Introduction to Docker

Mikhail Dozmorov

Summer 2018

Reproducibility: tools that enable reproducibility of analysis results

Tool	Website	Notes
Docker	docker.com	Packages software with all necessary components to enable reproducible deployment in different computing environments
Galaxy	galaxyproject.org	Cloud-based platform for analysis driven by reproducible workflows
Omics Pipe	sulab.org/tools/omics-pipe	Automates deployment of best-practice omics data analysis pipelines in the cloud
Singularity	singularity.lbl.gov	Similar to Docker but designed for scientific software running in high-performance computing environments

Docker is

- An open-source project to easily create lightweight, portable, self-sufficient containers from any application
- A tool for creating a layered filesystem; each layer is versioned and can be shared across running instances, making for much more lightweight deployments
- A company behind the project, as well as a site called the "Docker Hub" for sharing containers

Docker **is not** a virtual machine. Unlike a true VM, a docker container does not require a host OS, meaning it's much slimmer than a real VM

Docker timeline

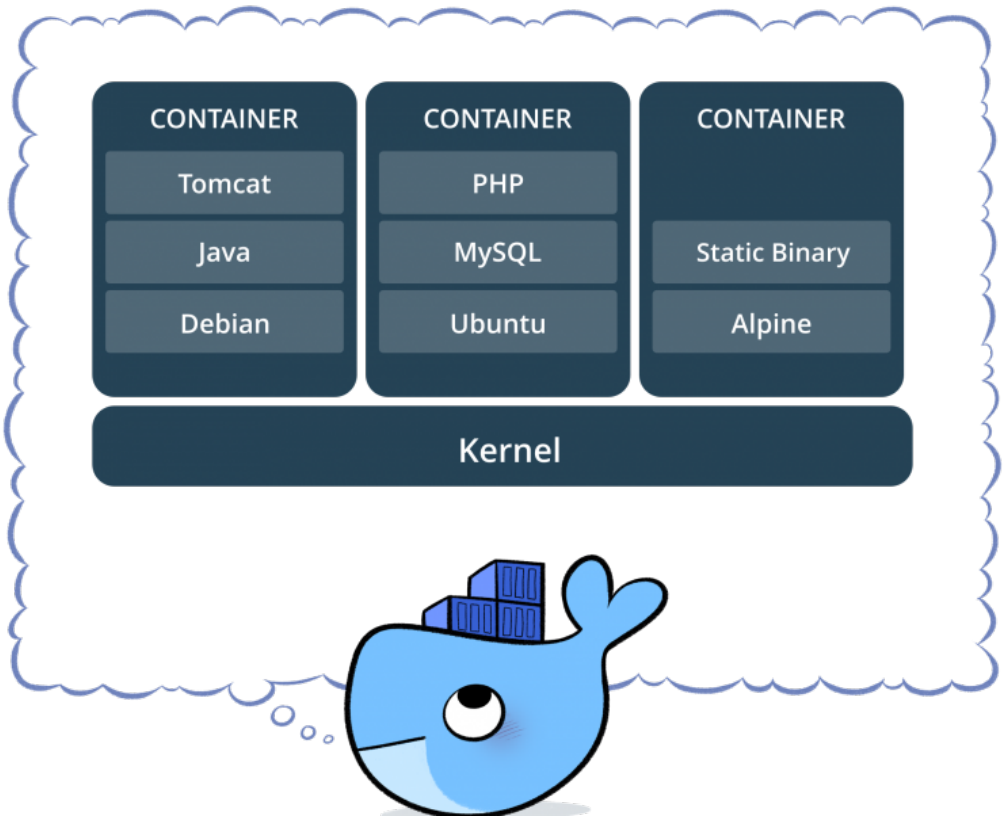
- January 2013 - First commit
- March 2013 - Docker 0.1.0 released
- April 2014 - Docker Governance Advisory Board announced with representation from IBM
- June 2014 - Docker 1.0 released
- September 2014 - $40 million investment round
- December 2014 - Docker and IBM announce strategic partnership
- February 2016 - Docker introduces first commercial product, now Docker Enterprise Edition
- Today - 3,300+ contributions, 43,000+ start, 12,000+ forks

https://github.com/docker

The Docker Family tree

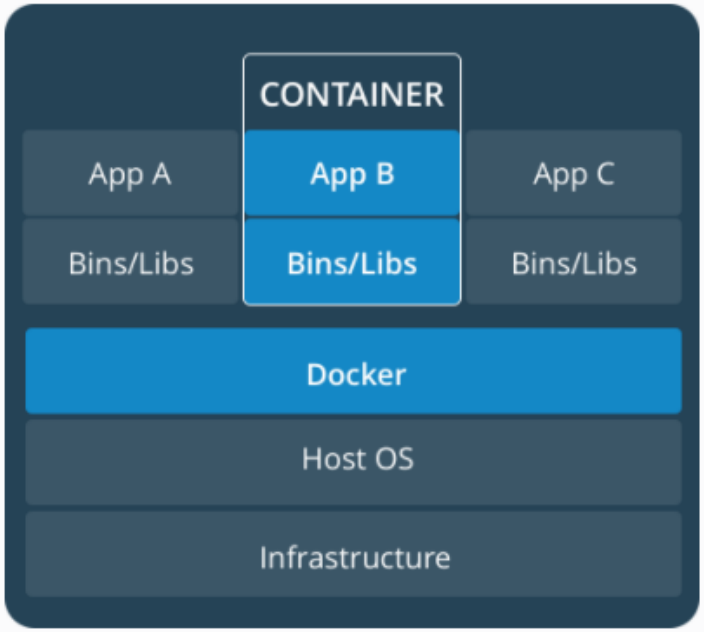
- `moby` project - open source framework for assembling core components that make a container platform. Intended for open souce contributors + ecosystem developers
  - `Docker Enterprise Edition` - subscription-based, commercially supported products for delivering a secure software supply chain. Intended for product deployment + enterprise customers
  - `Docker Community Edition` - free, community-supported product for delivering a container solution. Intended for software developers

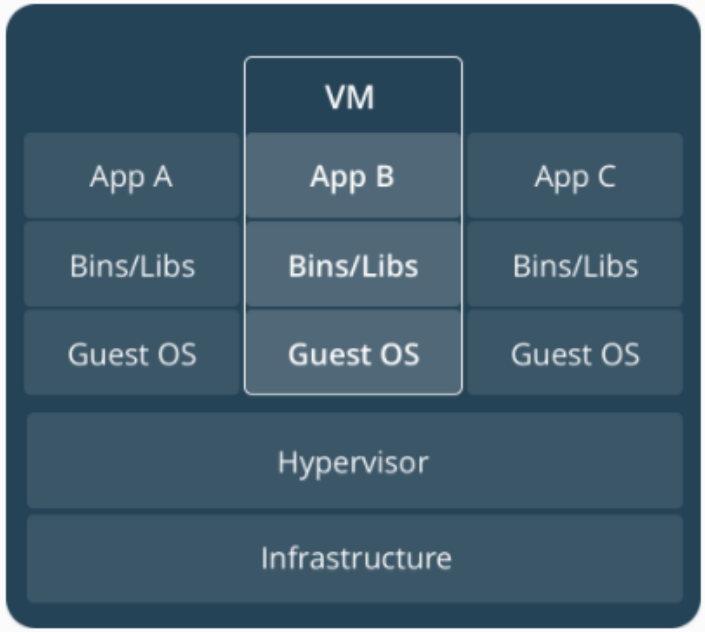
What is a container?



- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works with all major Linux and Windows Server

Container vs. VM





CONTAINERS

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), and start almost instantly.

VIRTUAL MACHINES

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

Key Benefits of Docker Containers

- **Speed** - No OS to boot = applications online in seconds
- **Portability** - Less dependencies between process layers = ability to move between infrastructure
- **Efficiency** - Less OS overhead, Improved VM density

Docker components

- (Docker) client
- daemon
- engine
- machine
- compose
- swarm
- registry

Docker components

- **Docker Daemon** - The background service running on the host that manages building, running and distributing Docker containers. The daemon is the process that runs in the operating system to which clients talk to
- **Docker Client** - The command line tool that allows the user to interact with the daemon. More generally, there can be other forms of clients too - such as `Kitematic` which provide a GUI to the users
- **Docker Hub** - A registry of Docker images

Docker components

- **Docker Machine** - Create Docker hosts on your computer, on cloud providers, and inside your own data center
- **Docker Compose** - A tool for defining and running multi-container Docker applications.
- **Docker Swarm** - A native clustering solution for Docker

Docker terminology for developers

- **Dockerfile** - build script that defines:
  - an existing image as the starting point
  - a set of instructions to augment that image (each of which results in a new layer in the file system)
  - meta-data such as the ports exposed
  - the command to execute when the image is run

Docker terminology for users

- **Image** - The blueprints of applications which form the basis of containers. Get them with `docker pull`
  - A snapshot of a filesystem at a certain point in time
  - The image is composed of layers that progressively stack on top of each other
  - Layers can be shared by running instances of an image

```
$ docker pull ubuntu
Pulling repository ubuntu
c4ff7513909d: Download complete
511136ea3c5a: Download complete
1c9383292a8f: Download complete
...
```

Docker container

- **Container** - runtime instance of an image plus a read/write layer. Create them with `docker run`, list running containers with `docker ps`
- A Docker container can be seen as a computer inside your computer
- It can send to your friends; and when they start this computer and run your code they will get exactly the same results as you did

What if I want to change an image?

- An image is read-only, how do we change it?
- We don't
- We create a new container from that image
- Then we make changes to that container
- When we are satisfied with those changes, we transform them into a new layer
- A new image is created by stacking the new layer on top of the old image

Safety in Docker

- Is it safe to run applications in containers?
- Can one container break out, and into another?
- How isolated are containers?

Safety in Docker

- The docker user is `root` equivalent
- It provides root-level access to the host
- You should restrict access to it like you would protect root
- If you give somebody the ability to access the Docker API, you are giving them full access on the machine
- Therefore, the Docker control socket is (by default) owned by the docker group, to avoid unauthorized access on multi-user machines

Safety in Docker

- Add the Docker group

```
$ sudo groupadd docker
```

- Add ourselves to the group

```
$ sudo gpasswd -a $USER docker
```

- Restart the Docker daemon

```
$ sudo service docker restart
```

- Log out

```
$ exit
```

Image provenance

- How can I trust `docker pull` an image?
  - Must trust the upstream image, Docker Hub, the transport between the Hub and my Docker host
- If you don't trust upstream, don't use `apt-get` and `yum`, verify all source code + changes, compile everything from source
- If you don't trust Docker, audit the whole Docker Engine code
- If you don't trust transport, learn the protocol Docker uses to distribute signed content
- Be reasonable

Immutable containers

- `docker run --read-only` - makes it impossible to entrench in a container
- Even without `--read-only` flag:
  - copy-on-write prevents changes from being permanent
  - break a container - it gets recycled
  - docker diff allows easy audit of changes
- Many more other security hardening features are getting developed

https://www.docker.com/docker-security

Docker quick start

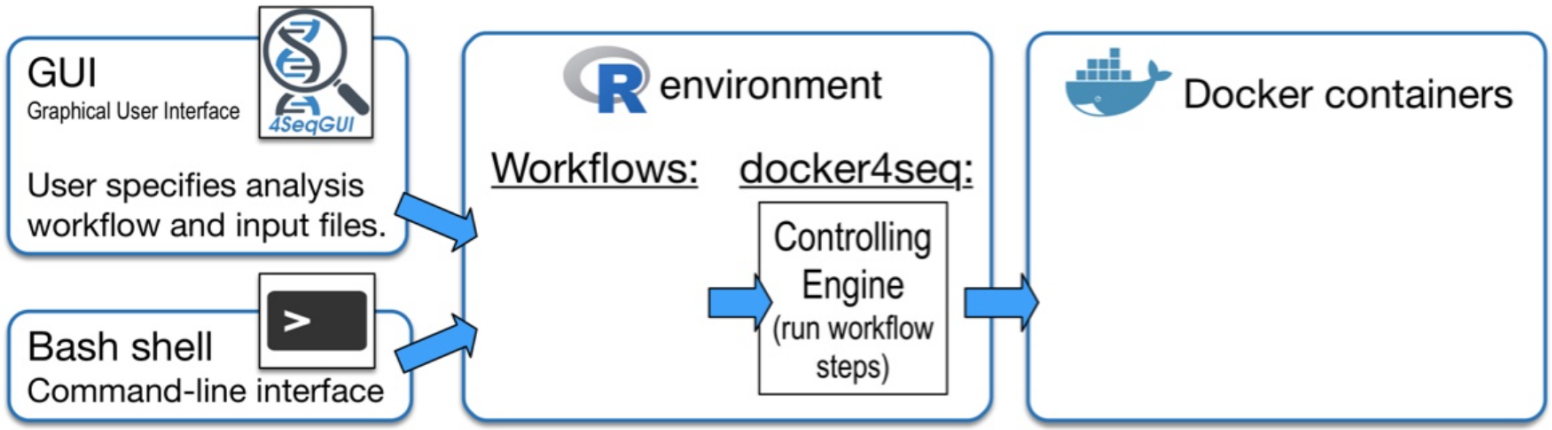
- Linux - runs natively, e.g., on Ubuntu
  - `sudo apt-get install docker.io`
  - `curl -sSL https://get.docker.com/ubuntu/ | sudo sh`
- Windows/Mac - native Docker apps
- Anywhere (old systems, remote hosts) - Docker Machine

https://docs.docker.com/docker-for-mac/

https://docs.docker.com/docker-for-windows/

https://docs.docker.com/machine/overview/

Reproducible Bioinformatics project



- Based on Docker images and an R package.
- Three modules:
  1. `docker4seq` R package (https://github.com/kendomaniac/docker4seq)
  2. dockers images (https://hub.docker.com/u/repbioinfo/)
  3. `4SeqGUI` (https://github.com/mbeccuti/4SeqGUI)

Kulkarni, Neha, Luca Alessandri, Riccardo Panero, Maddalena Arigoni, Martina Olivero, Francesca Cordero, Marco Beccuti, and Raffaele A Calogero. "Reproducible Bioinformatics Project: A Community for Reproducible Bioinformatics Analysis Pipelines." BioRxiv, January 1, 2017. https://doi.org/10.1101/239947.

http://reproducible-bioinformatics.org/

Docker on Amazon Web Services

- **AWS Elastic Beanstalk** is an easy-to-use PaaS (Platform as a Service) for deploying and scaling web applications and services
  - Comes with reasonable defaults, easy to set up
- **Amazon Elastic Container Service (Amazon ECS)** is a highly scalable, high-performance container orchestration service that supports Docker containers and allows you to easily run and scale containerized applications on AWS
  - Flexible customization, can be complex to start with

https://aws.amazon.com/elasticbeanstalk/

https://aws.amazon.com/ecs/

https://docker-curriculum.com/

More resources

- Container engines
  - `rkt` - A security-minded, standards-based container engine
  - `Singularity` - scientific-oriented Docker images that can be run without superuser privileges
- Image repositories
  - `Biocontainers` - bioinformatics software containers
  - `Quay` - secure container storage
  - `Dockstore` - platform for sharing workflows and pipelines as Docker images

https://coreos.com/rkt/

https://singularity.lbl.gov/

http://biocontainers.pro/

https://quay.io/

https://dockstore.org/