Docker Concepts

Cloud Tech BLR

Agenda

What is Docker?

- Docker vs. Virtual Machine
- History, Status, Run Platforms
- Hello World

Images and Containers

Volume Mounting, Port Publishing, Linking

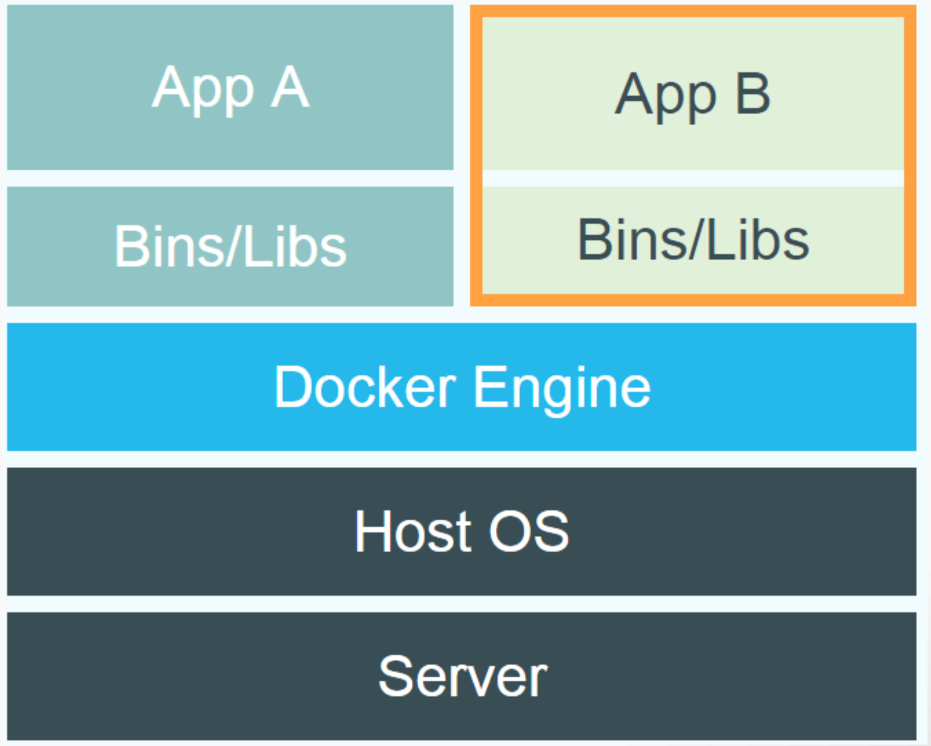
Around Docker, Docker Use Cases

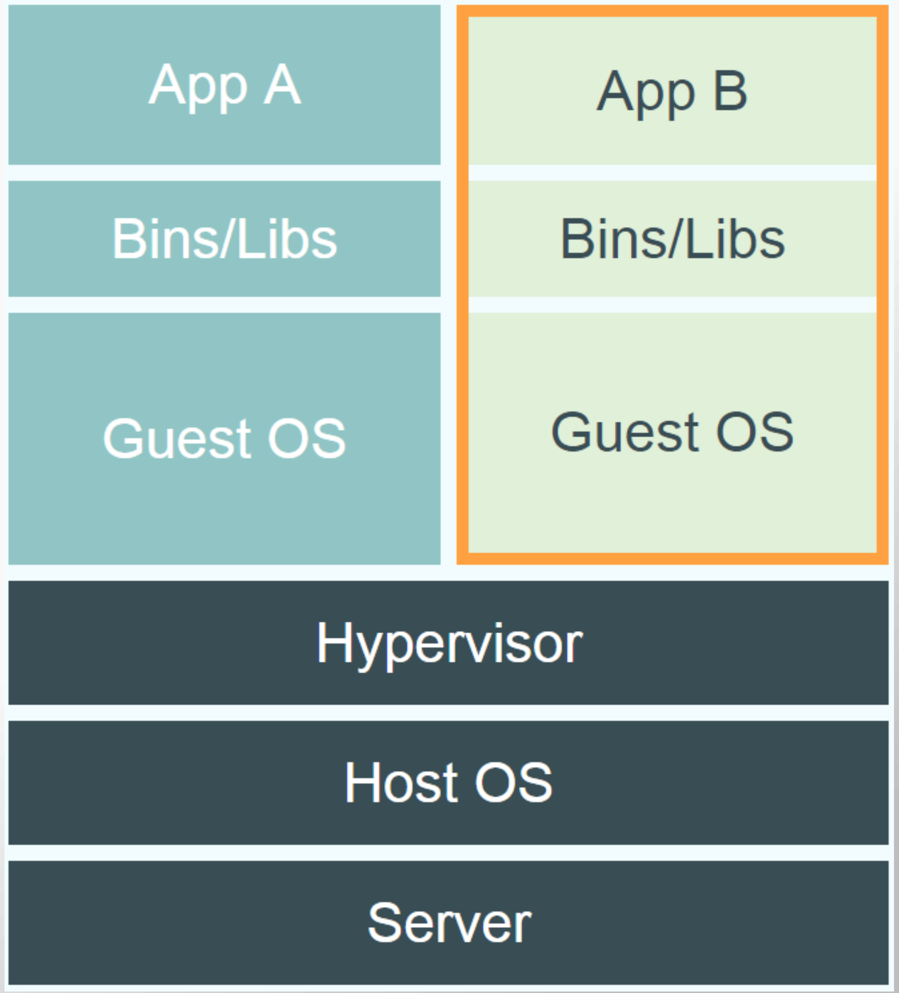
Hands-On Workshop

What is Docker?

Docker is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating system–level virtualization on Linux.

Docker vs. Virtual Machine



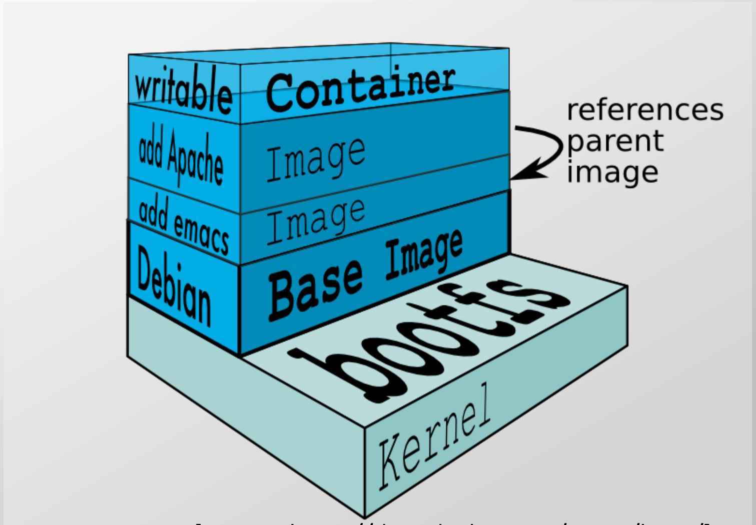


Docker Technology

libvirt: Platform Virtualization

LXC (LinuX Containers): Multiple isolated Linux systems (containers) on a single host

Layered File System



[Source: https://docs.docker.com/terms/layer/]

Docker History

2013-03: Releases as Open Source

2013-09: Red Hat collaboration (Fedora, RHEL, OpenShift)

2014-03: 34th most starred GitHub project

2014-05: JAX Innovation Award (most innovative open technology)

Run Platforms

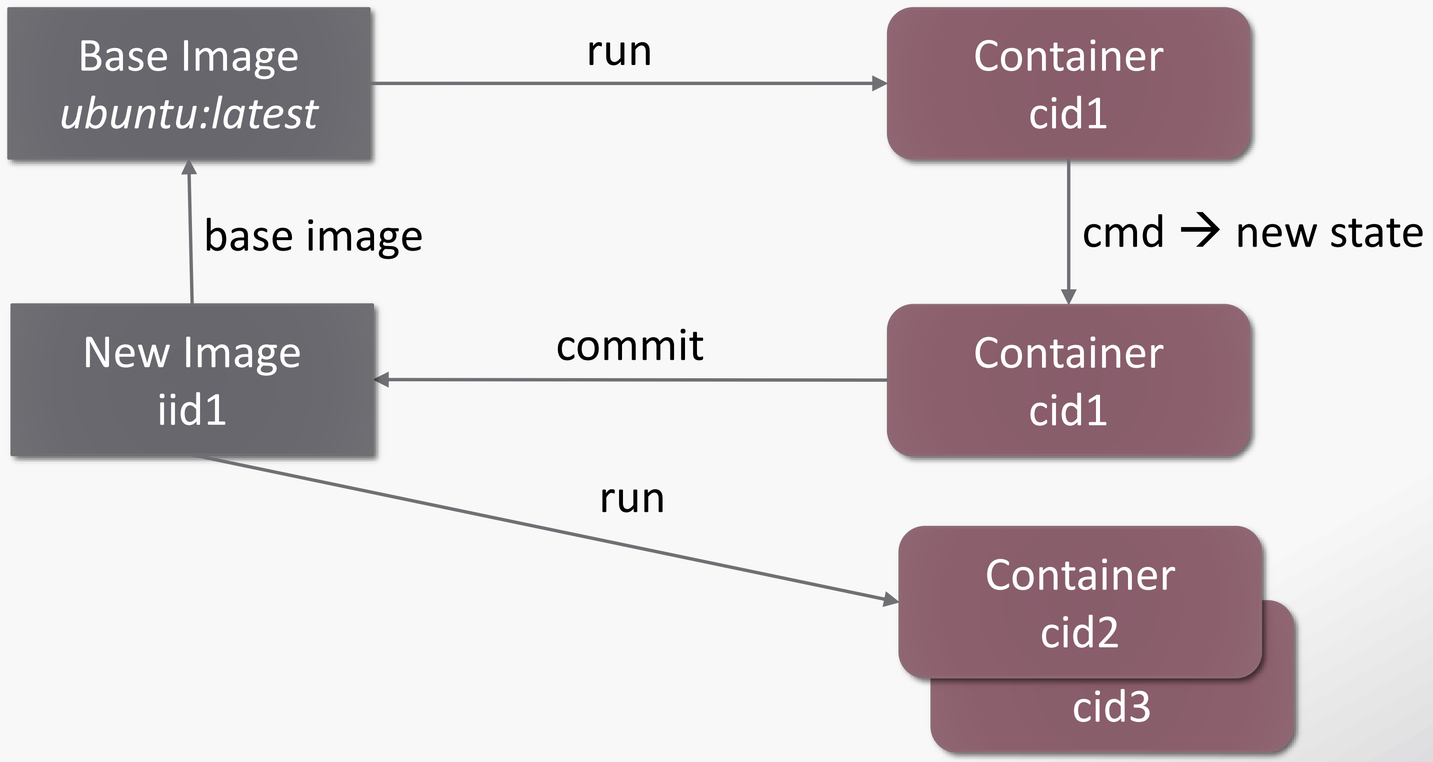
- ➢Various Linux distributions (Ubuntu, Fedora, RHEL, Centos, openSUSE, ...)
- ➢Cloud (Amazon EC2, Google Compute Engine, Rackspace)
- ➢2014-10: Microsoft announces plans to integrate Docker with next release of Windows Server

Hello World

Simple Command - Ad-Hoc Container

```
 docker run ubuntu echo
Hello World
  -- docker images [-a]
  -- docker ps -a
```

Image vs. Container



Terminology - Image

Persisted snapshot that can be run

# docker images ls

- *images:* List all local images
- *run*: Create a container from an image and execute a command in it
- *tag*: Tag an image
- *pull*: Download image from repository
- *rmi*: Delete a local image
  - This will also remove intermediate images if no longer used

Publish Port

```
 docker run -t -p 8080:80 ubuntu
nc -l 80
```

-- Map container port 80 to host port 8080

-- Check on host: nc localhost 8080

Link with other docker container

```
- docker run -ti --link
containerName:alias ubuntu
```

-- See link info with `set`

Terminology - Container

- Runnable instance of an image

  # docker ps

  -- ps: List all running containers

  # docker ps -a

  -- *ps –a*: List all containers (incl. stopped)

  # docker top ContainerName/ID

  -- *top*: Display processes of a container

  # docker start ContainerName/ID

  -- *start*: Start a stopped container

  # docker stop ContainerName/ID

  -- *stop*: Stop a running container

Terminology - Container

- Runnable instance of an image

  # docker pause ContainerName/ID

  -- *pause*: Pause all processes within a container

  -- *unpause:* resumes the container

  # docker stats ContainerName/ID

  --*stats:*

  # docker rm ContainerName/ID

  -- *rm*: Delete a container

  # docker rm ContainerName/ID

  -- *kill*: kills the container abruptly

  # docker commit ContainerName/ID

  -- *commit*: Create an image from a container

Difference between stop vs kill

- ***docker kill*** will stop the main entrypoint process/program abruptly
- ***docker stop*** will try to stop it gracefully (will ask politely :P)

By running **docker events** after **docker stop** shows events

```
kill (signal 15): where signal 15 = SIGTERM
die
stop
```

By running **docker events** after **docker kill** shows events

```
kill (signal 9): where signal 9 = SIGKILL
Die (exit Code 137)
```

Docker Hub

Public repository of Docker images

- https://hub.docker.com/
- docker search [term]

Automated: Has been automatically built from Dockerfile

- Source for build is available on GitHub

Dockerfile

- Create images automatically using a build script: «Dockerfile»
- Can be versioned in a version control system like Git or SVN, along with all dependencies
- Docker Hub can automatically build images based on dockerfiles on Github

Dockerfile Example

Dockerfile:

```
FROM ubuntu
ENV DOCK_MESSAGE Hello My World
ADD dir /files
CMD ["bash", "someScript"]
```

```
docker build [DockerFileDir]
```

```
docker inspect [imageId]
```

Mount Volumes

```
docker run -ti -v /hostLog:/log ubuntu
```

Run second container: Volume can be shared

-
  ```
  docker run -ti --volumes-from firstContainerName ubuntu
  ```

Docker Use Cases

- Development Environment
- Environments for Integration Tests
- Quick evaluation of software
- Microservices
- Multi-Tenancy
- Unified execution environment (dev → test → prod (local, VM, cloud, ...)