environments where applications can execute. Images are static and reusable, while containers are dynamic and disposable, allowing for efficient deployment and scaling of applications.

You can **run multiple containers from 1 image**.



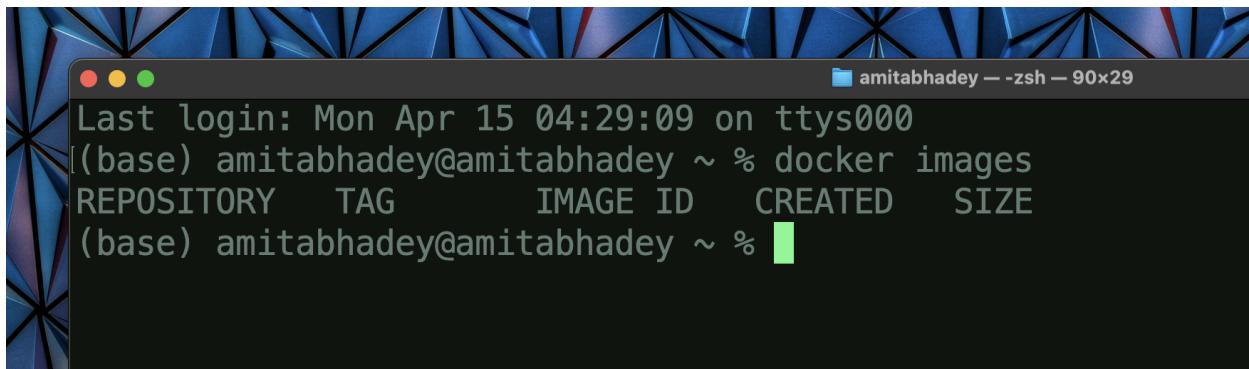
Docker Commands

List Docker Images

```
docker images
```

Lists all Docker images available locally on your system.

docker images will display a list of all images along with their tags and sizes.



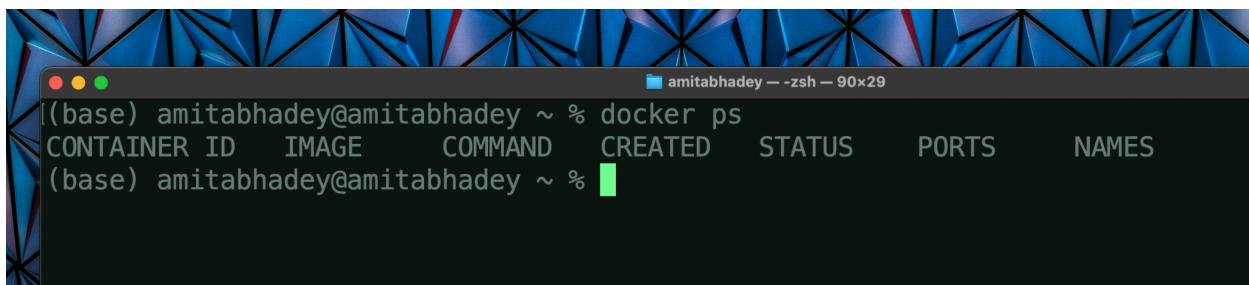
```
Last login: Mon Apr 15 04:29:09 on ttys000
(base) amitabhadey@amitabhadey ~ % docker images
REPOSITORY      TAG          IMAGE ID   CREATED     SIZE
(base) amitabhadey@amitabhadey ~ %
```

List Docker Containers

```
docker ps
```

Lists all running containers.

docker ps will display a list of active containers along with their IDs, names, and other details.



```
(base) amitabhadey@amitabhadey ~ % docker ps
CONTAINER ID   IMAGE      COMMAND   CREATED     STATUS      PORTS      NAMES
(base) amitabhadey@amitabhadey ~ %
```

Docker Registry

A Docker registry is a storage and distribution system for Docker images. It's a central repository where Docker users can store and share their Docker images.

Official images available from applications like Redis, Mongo, Postgres, etc.

Docker Hub, one of the biggest Docker Registry hosted by Docker:

<https://hub.docker.com/>

The screenshot shows the Docker Hub search interface with the query 'redis'. The results page displays three official Redis images:

- redis** (Official Image): Updated 2 days ago. Description: Redis is an open source key-value store that functions as a data structure server. Tags: Windows, Linux, unknown, 386, ARM, ARM 64, mips64le, PowerPC 64 LE, IBM Z, x86-64, unknown. Pulls: 41,444,932 (Mar 31 to Apr 6). Last week: 10K+. Status: Best Match.
- redislabs/redisinsight** (Official Image): By Redis, Updated 12 days ago. Description: RedisInsight - The GUI for Redis. Tags: Linux, x86-64, arm64. Pulls: 82,942 (Last week). Status: 10K+.
- redislabs/redisearch** (Official Image): By Redis, Updated 6 months ago. Description: RedisSearch - A full-text search engine for Redis. Tags: Linux, x86-64, arm64. Pulls: 219,008 (Last week). Status: 10M+.

The screenshot shows the official Redis Docker Hub page. Key elements include:

- redis** (Official Image): Docker Official Image · 1B+ · 10K+. Redis is an open source key-value store that functions as a data structure server.
- Copy** button for the command `docker pull redis`.
- Overview** tab selected.
- Quick reference** section with links to maintainers and help resources.
- Supported tags and respective Dockerfile links** section listing tags: 7.2.4, 7.2, 7, latest, 7.2.4-bookworm, 7.2-bookworm, 7-bookworm, bookworm, 7.2.4-alpine, 7.2-alpine, 7-alpine, alpine, 7.2.4-alpine3.19, 7.2-alpine3.19, 7-alpine3.19, alpine3.19.
- Recent Tags** section showing tags: 6.0.20-bookworm, 6.0.20, 6.0-bookworm, 6.0, latest, bookworm, 7.2.4-bookworm, 7.2.4, 7.2-bookworm, 7.2.
- About Official Images** section stating: Docker Official Images are a curated set of Docker open source and drop-in solution repositories.
- Why Official Images?** section stating: These images have clear documentation, promote best practices, and are designed for the most common use cases.

Terminology

- **Repository:** The repository is the collection of related Docker images. It serves as the primary namespace for organizing and managing images. Each repository can contain multiple versions or tags of the same image.
- **Tag:** Tags are used to differentiate between different versions or variants of an image within a repository. They provide a way to identify and reference specific image versions. Common tags include version numbers (e.g., "latest", "v1.0"), release names, or specific configurations.
- **Description:** The description provides a brief overview or summary of the Docker image. It may include details about the software application, its purpose, features, and usage instructions. A descriptive summary helps users understand the image's contents and capabilities.
- **Dockerfile:** Some Docker images on Docker Hub include a link to the Dockerfile used to build the image. The Dockerfile is a text file that contains instructions for building the image, including base image, dependencies, environment configuration, and application setup. Providing access to the Dockerfile allows users to understand how the image was created and customize it if necessary.
- **License:** The license attribute specifies the software license under which the Docker image is distributed. It ensures that users understand the terms and conditions governing the use, modification, and distribution of the image. Common license types include open-source licenses like MIT, Apache, GPL, and proprietary licenses.
- **Stars:** Stars represent the popularity or community interest in a Docker image. Users can star images they find useful or valuable, indicating their endorsement or preference. Higher star ratings often indicate widely used or well-regarded images within the Docker community.
- **Pulls:** Pulls indicate the number of times the Docker image has been downloaded or pulled from Docker Hub. It provides insights into the image's usage and popularity among users. Images with a high number of pulls are typically considered reliable and widely used.
- **Size:** The size attribute specifies the total disk space occupied by the Docker image. It helps users estimate the storage requirements for pulling and storing the image locally. Image size is an important consideration, especially for resource-constrained environments or when deploying images over the network.

Container Properties

- **Container ID:** A unique identifier assigned to the container upon creation. It is a hexadecimal string used to reference the container in Docker commands and operations.
- **Image:** Specifies the Docker image used to create the container. It defines the filesystem, environment, and runtime configuration of the container.
- **Command:** Specifies the command or process to be executed inside the container when it starts. It can be overridden during container creation or specified in the Dockerfile.
- **Ports:** Ports can be bound to the container to enable communication between the container and the host system or external network. Port mappings define how network traffic is routed to services running inside the container.

Pull an Image

```
docker pull IMAGE_NAME  
docker pull IMAGE_NAME:VERSION_NUMBER
```

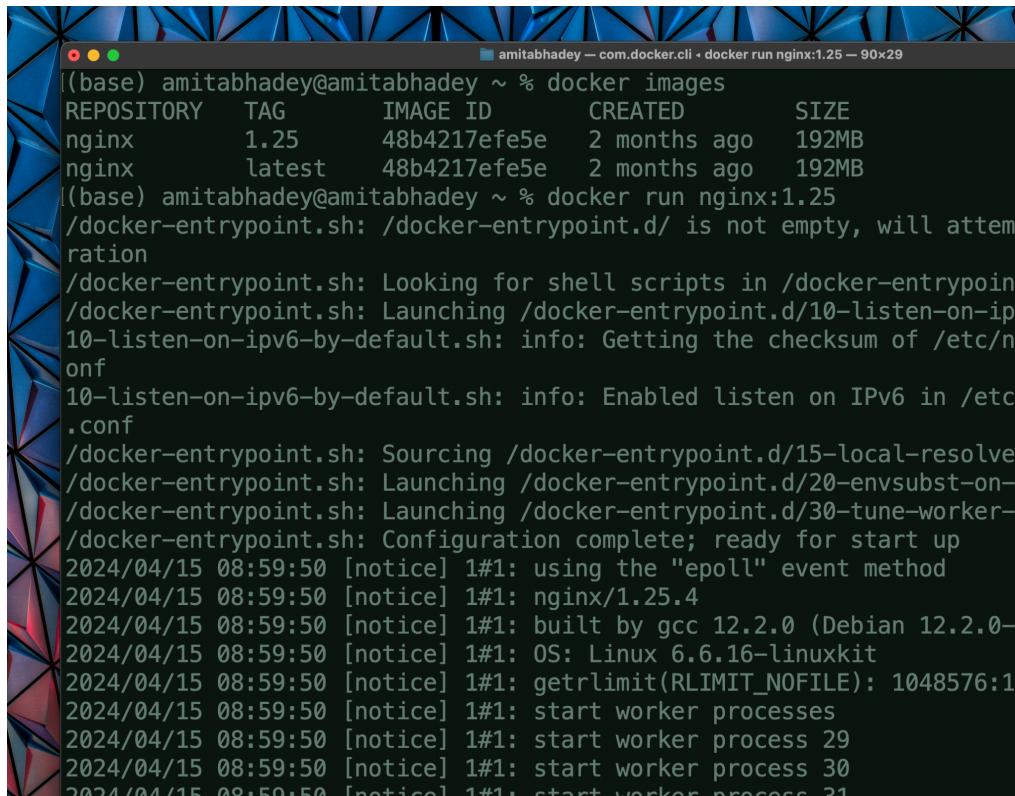
The screenshot shows the Docker Hub interface for the official nginx image. At the top, there's a search bar with 'nginx' and a 'Copy' button next to the search term. Below the search bar, the nginx logo is displayed along with the text 'Docker Official Image • 1B+ • 10K+'. A note below the logo says 'Official build of Nginx.' On the left, there are tabs for 'Overview' (which is selected) and 'Tags'. The 'Overview' section contains a 'Quick reference' box with information about maintainers and help resources, and a 'Supported tags and respective Dockerfile links' box listing various tag names. To the right, there's a 'Recent Tags' box showing tags like 'stable-perl', 'mainline-bookworm-perl', etc., and an 'About Official Images' box with a brief description and a 'Why Official Images?' link.

The screenshot shows a terminal window titled 'amitabhadey ~ com.docker.cli - docker pull nginx - 90x29'. The command 'docker pull nginx' is being run, and the output shows the progress of pulling the latest tag from the library/nginx repository. The output includes file names like '26070551e657', '5745264e68a8', '6f07c61775e7', 'c4f29c7c07f7', '5a639d96fb1', '3ba04a51efe1', and '716495aa6d18', followed by status messages like 'Extracting' and 'Download complete'.

Run a Container

docker run

This command is used to create and start a new container from a Docker image. `docker run -it ubuntu:latest bash` will start a new container based on the latest Ubuntu image and open a bash shell inside it.



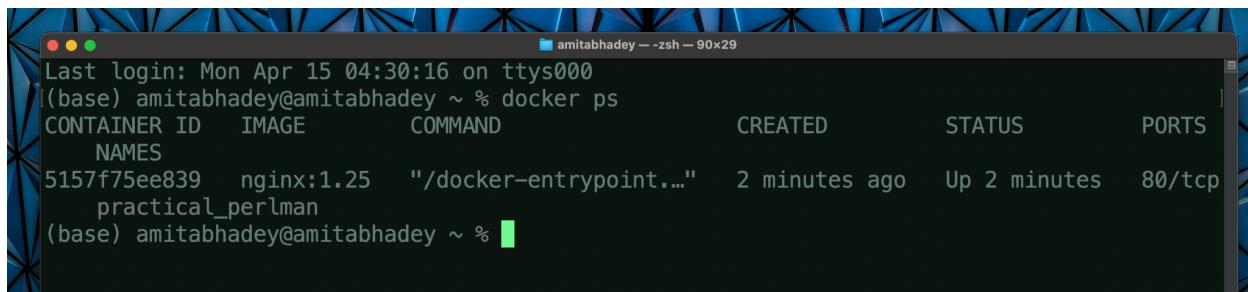
The screenshot shows a terminal window titled "amitabhadey ~ com.docker.cli • docker run nginx:1.25 — 90x29". The terminal displays the following command and its execution:

```
(base) amitabhadey@amitabhadey ~ % docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
nginx           1.25     48b4217efe5e   2 months ago   192MB
nginx           latest    48b4217efe5e   2 months ago   192MB
(base) amitabhadey@amitabhadey ~ % docker run nginx:1.25
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt
at repopulation
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ip
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/n
onf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/
.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolve
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/04/15 08:59:50 [notice] 1#1: using the "epoll" event method
2024/04/15 08:59:50 [notice] 1#1: nginx/1.25.4
2024/04/15 08:59:50 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-
2024/04/15 08:59:50 [notice] 1#1: OS: Linux 6.6.16-linuxkit
2024/04/15 08:59:50 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1
2024/04/15 08:59:50 [notice] 1#1: start worker processes
2024/04/15 08:59:50 [notice] 1#1: start worker process 29
2024/04/15 08:59:50 [notice] 1#1: start worker process 30
2024/04/15 08:59:50 [notice] 1#1: start worker process 31
```

[Ctrl + C : Container exits, and the process dies]

Check if Container is running

```
docker ps
```

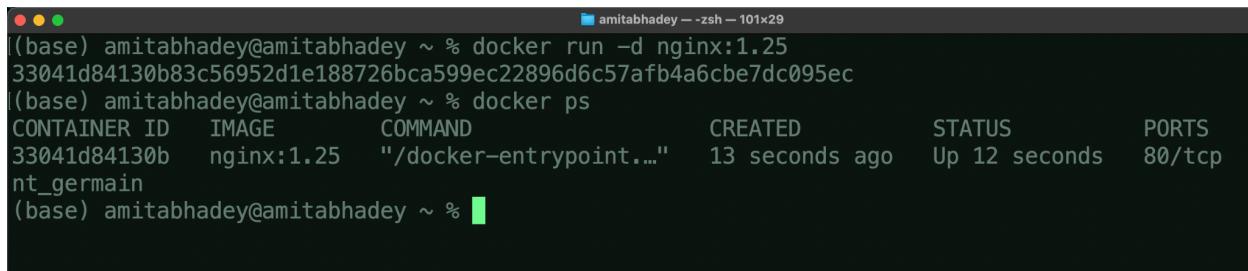


CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
5157f75ee839	nginx:1.25	"/docker-entrypoint...."	2 minutes ago	Up 2 minutes	80/tcp

Run Containers in Background

```
docker run -d or --detach
```

Runs container in background and prints the container ID

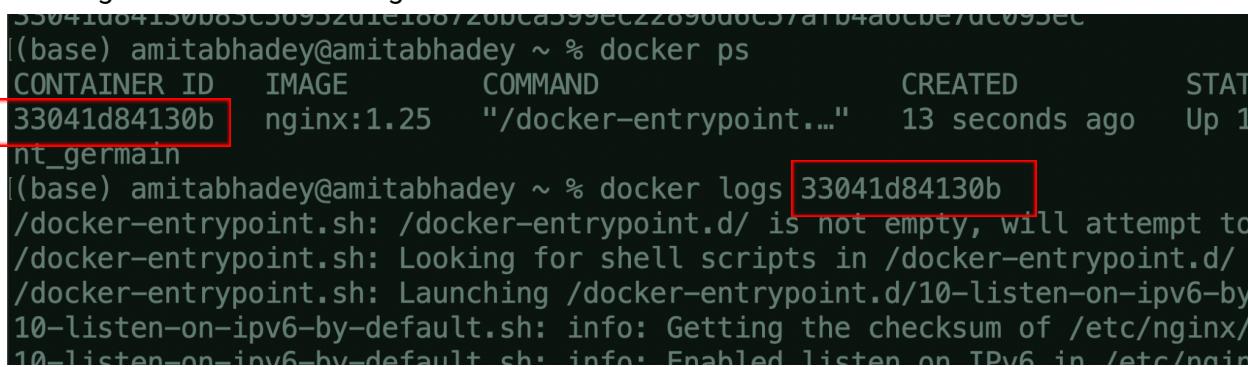


CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
33041d84130b	nginx:1.25	"/docker-entrypoint...."	13 seconds ago	Up 13 seconds	80/tcp

See the Container Logs

```
docker logs CONTAINER_ID  
docker logs CONTAINER_NAME
```

View logs from service running inside the container



```
33041d84130b83c56952d1e188726bca599ec22896d6c57afb4a6cbe7dc095ec  
|(base) amitabhadey@amitabhadey ~ % docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS  
33041d84130b nginx:1.25 "/docker-entrypoint...." 13 seconds ago Up 13 seconds nt_germain  
|(base) amitabhadey@amitabhadey ~ % docker logs 33041d84130b  
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to  
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/  
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-  
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/  
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/
```

[With docker run, you can run any image which may not necessarily be in your local machine]

Port Binding

```
docker run -p HOST_PORT:CONTAINER_PORT IMAGE_NAME:VERSION_TAG
```

- When you run a Docker container, you can specify port bindings to map ports on the host machine to ports within the container. This allows traffic coming to the host's specific port to be forwarded to the corresponding port inside the container.
- Port binding enables external clients to communicate with services running inside Docker containers as if they were running directly on the host machine.

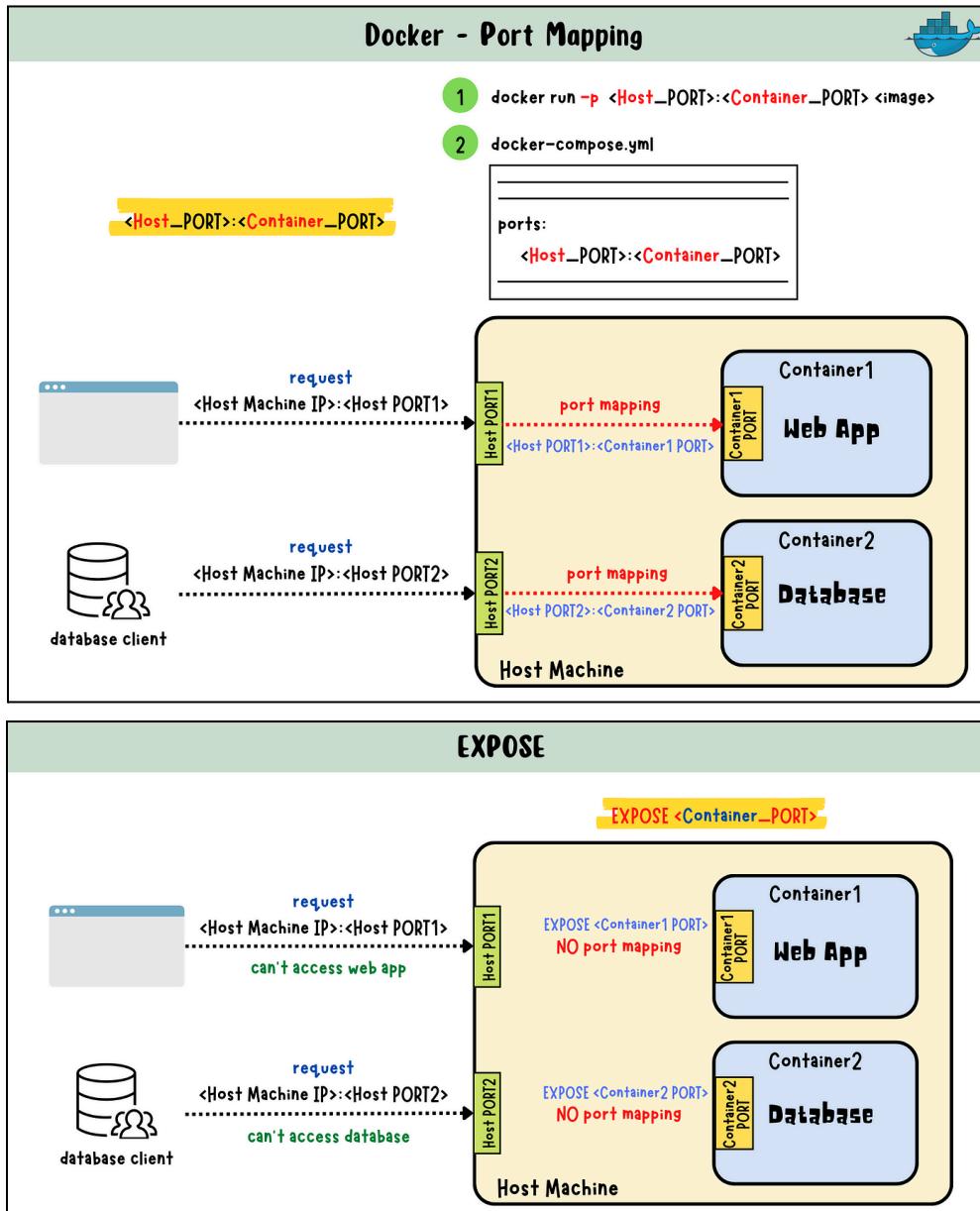
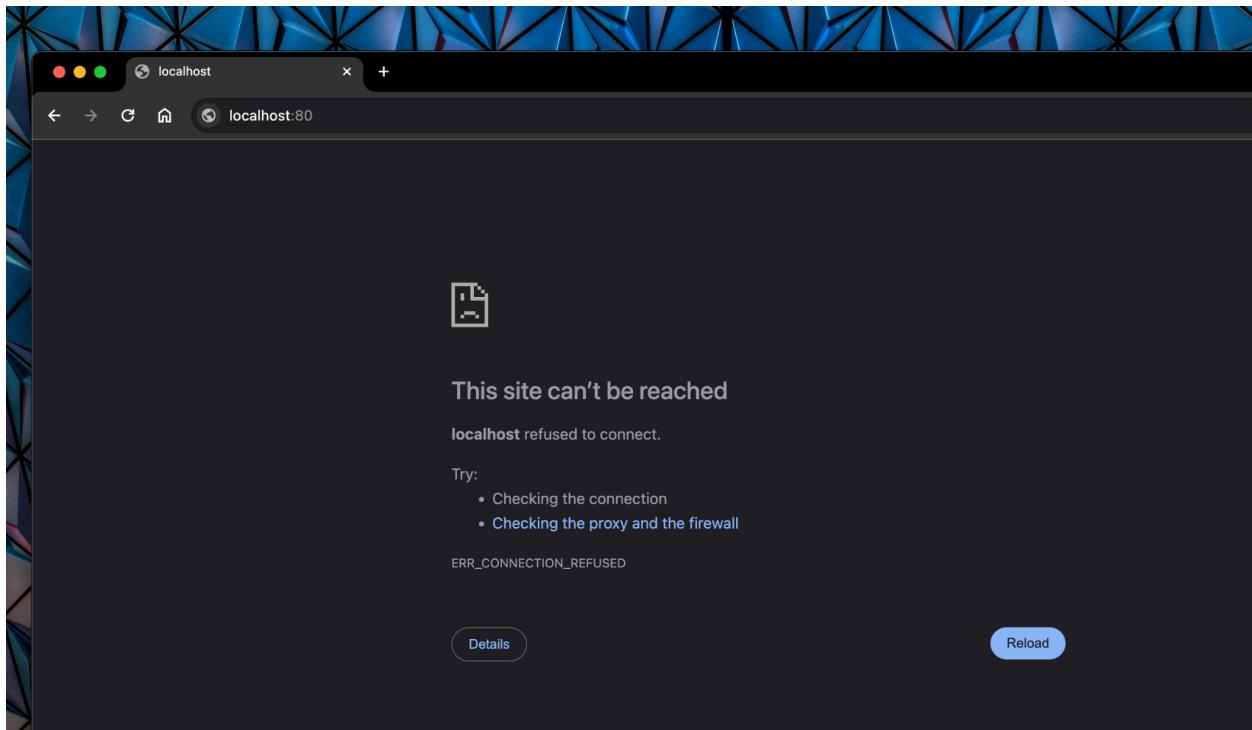
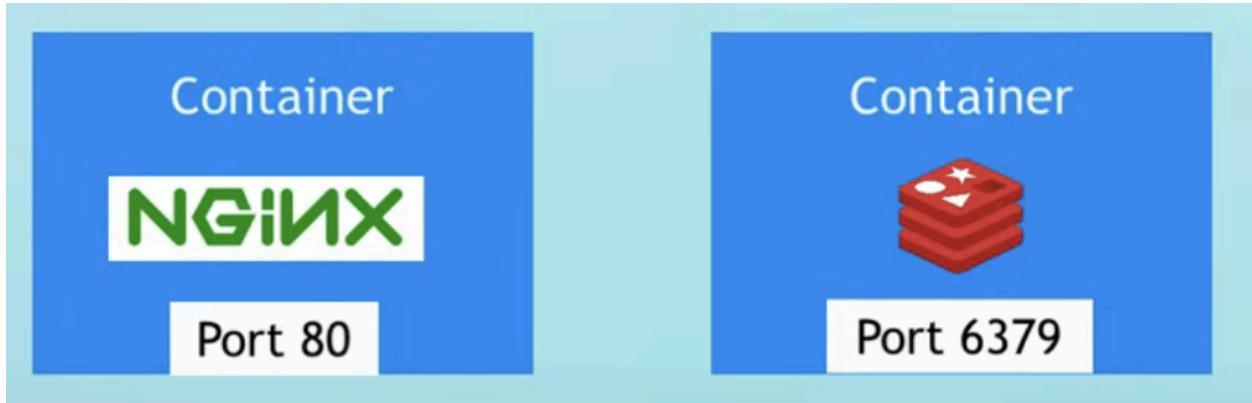


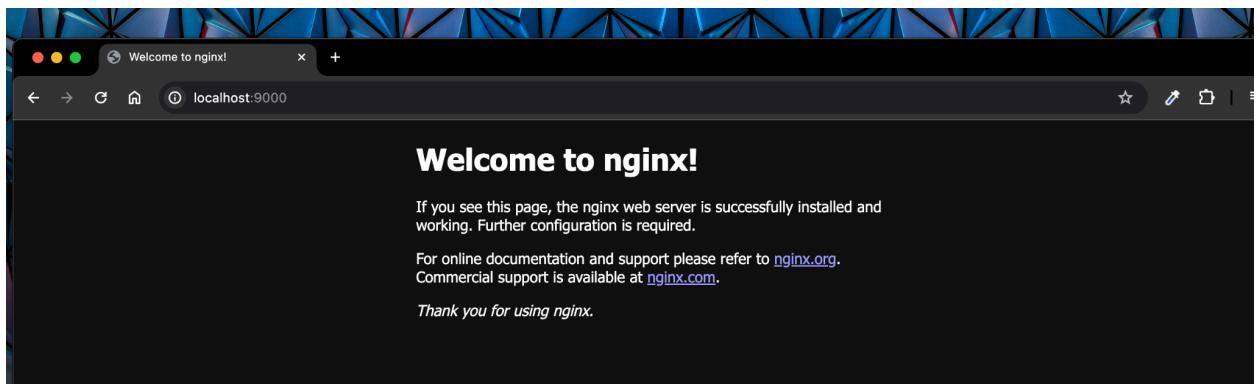
Image credit - Claire Lee @Medium



```
(base) amitabhadey@amitabhadey ~ % docker run -d -p 9000:80 nginx:1.25  
13c66171762b6306601c1484c299dba6916c75418851d1a40e0a8363fa77e95a  
(base) amitabhadey@amitabhadey ~ %
```

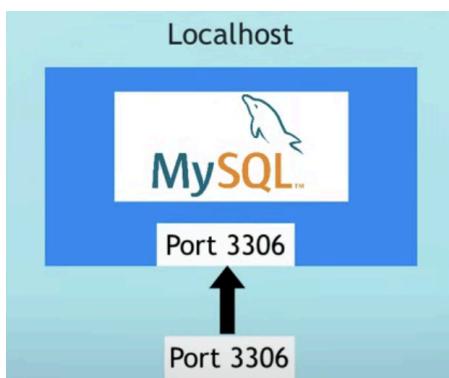
```
(base) amitabhadey@amitabhadey ~ % docker run -d -p 9000:80 nginx:1.25
13c66171762b6306601c1484c299dba6916c75418851d1a40e0a8363fa77e95a
(base) amitabhadey@amitabhadey ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
NAMES
13c66171762b nginx:1.25 "/docker-entrypoint..." About a minute ago Up About
:9000->80/tcp elegant_fermat
(base) amitabhadey@amitabhadey ~ %
```

[Only 1 service can run on a specific port on the host]



```
(base) amitabhadey@amitabhadey ~ % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
NAMES
13c66171762b nginx:1.25 "/docker-entrypoint..." 6 minutes ago Up
tcp elegant_fermat
(base) amitabhadey@amitabhadey ~ % docker logs 13c66171762b
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint
```

[A standard practice is to employ the identical port on your host system as the one utilized by the container]

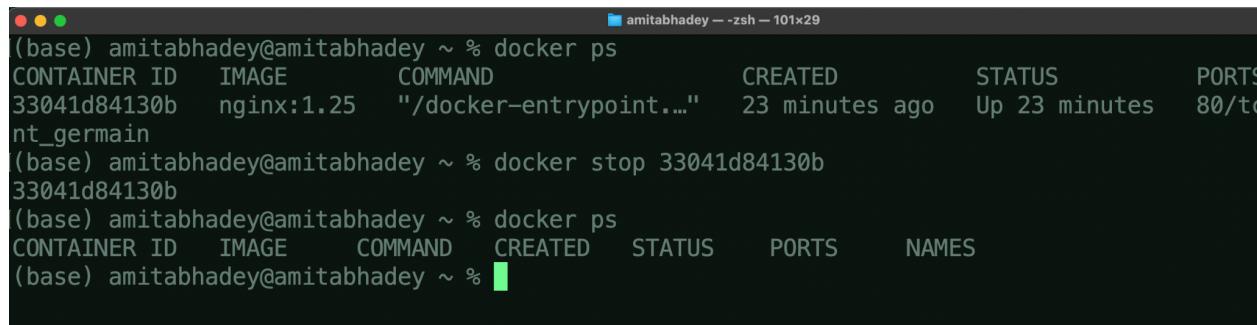


Stop the Container

```
docker stop CONTAINER_ID  
docker stop CONTAINER_NAME
```

Stops one or more running containers.

docker stop <container_id> will stop the container with the specified ID.



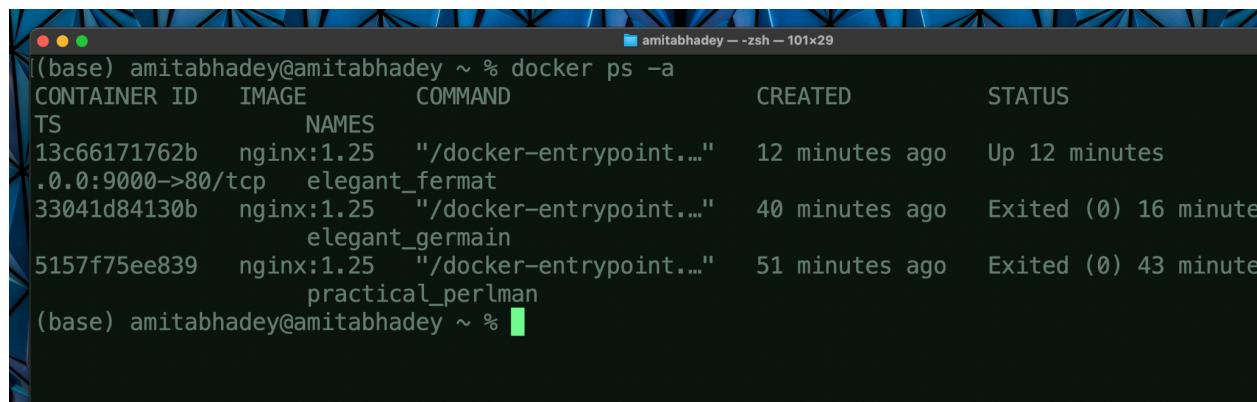
```
(base) amitabhadey@amitabhadey ~ % docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
33041d84130b nginx:1.25 "/docker-entrypoint...." 23 minutes ago Up 23 minutes 80/tcp  
nt_germain  
(base) amitabhadey@amitabhadey ~ % docker stop 33041d84130b  
33041d84130b  
(base) amitabhadey@amitabhadey ~ % docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
(base) amitabhadey@amitabhadey ~ %
```

List all containers (stopped and running)

```
docker ps -a
```

Lists all containers, including those that are stopped or exited.

docker ps -a will show all containers, regardless of their state.

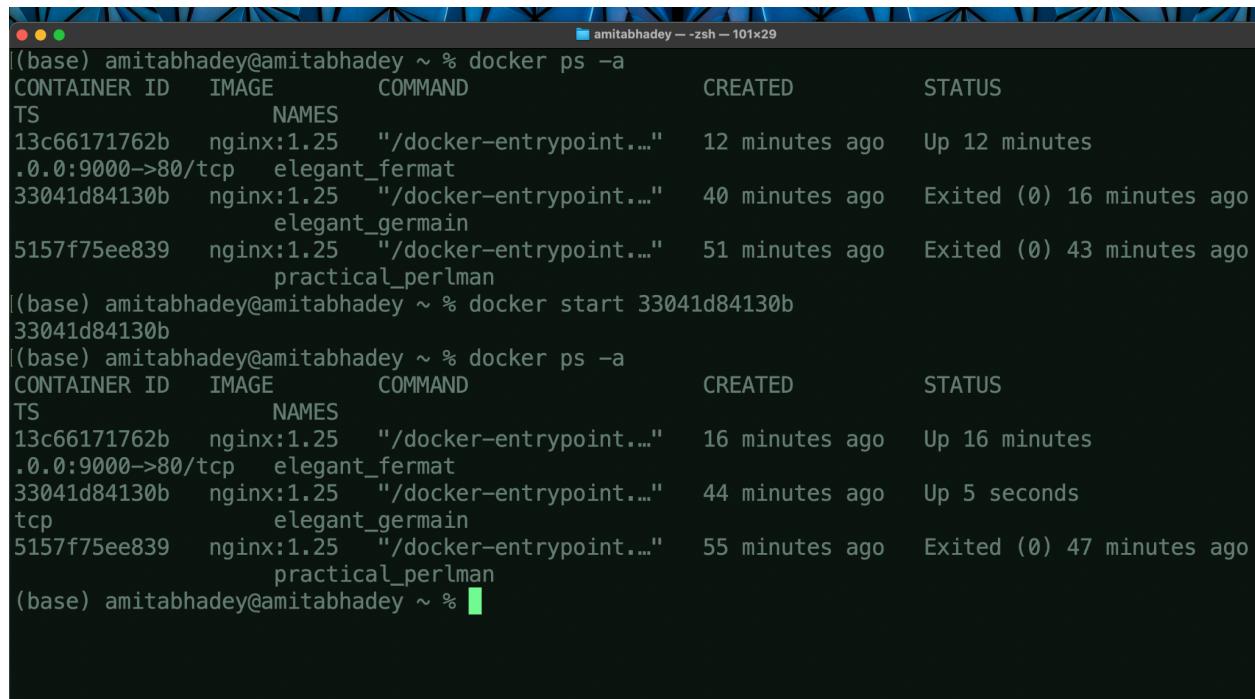


```
(base) amitabhadey@amitabhadey ~ % docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS NAMES  
TS NAMES  
13c66171762b nginx:1.25 "/docker-entrypoint...." 12 minutes ago Up 12 minutes  
.0:9000->80/tcp elegant_fermat  
33041d84130b nginx:1.25 "/docker-entrypoint...." 40 minutes ago Exited (0) 16 minutes  
elegant_germain  
5157f75ee839 nginx:1.25 "/docker-entrypoint...." 51 minutes ago Exited (0) 43 minutes  
practical_perlman  
(base) amitabhadey@amitabhadey ~ %
```

Restart a Container

```
docker start CONTAINER_ID
```

Containers can be restarted with **docker start CONTAINER_ID** without having to create new containers



The screenshot shows a terminal window titled "amitabhadey ~ zsh - 101x29". It displays two sessions of Docker commands. The first session shows the output of `docker ps -a`, listing four containers: `13c66171762b` (status Up 12 minutes), `33041d84130b` (status Exited (0) 16 minutes ago), `5157f75ee839` (status Exited (0) 43 minutes ago), and `33041d84130b` again (status Up 16 minutes). The second session shows the command `docker start 33041d84130b` being run, followed by another `docker ps -a` which now shows the container `33041d84130b` with a status of Up 5 seconds.

```
(base) amitabhadey@amitabhadey ~ % docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS
TS             NAMES
13c66171762b   nginx:1.25  "/docker-entrypoint..."  12 minutes ago  Up 12 minutes
.0.0:9000->80/tcp   elegant_fermat
33041d84130b   nginx:1.25  "/docker-entrypoint..."  40 minutes ago  Exited (0) 16 minutes ago
                                         elegant_germain
5157f75ee839   nginx:1.25  "/docker-entrypoint..."  51 minutes ago  Exited (0) 43 minutes ago
                                         practical_perlman
(base) amitabhadey@amitabhadey ~ % docker start 33041d84130b
33041d84130b
(base) amitabhadey@amitabhadey ~ % docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS
TS             NAMES
13c66171762b   nginx:1.25  "/docker-entrypoint..."  16 minutes ago  Up 16 minutes
.0.0:9000->80/tcp   elegant_fermat
33041d84130b   nginx:1.25  "/docker-entrypoint..."  44 minutes ago  Up 5 seconds
                                         elegant_germain
5157f75ee839   nginx:1.25  "/docker-entrypoint..."  55 minutes ago  Exited (0) 47 minutes ago
                                         practical_perlman
(base) amitabhadey@amitabhadey ~ % █
```

Assigning a Container Name

```
docker run --name CONTAINER_NAME
```

Complete command:

```
docker run --name CONTAINER_NAME -d -p HOST_PORT:CONTAINER_PORT  
IMAGE_NAME:VERSION_TAG
```

Assign a name to the container

```
(base) amitabhadey@amitabhadey ~ % docker run --name csc360 -d -p 9000:80 nginx:1.25  
30de6cf6f84479f359a9570af4087a98fb29456419fc35e24a351439043386c0  
(base) amitabhadey@amitabhadey ~ % docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS  
PORTS NAMES  
30de6cf6f844 nginx:1.25 "/docker-entrypoint...."  
0.0.0.0:9000->80/tcp csc360  
0e963394a5c4 nginx:1.25 "/docker-entrypoint...."  
web-app  
13c66171762b nginx:1.25 "/docker-entrypoint...."  
0 elegant_fermat  
33041d84130b nginx:1.25 "/docker-entrypoint...."  
0 elegant_germain
```

If a user doesn't specify a name explicitly when creating a container, Docker automatically generates a random, unique name for the container.

These names typically consist of two randomly chosen words separated by an underscore, such as "adoring_mahavira" or "hopeful_albattani".

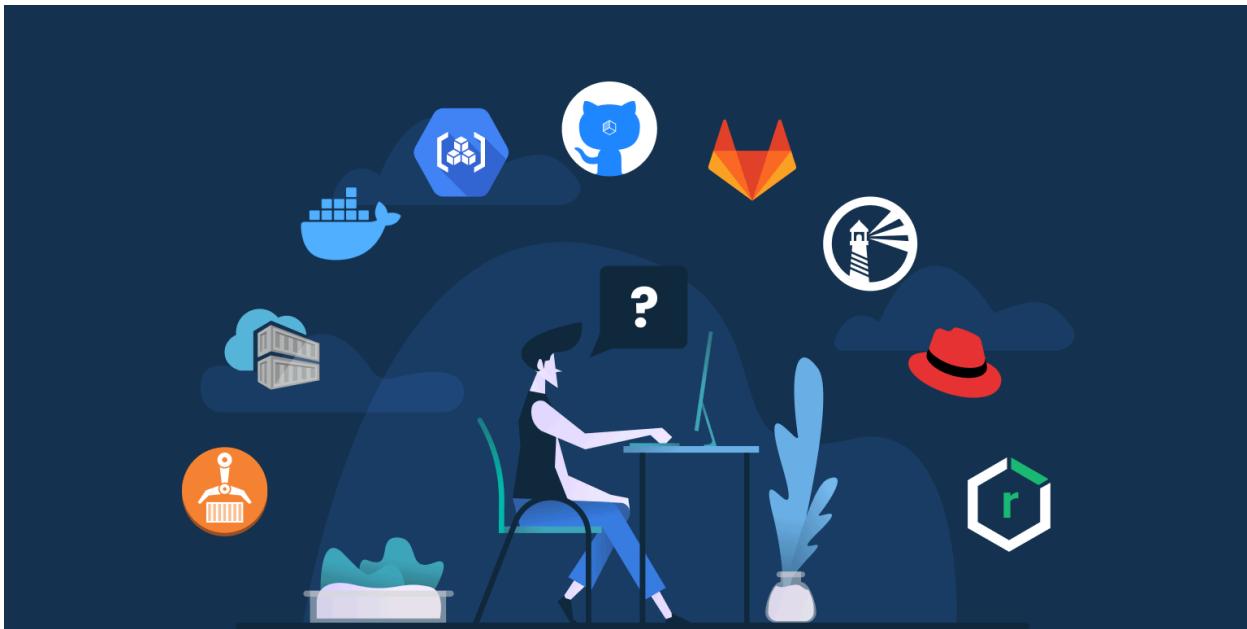
Docker imposes some rules and limitations on container names:

- Names must start with a letter or a lowercase letter.
- Subsequent characters in the name can be letters (uppercase or lowercase), numbers, underscores, or hyphens.
- Names can be up to 255 characters long.
- Names must be unique among the container names on a Docker host.

Renaming a Container Name

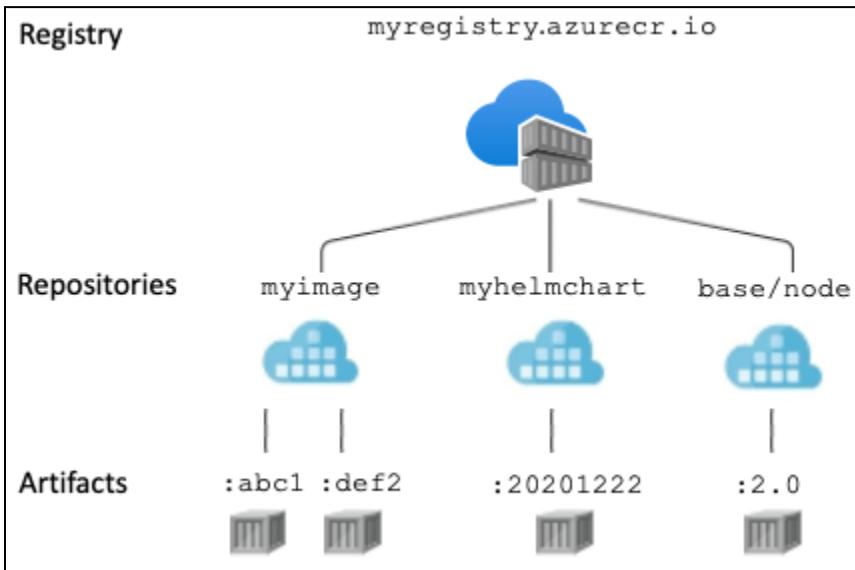
```
docker rename old_container new_container
```

Public and Private Docker Registries



A Docker **registry** is a service that stores Docker images, while a **repository** is a collection of related Docker images within a registry.

Registries provide storage and distribution capabilities, while repositories allow users to organize and manage Docker images with similar content or purpose.



Artifacts are the output files or products generated during the software development process. They include compiled code, libraries, executables, documentation, configuration files, and any other files produced as a result of building or deploying software.

Artifacts are created at various stages of the development lifecycle, such as compiling source code, running tests, packaging applications, and deploying releases.

Registry vs Repository

A Docker registry is the storage and distribution service for Docker images, a repository is a collection of related Docker images within a registry, providing organization and management capabilities.

Registry:

- A Docker registry is a service that stores Docker images.
- It serves as a centralized location where Docker users can push, pull, and manage Docker images.
- Examples include Docker Hub, which is the default public Docker registry, and private registries hosted by organizations or individuals.

Repository:

- A Docker repository is a collection of related Docker images within a registry.
- It serves as the primary namespace for organizing and managing Docker images.
- Repositories allow users to version, tag, and organize Docker images with similar content or purpose.

