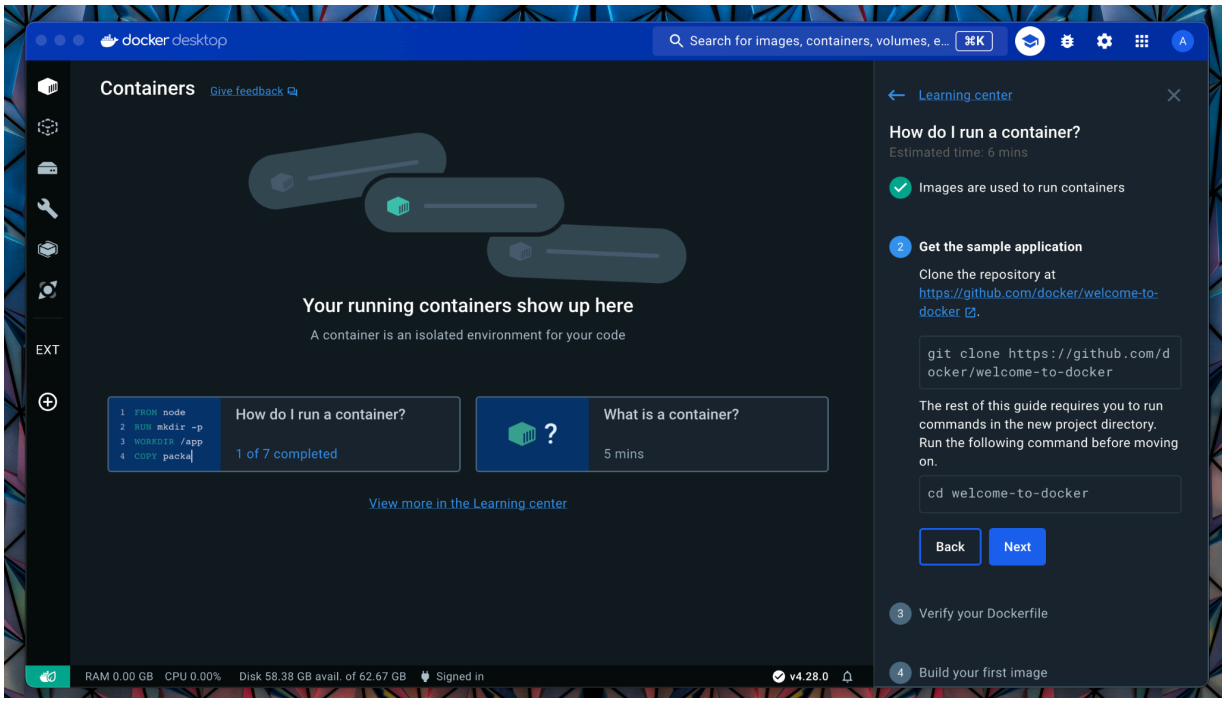
Install Docker on your local machine

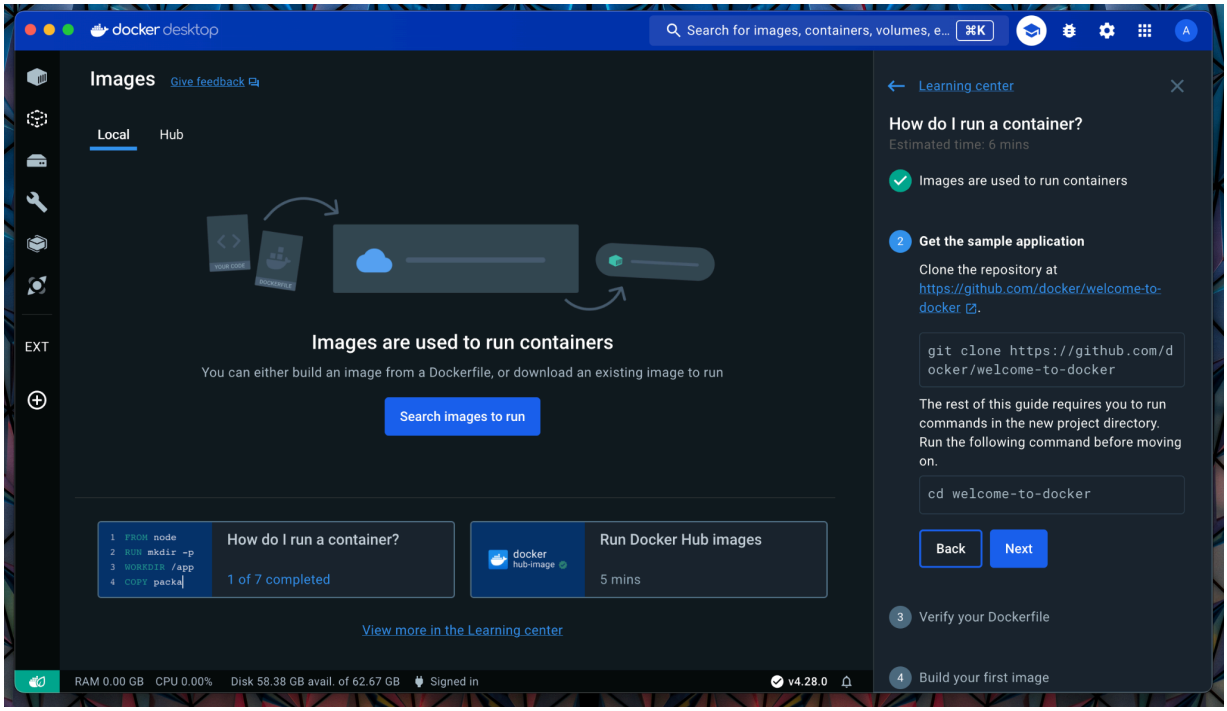
Download the correct installer for your operating system and run the installation:
https://docs.docker.com/get-docker/

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

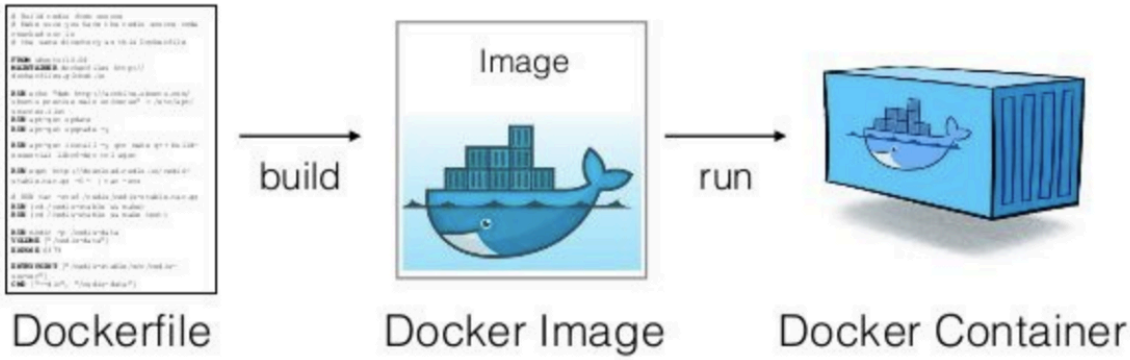
Always best to **refer to the latest installation guide** in the official documentation.

Open Docker Engine





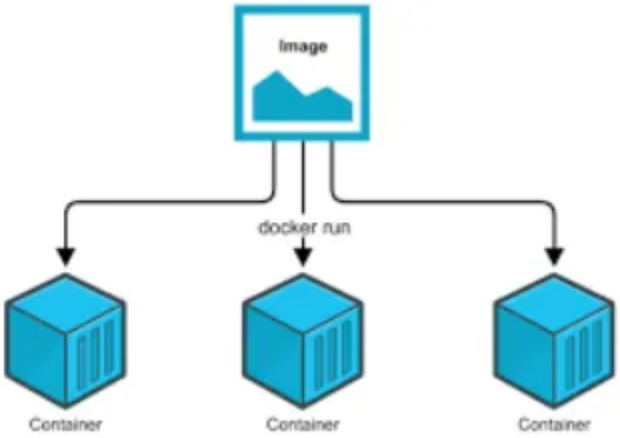
Docker Images vs Docker Containers



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**.

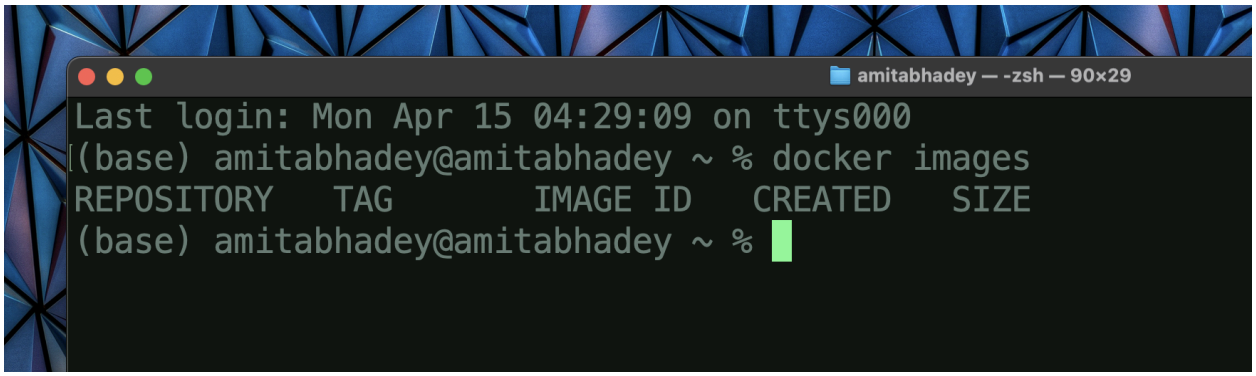


Docker Commands

List Docker Images

`docker images`

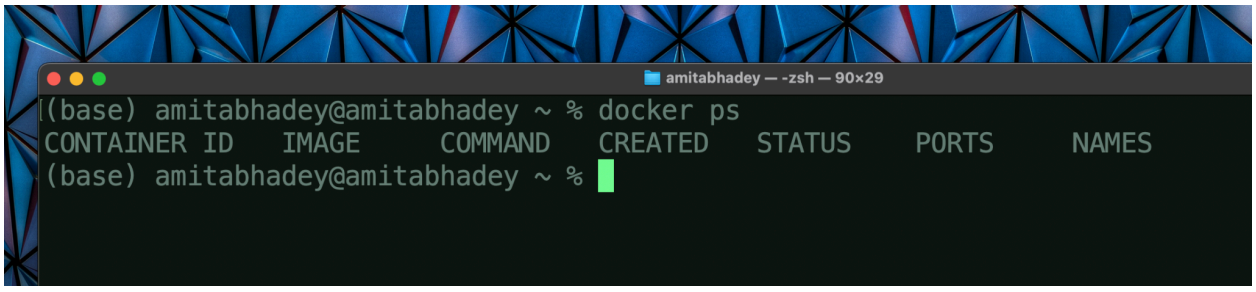
Lists all Docker images available locally on your system.
docker images will display a list of all images along with their tags and sizes.



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.

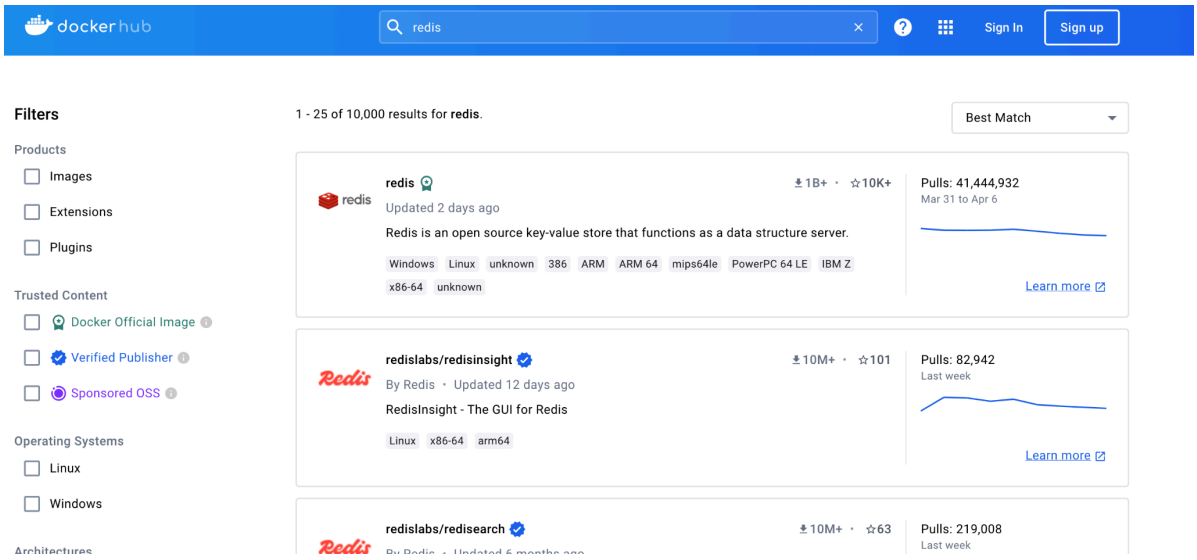


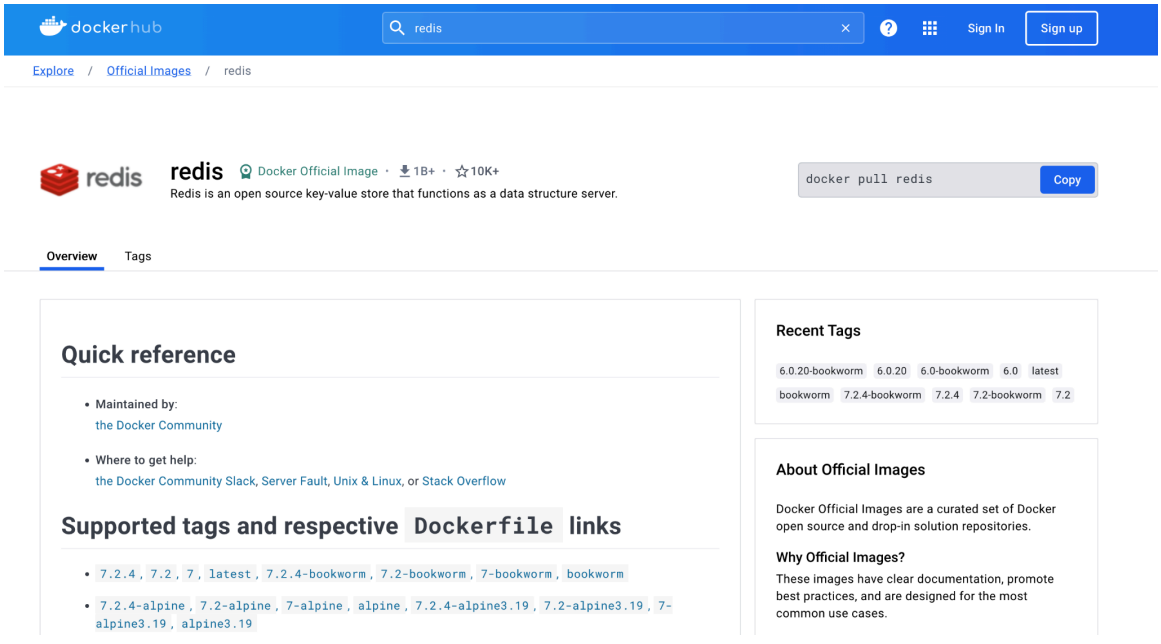
Docker Registry

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/





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.

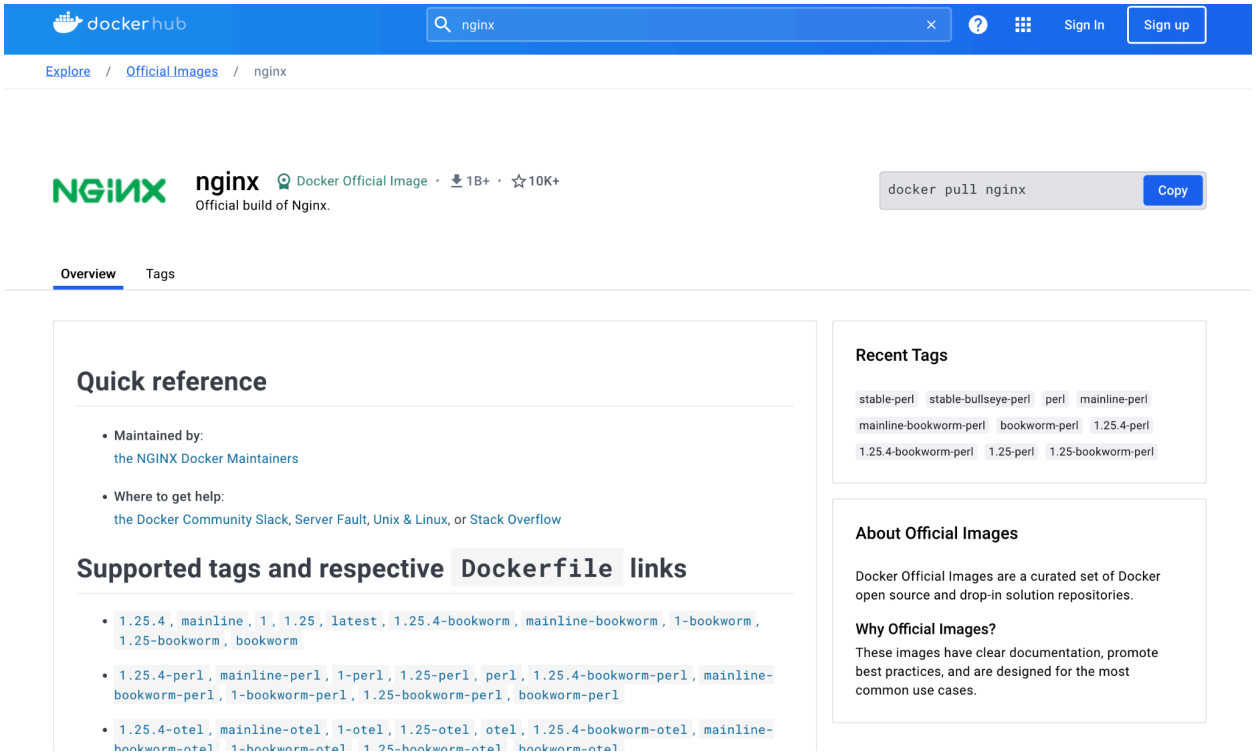
- **Image:** Specifies the Docker image used to create the container. It defines the filesystem, environment, and runtime configuration of the container.

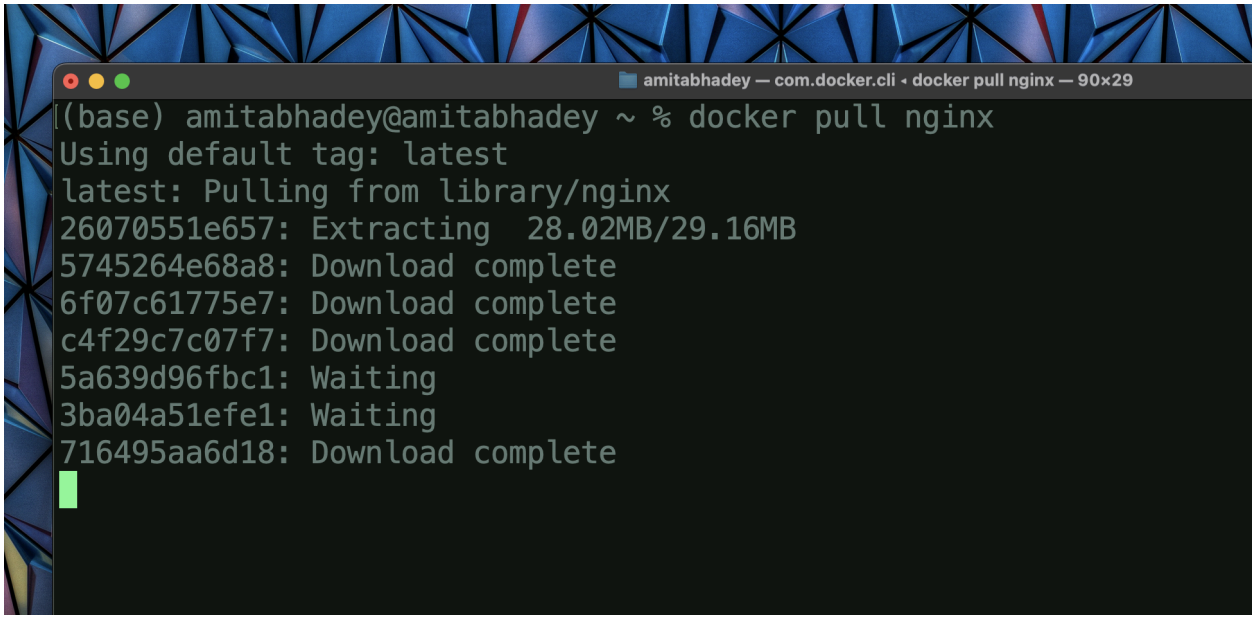
- **Command:** Specifies the command or process to be executed inside the container when it starts. It can be overridden during container creation or specified in the Dockerfile.

- **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
```

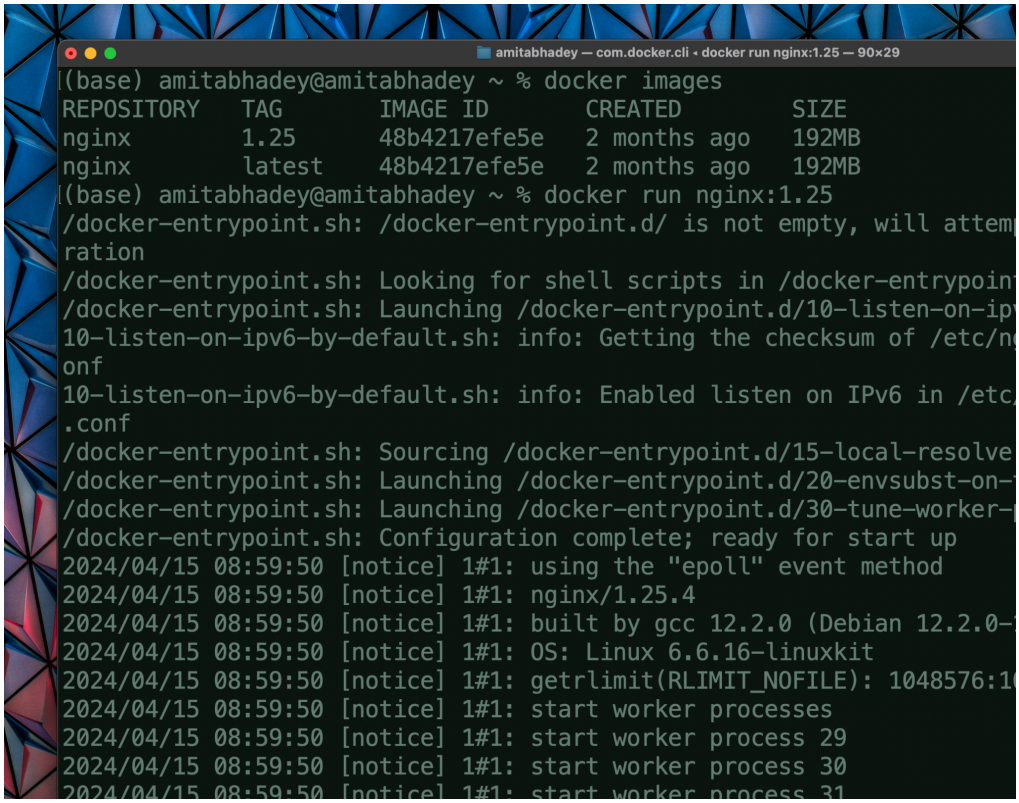




Run a Container

`docker run`

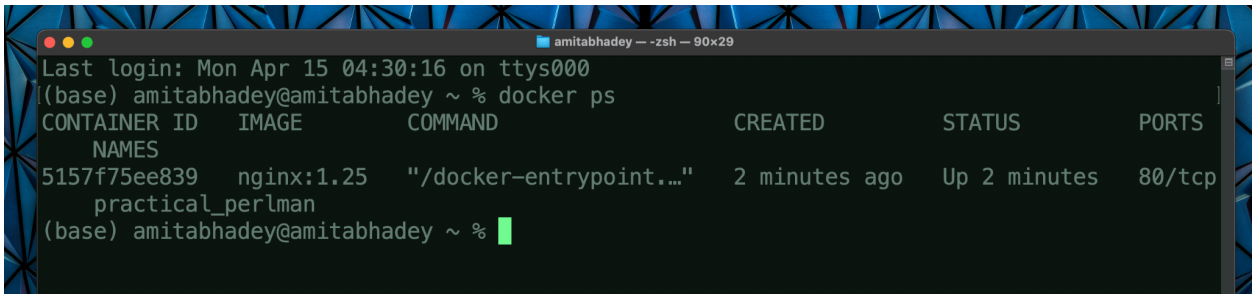
This command is used to create and start a new container from a Docker image.
docker run -it ubuntu:latest bash will start a new container based on the latest Ubuntu image and open a bash shell inside it.



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

Check if Container is running

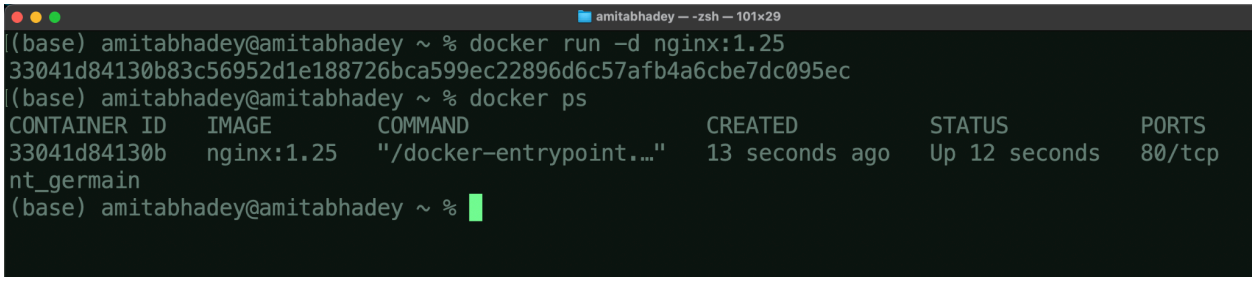
`docker ps`



Run Containers in Background

`docker run -d or --detach`

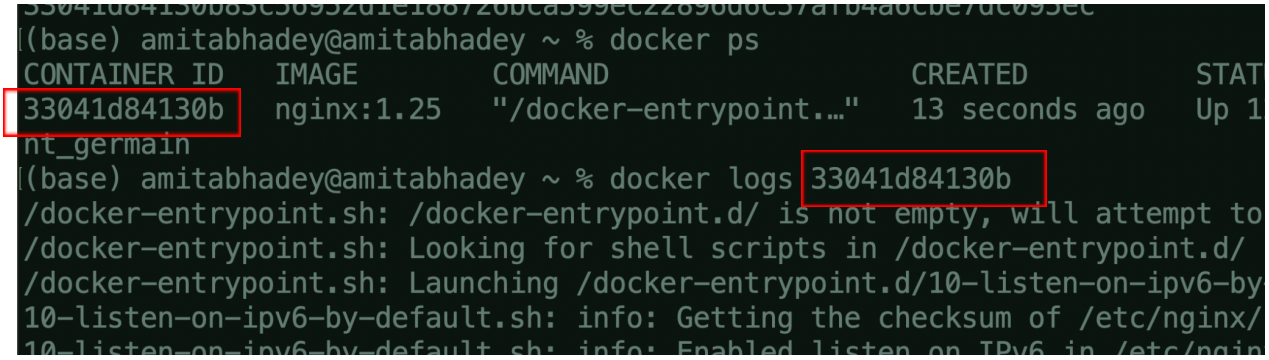
Runs container in background and prints the container ID



See the Container Logs

`docker logs CONTAINER_ID`
`docker logs CONTAINER_NAME`

View logs from service running insider the container

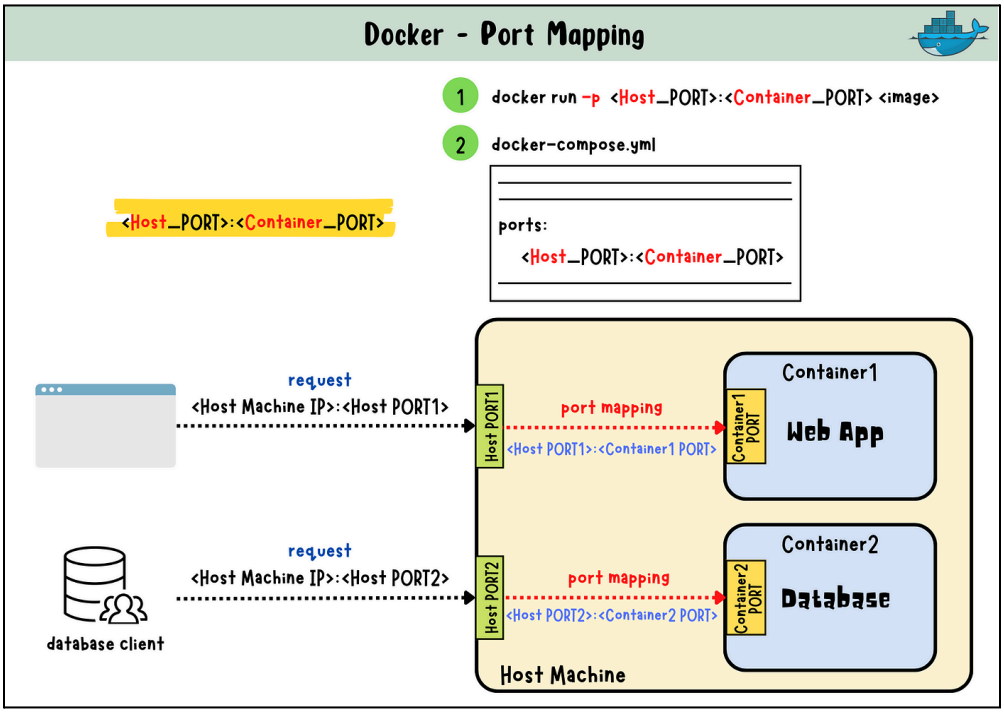


[With docker run, you can run any image which may not necessarily be in your local machine]

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.



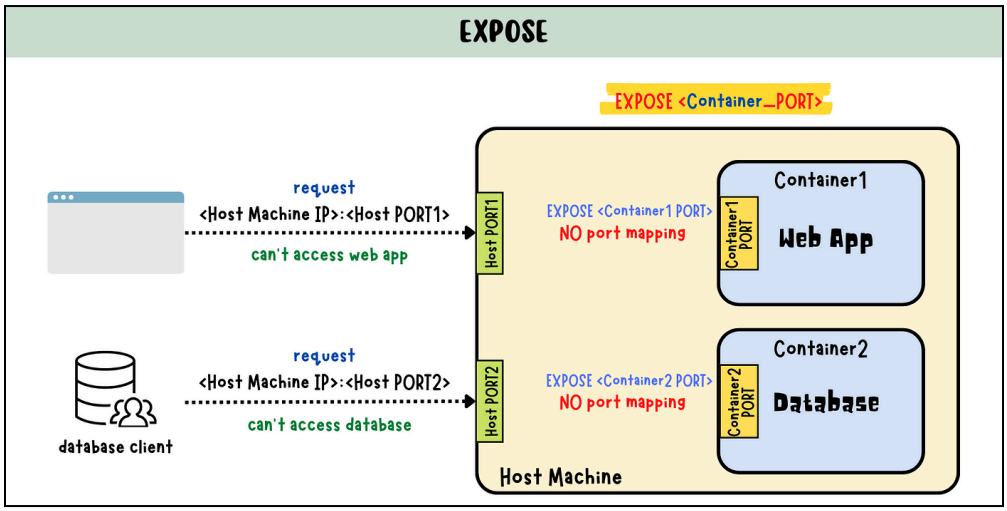
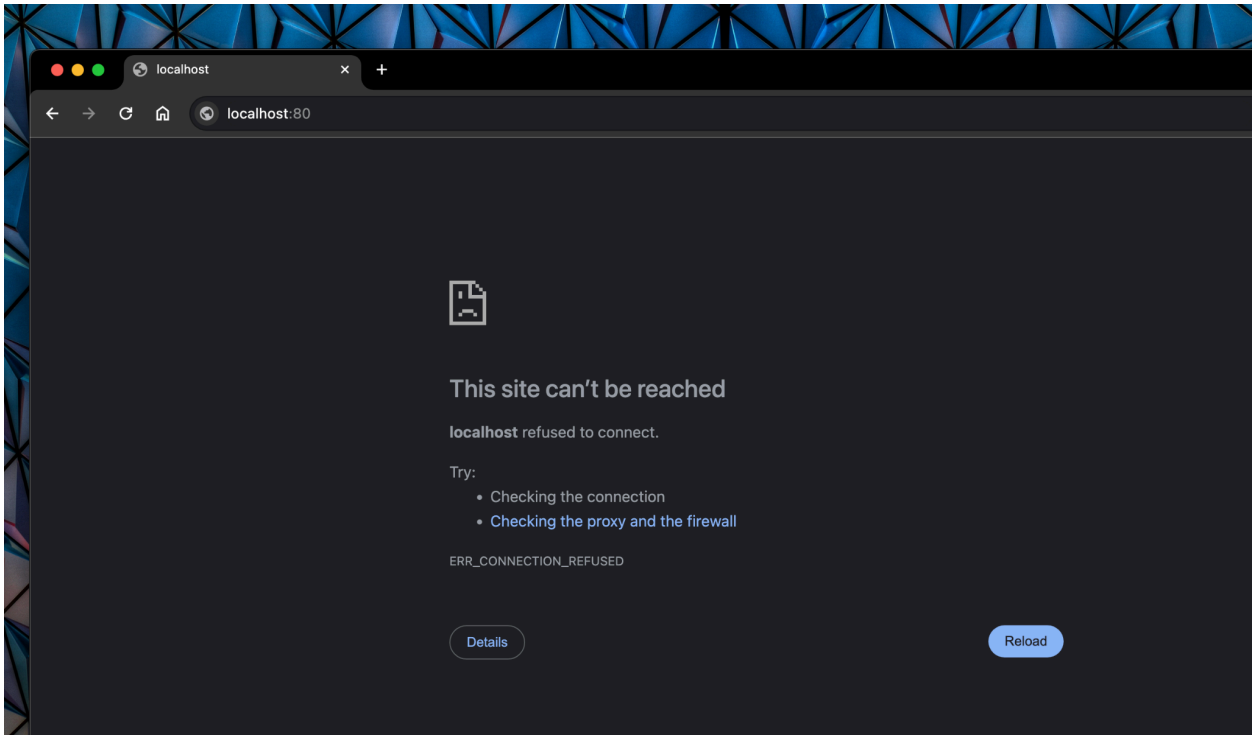
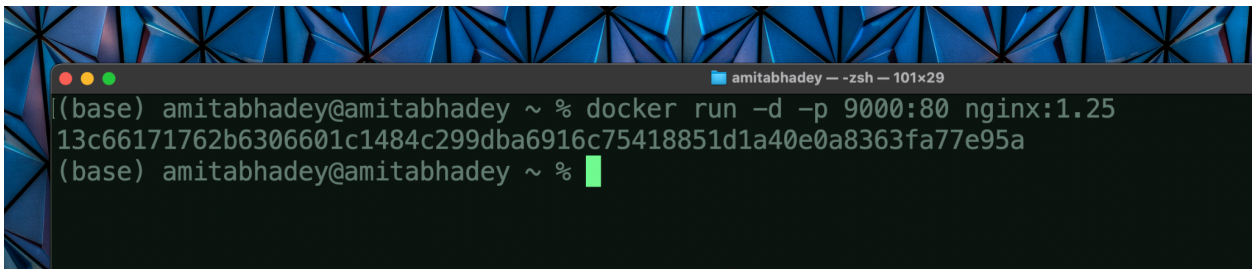
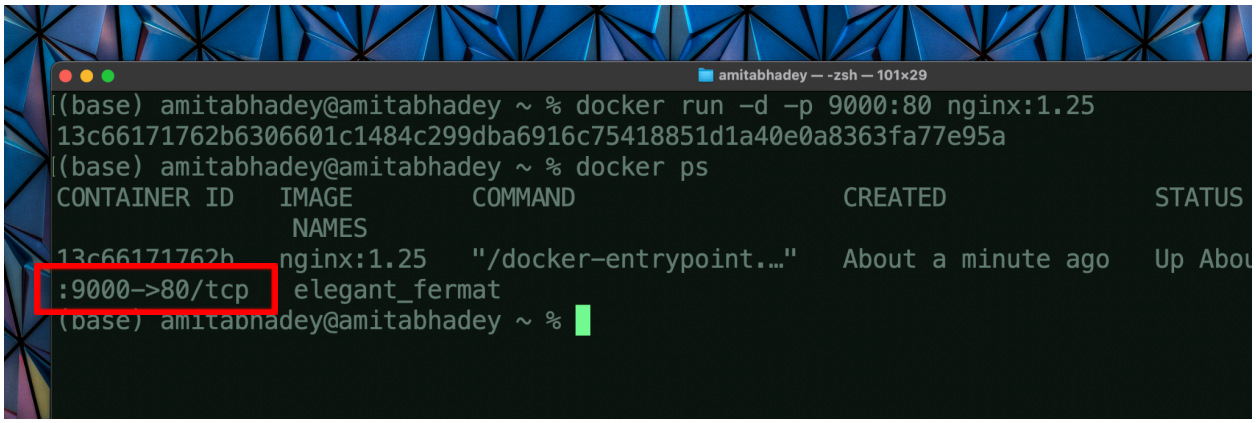


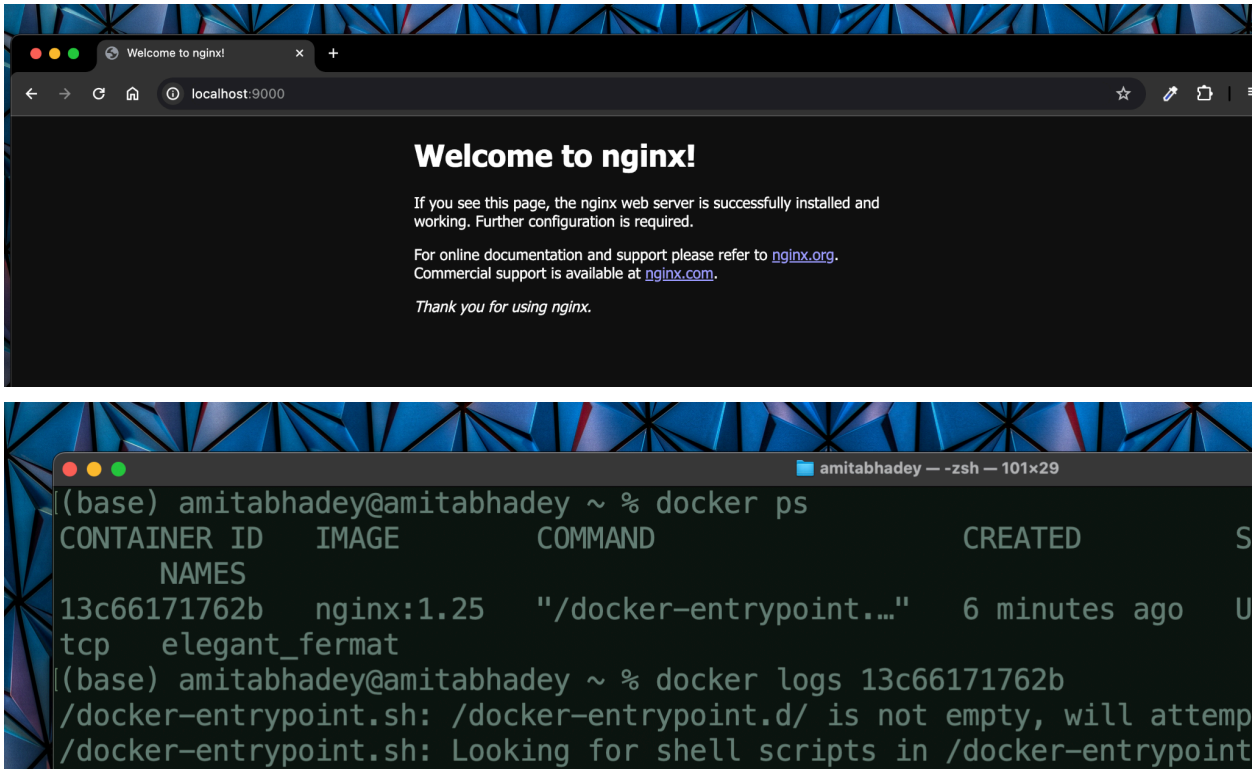
Image credit - Claire Lee @Medium



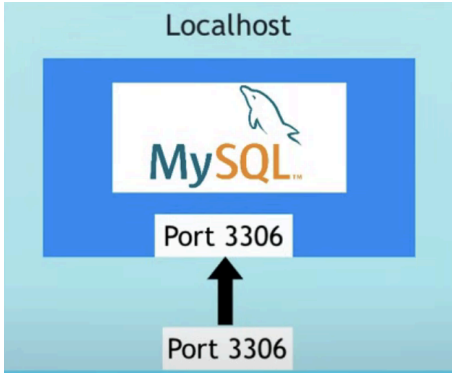




[Only 1 service can run on a specific port on the host]



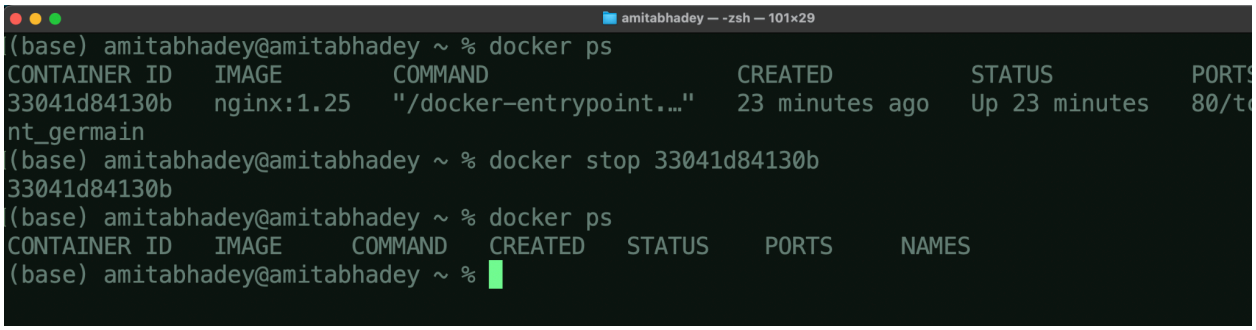
[A standard practice is to employ the identical port on your host system as the one utilized by the container]



Stop the Container

`docker stop CONTAINER_ID`
`docker stop CONTAINER_NAME`

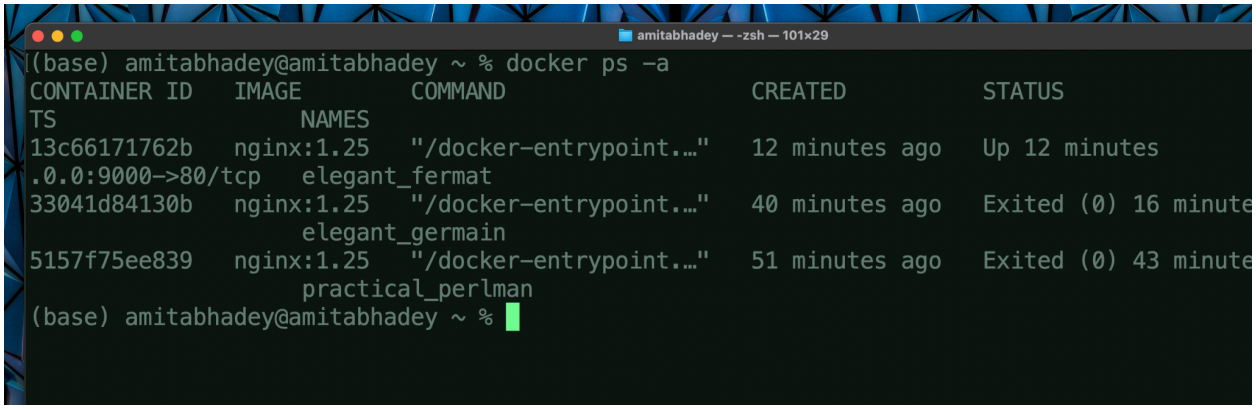
Stops one or more running containers.
docker stop <container_id> will stop the container with the specified ID.



List all containers (stopped and running)

`docker ps -a`

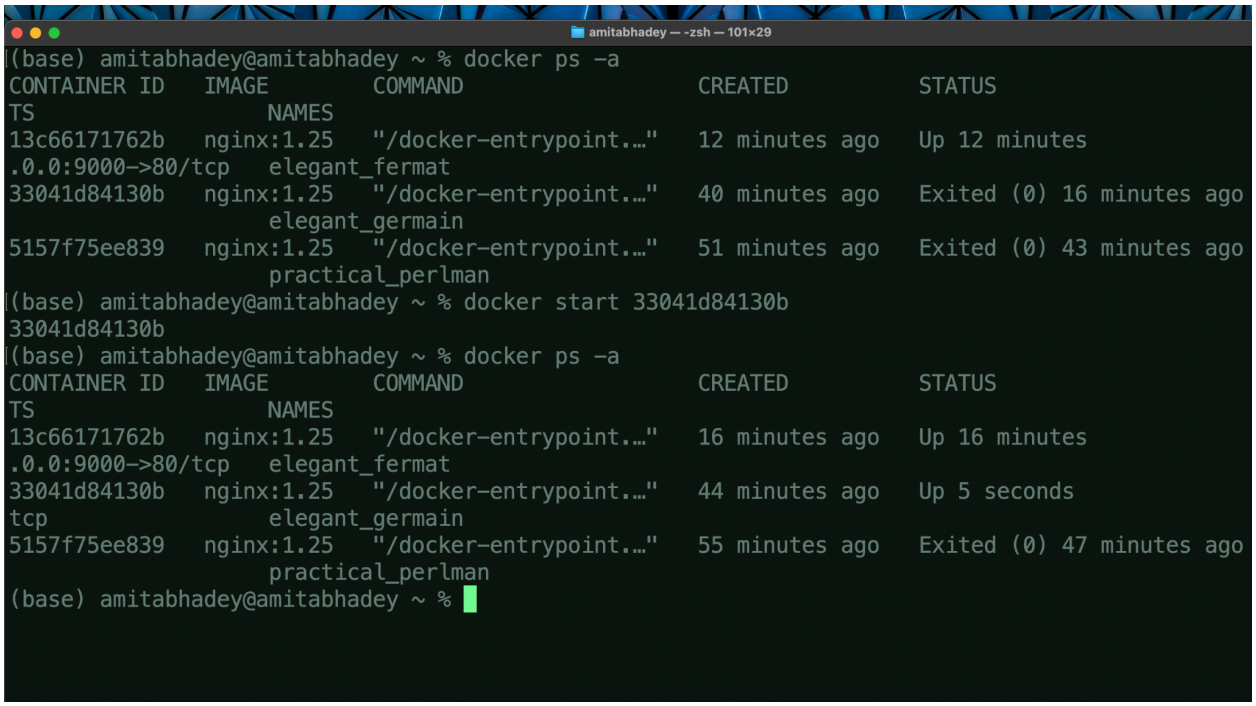
Lists all containers, including those that are stopped or exited.
docker ps -a will show all containers, regardless of their state.



Restart a Container

`docker start CONTAINER_ID`

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



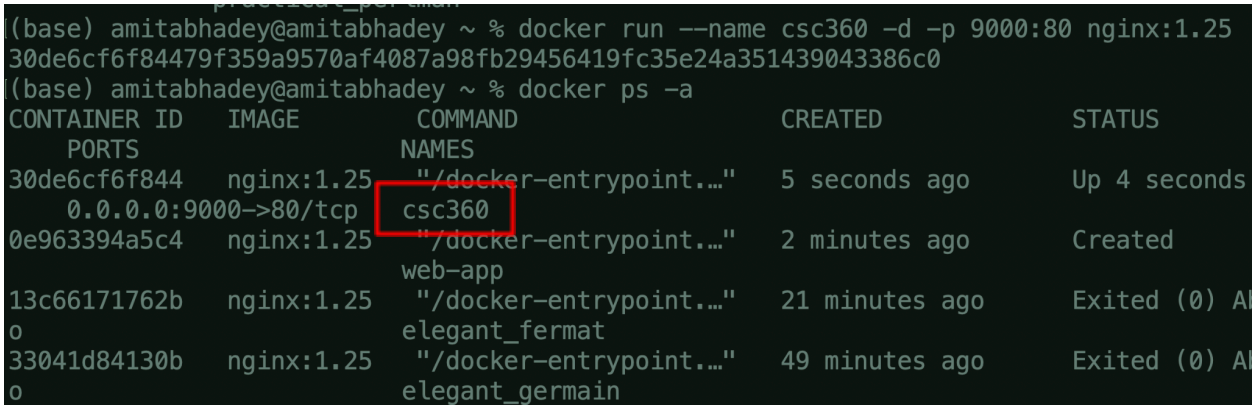
Assigning a Container Name

```
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



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".

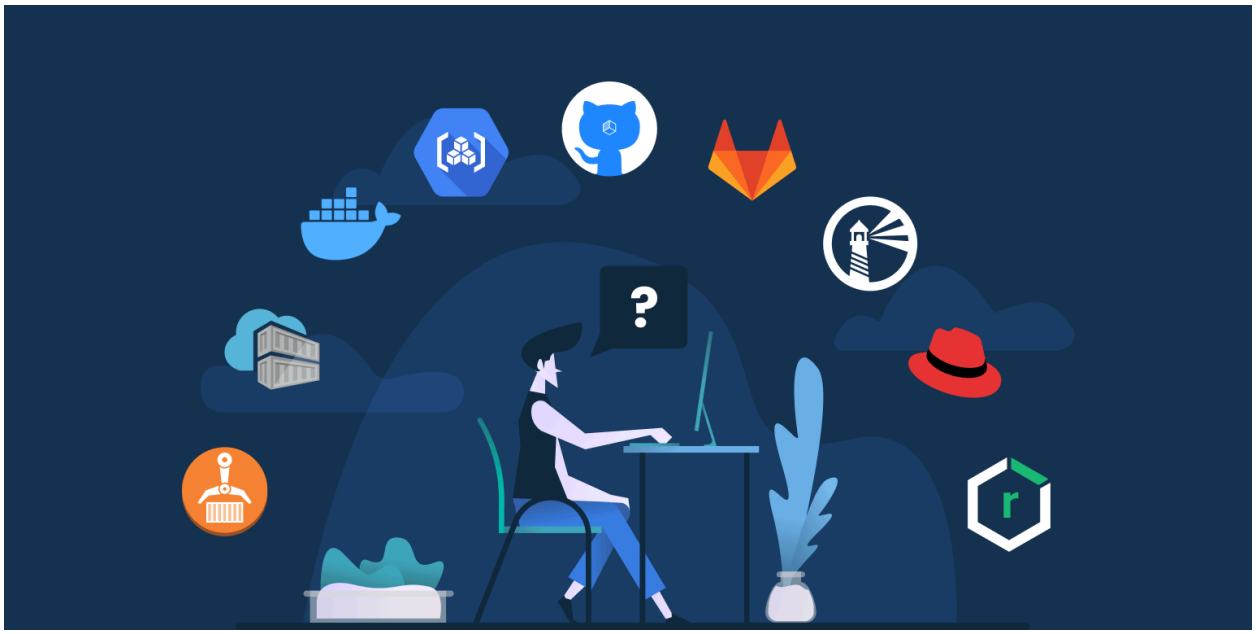
Docker imposes some rules and limitations on container names:

- 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

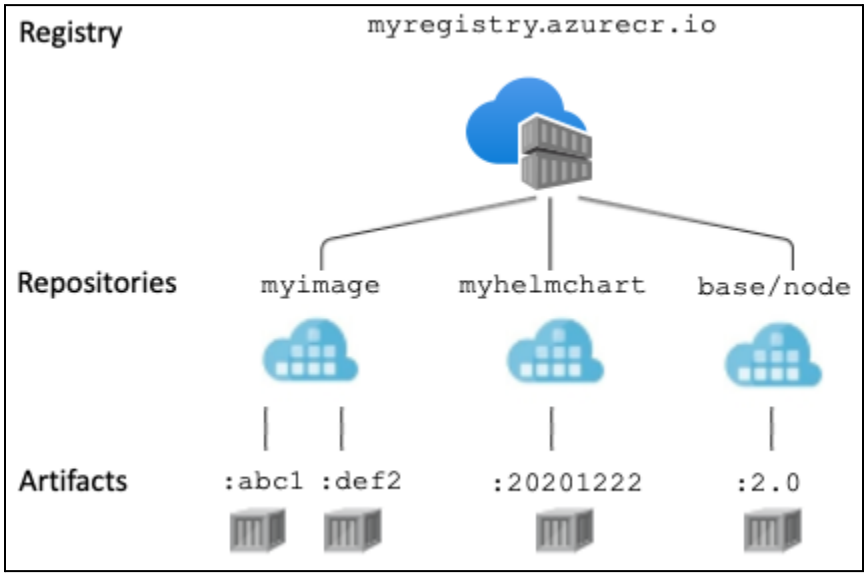
```
docker rename old_container new_container
```

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.