Docker

Reporter

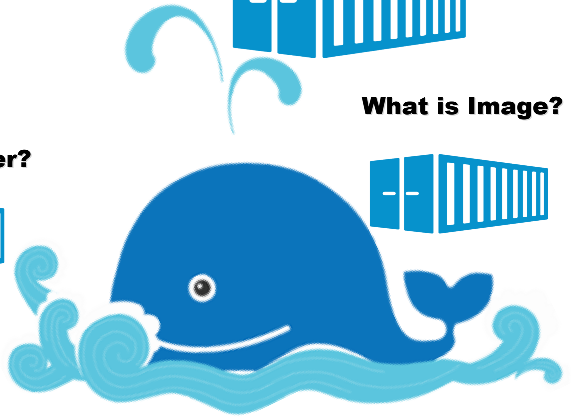
Andranik Barseghyan

https://github.com/Rebiss/Presentation

What is Docker?

When is it used?

What is Container?



What is Image?

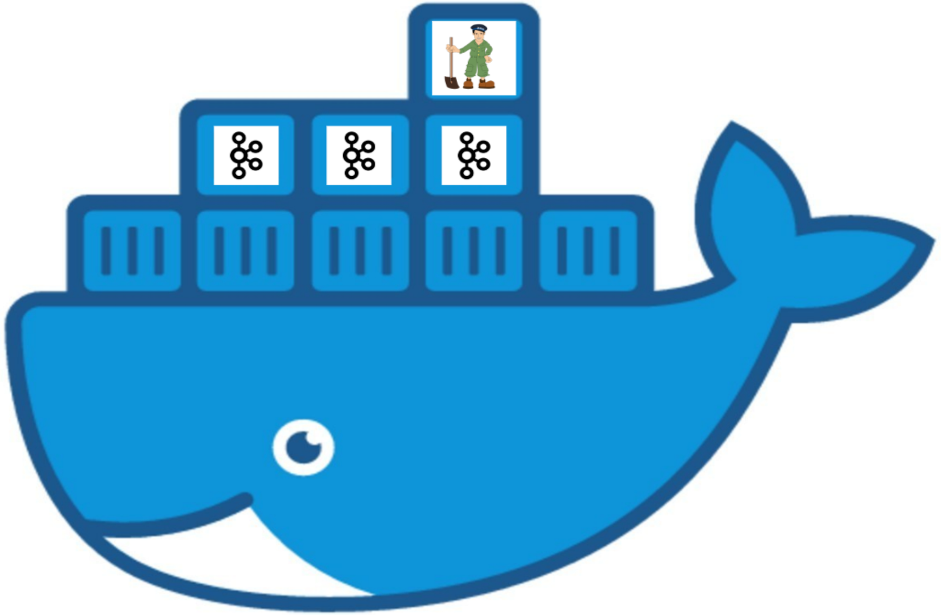
What is Docker-Compose?

Example

Containers vs Virtual Machines
What’s the Difference?

What is Docker?

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

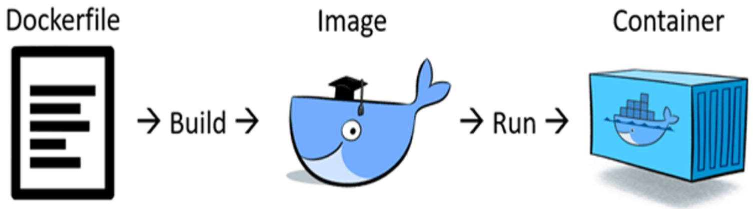


Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application.

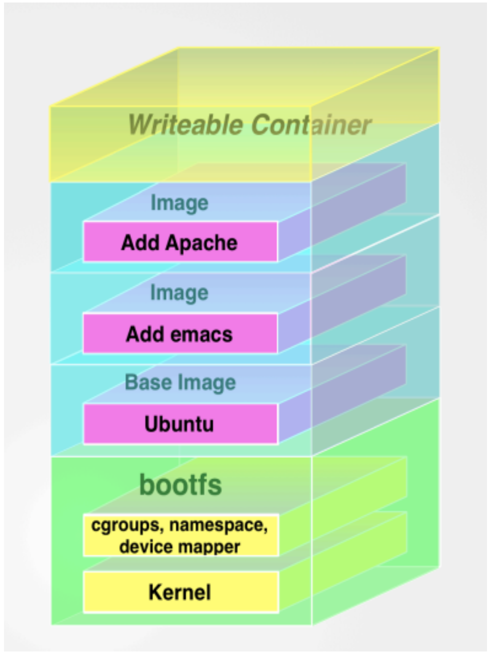
And importantly, Docker is open source. This means that anyone can contribute to Docker and extend it to meet their own needs if they need additional features that aren't available out of the box.

Image

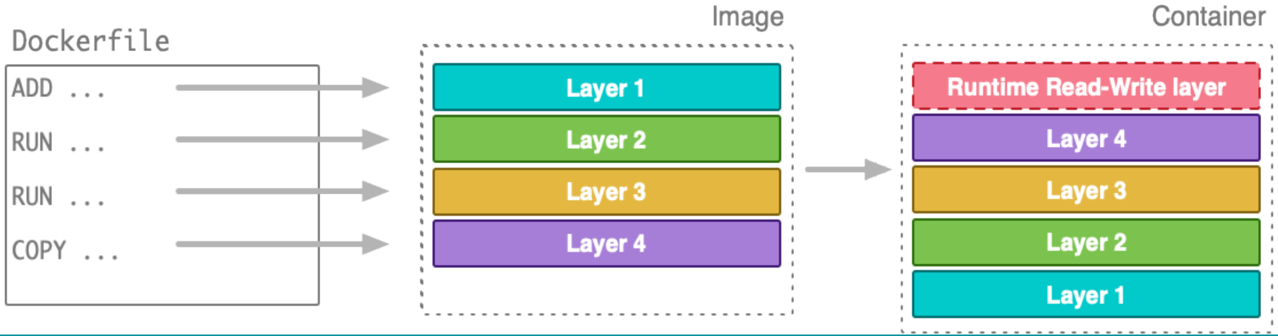
Image is a file, comprised of multiple layers, that is used to execute code in a Docker container. An image is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel. When the Docker user runs an image, it can become one or multiple instances of that container.



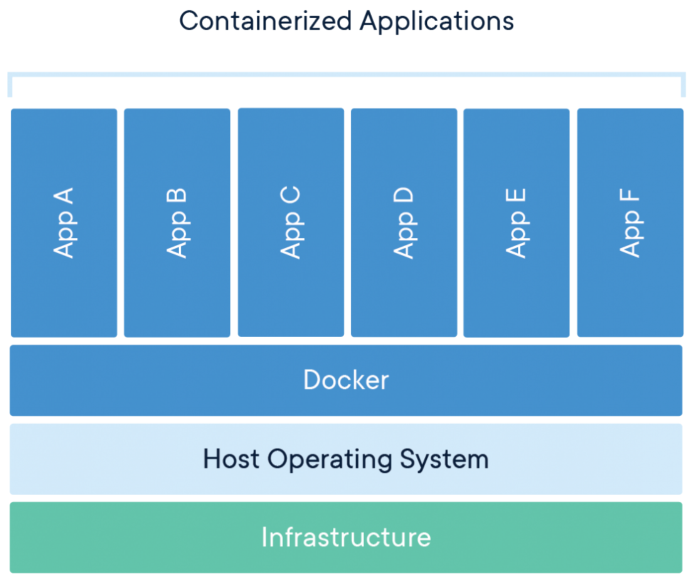
not monolith



Union File System



Container



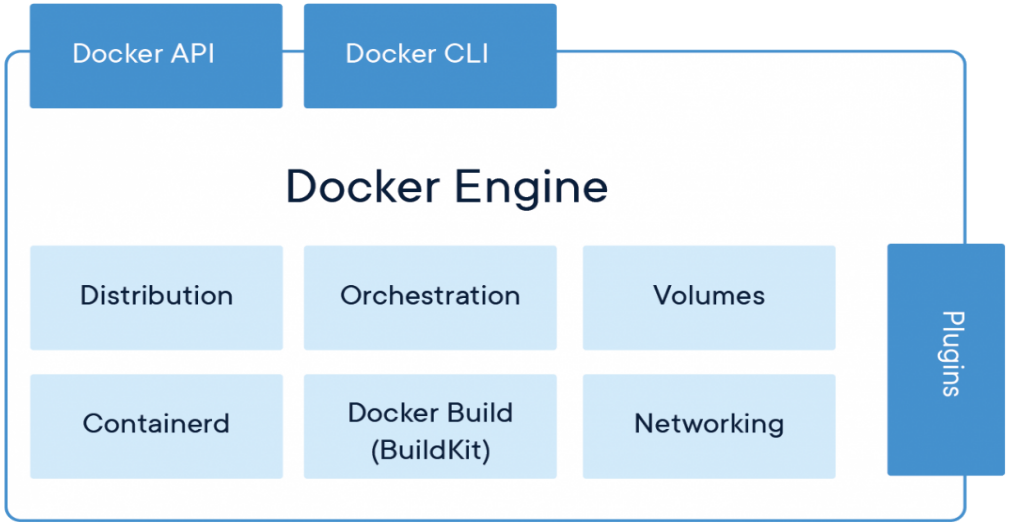
A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Container images become containers at runtime and in the case of Docker containers - images become containers when they run on Docker Engine. Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Docker containers that run on Docker Engine:

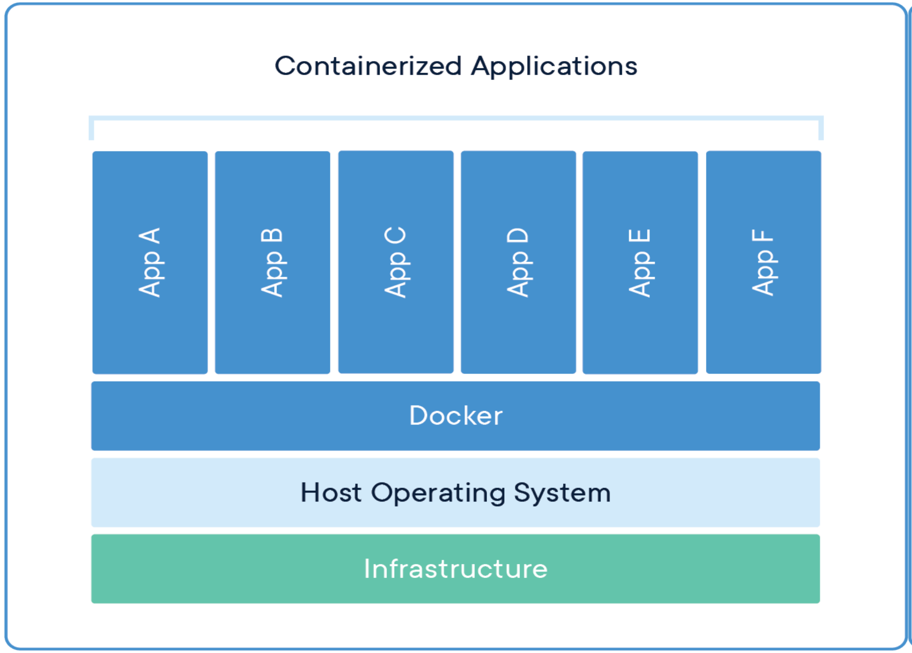
Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.

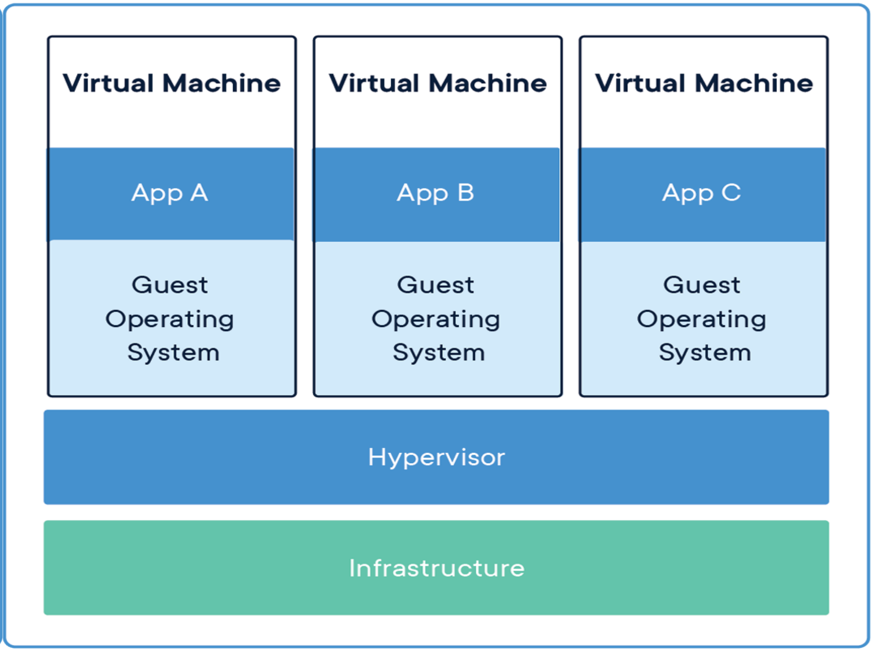
Docker Engine



Docker Engine is the industry's de facto container runtime that runs on various Linux (CentOS, Debian, Fedora, Oracle Linux, RHEL, SUSE, and Ubuntu) and Windows Server operating systems. Docker creates simple tooling and a universal packaging approach that bundles up all application dependencies inside a container which is then run on Docker Engine. Docker Engine enables containerized applications to run anywhere consistently on any infrastructure, solving "dependency hell" for developers and operations teams, and eliminating the "it works on my laptop!" problem.

Difference

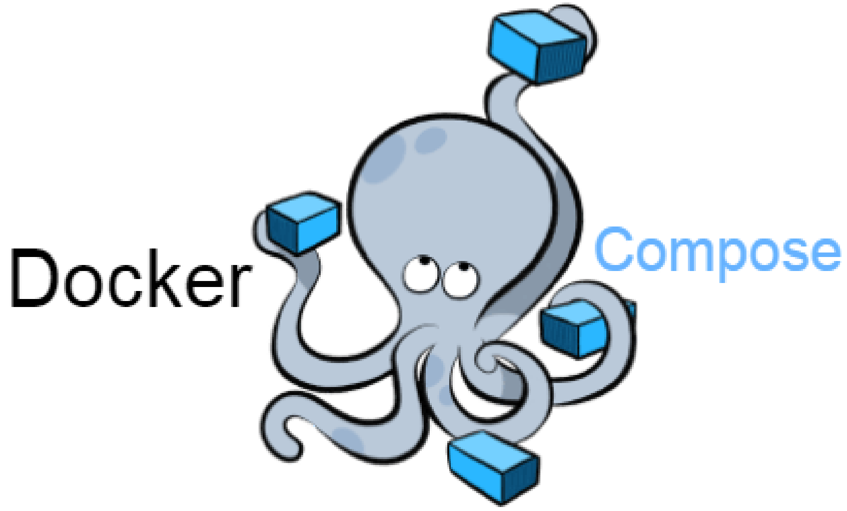




VIRTUAL MACHINES

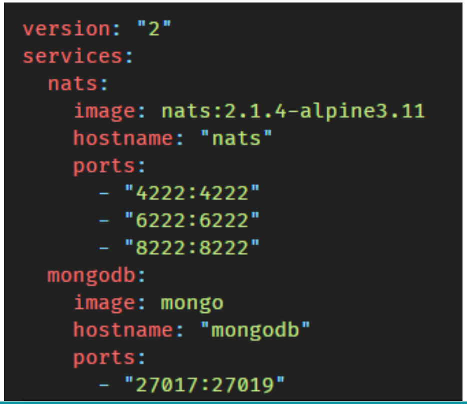
Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

Docker-Compose



Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. To learn more about all the features of Compose, see the list of features.

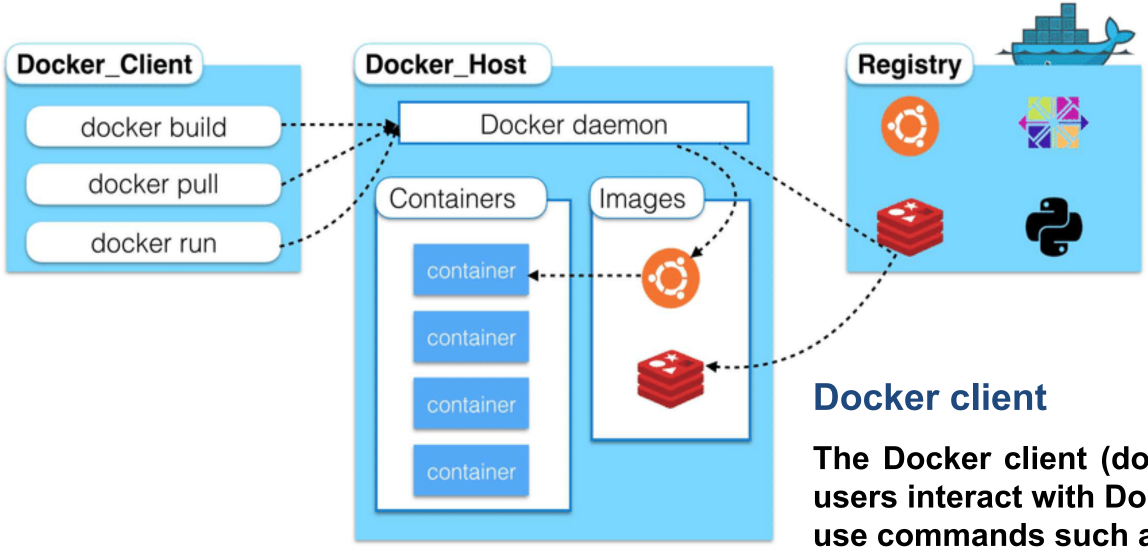
Compose works in all environments: production, staging, development, testing, as well as CI workflows. You can learn more about each case in Common Use Cases.



```yaml
version: "2"
services:
  nats:
    image: nats:2.1.4-alpine3.11
    hostname: "nats"
    ports:
      - "4222:4222"
      - "6222:6222"
      - "8222:8222"
  mongodb:
    image: mongo
    hostname: "mongodb"
    ports:
      - "27017:27019"
```

docker-compose.yml

Docker architecture



Docker daemon

The Docker daemon 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.

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 command uses the Docker API. The Docker client can communicate with more than one daemon.

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. You can even run your own private registry. If you use Docker Datacenter (DDC), it includes Docker Trusted Registry (DTR).

When is it used?

Use Docker as version control system for your entire app's operating system.

Use Docker when you want to distribute/collaborate on your app's operating system with a team.

Use Docker to run your code on your laptop in the same environment as you have on your server.

Use Docker whenever your app needs to go through multiple phases of development (dev/test/qa/prod, try Drone or Shippable, both do Docker CI/CD).

Use Docker with your Chef Cookbooks and Puppet Manifests (Docker doesn't do configuration management).

https://hub.docker.com

https://www.docker.com

Example ...

THANK YOU