Getting Started

with

Docker

npNOG 10

November 25 - 28, 2024

About Docker

- Docker debuted to the public in Santa Clara at PyCon in 2013.
- Written in Go Language.
- Products maintained by Docker, Inc.
  - Docker Engine
  - Docker Engine Enterprise
  - Docker Hub
  - Docker Desktop
- Original Author: Solomon Hykes
- It was released as open-source.

What is Docker?

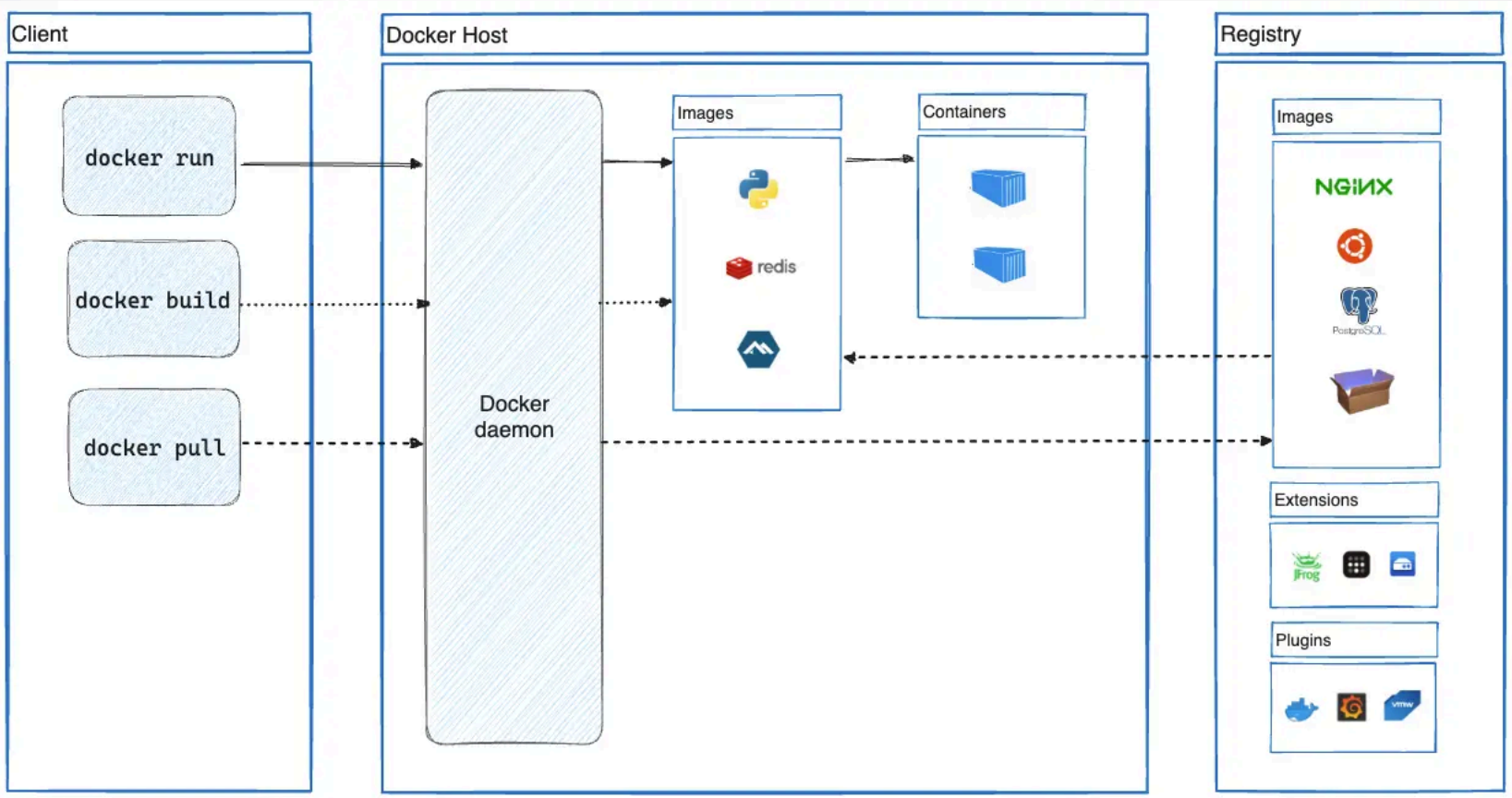
Docker is a containerization platform that allows you to package, ship, and run applications in containers. Containers are lightweight and portable, and they provide a consistent and reliable way to deploy applications across different environments.

- Installing Docker = Installing Docker Engine + Docker CLI
- Docker Engine is a daemon wihch manages containers, the same way that a hypervisor manages VMs.
- Docker Engine is the core of Docker, and it is responsible for managing containers and images.
- Docker CLI is used to interact with the Docker Engine.
- Docker CLI and Docker Engine communicate through an API.
- However, there are many other programs, and client libraries, to use the API.

Docker Architecture

- Docker uses a client-server architecture.
- Docker client talks to the Docker daemon.
- Docker daemon 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.
- The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.

Docker Architecture (diagram)



The Docker daemon

- The Docker daemon is the background service that manages building, running, and distributing Docker containers.
- The Docker daemon runs on the host machine and manages Docker objects such as images, containers, networks, and volumes.
- The Docker daemon is responsible for starting and stopping containers, managing images, and networking.
- The Docker daemon (dockerd) listens for Docker API requests.

The Docker client

- The Docker client is the command line interface to the Docker daemon.
- The Docker client (docker) is the primary way that many Docker users interact with Docker.
- When you type a docker command, the client sends the command to a Docker daemon (dockerd) to act upon.
- The docker command uses the Docker API.
- 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
- It enables you to build and share containerized applications and microservices.
- It includes the Docker daemon (dockerd), the Docker client (docker), Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper.
- It is a graphical user interface (GUI) for Docker.
- It provides a simple and easy-to-use interface for managing Docker containers and images.
- It is a great tool for developers who want to get started with Docker quickly and easily.

Docker Registries

- Docker Registries are repositories for Docker images.
- They are used to store and distribute Docker images.
- Docker Registries can be public or private.
- Public Registries are open to the public and anyone can access them.
- Docker Hub is a public registry that anyone can use, and Docker looks for images on Docker Hub by default.
- You can even run your own private registry.

Docker Objects

- Docker objects are the building blocks of Docker.
- They are the things that Docker manages.
- There are four main types of Docker objects:
  - Images
    - Images are the blueprint for containers.
  - Containers
    - Containers are the running instances of images.
  - Networks
    - Networks are the way containers communicate with each other.
  - Volumes
    - Volumes are the way containers store data.
- Each of these objects has its own set of commands and options.
- You can use the docker command to manage these objects.
- You can also use the Docker API to manage these objects.

Docker Images

- Docker images are the blueprint for containers.
- They are the files that are used to create containers.
- Image = files + metadata
- The files forms the root filesystem of the container.
- The metadata describes the image.
  - the author of the image
  - the command to execute in the container when starting it
  - environment variables to be set and etc
- Images are made of layers, conceptually stacked on top of each other.
- Each layer can add, change and remove files and/or metadata.
  - The layers are read-only.
  - The top layer is the writable layer.
- Images can share layers to optimize disk usage, transfer times, and memory use.
- Images are STATELESS and IMMUTABLE.
- Images are built from a Dockerfile. A Dockerfile is a text file that contains the instructions for building an image.
- The process of building a new image is called: "COMMITING A CHANGE".