Docker Day 2

Welcome

SAR INFOTECH
Making Intellectuals Meet Excellence

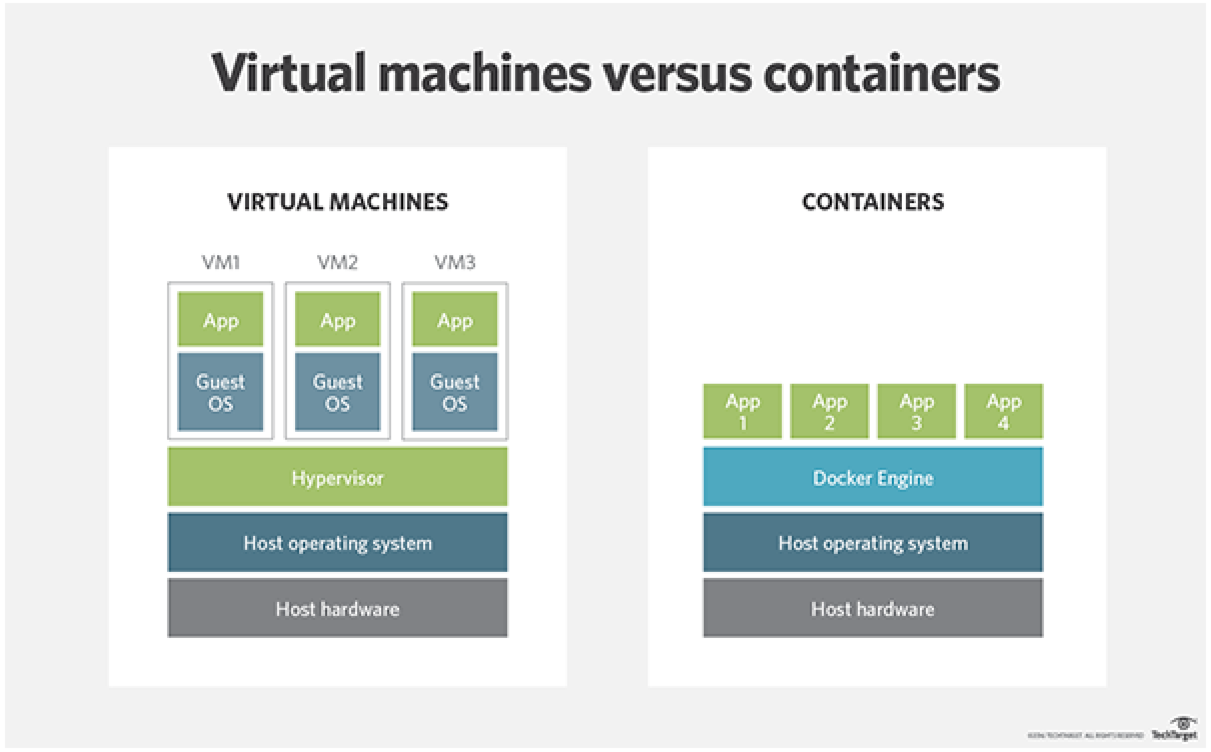
Safe Harbour

- All images used are taken with consensus for this education purpose and does not associate with any obligation of Intellectual Property.

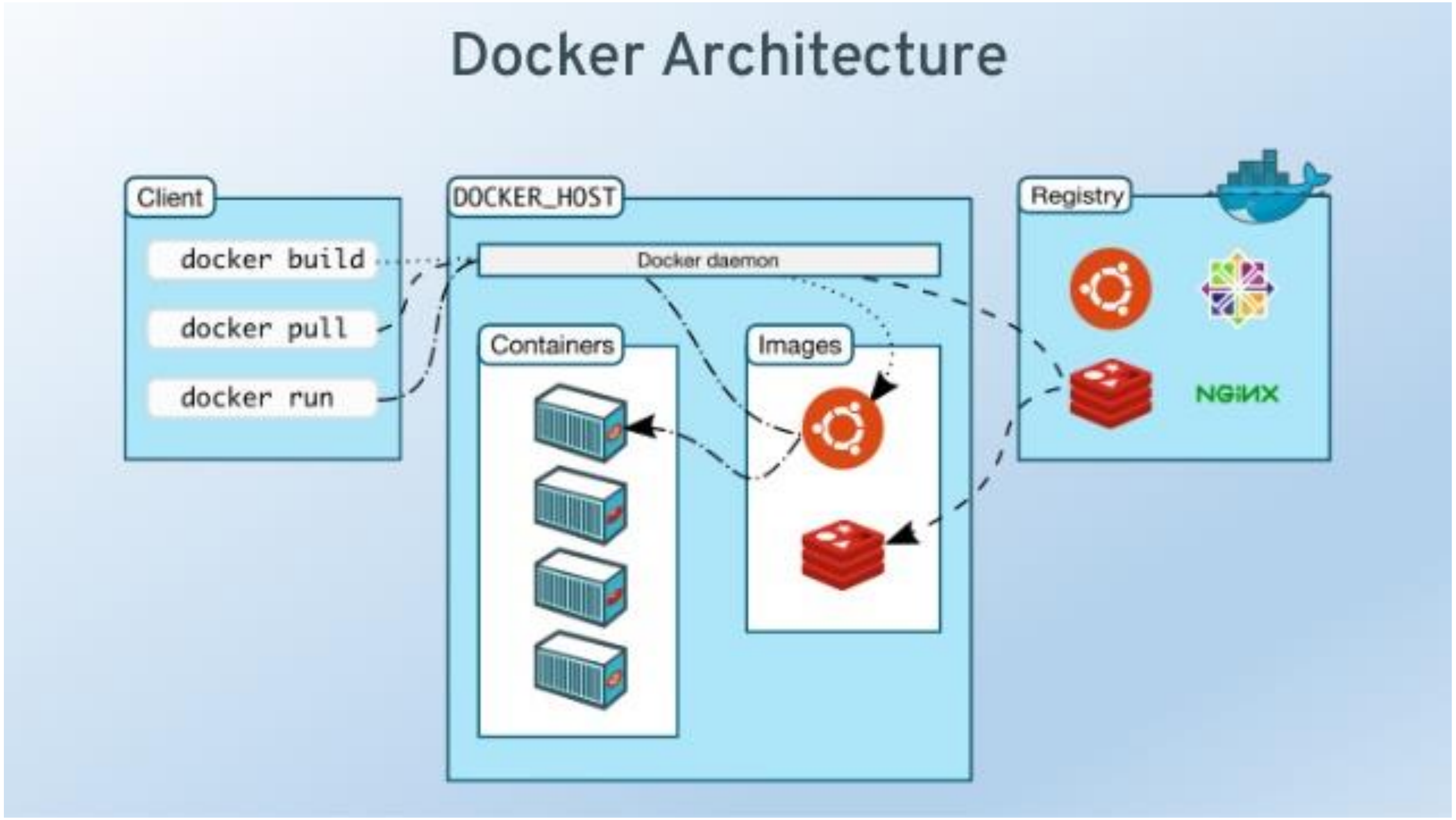
SAR INFOTECH
Making Intellectuals Meet Excellence

Co-existence

- if you need to run multiple applications on multiple servers, it probably makes sense to use VMs. On the other hand, if you need to run many *copies* of a single application, Docker offers some compelling advantages.

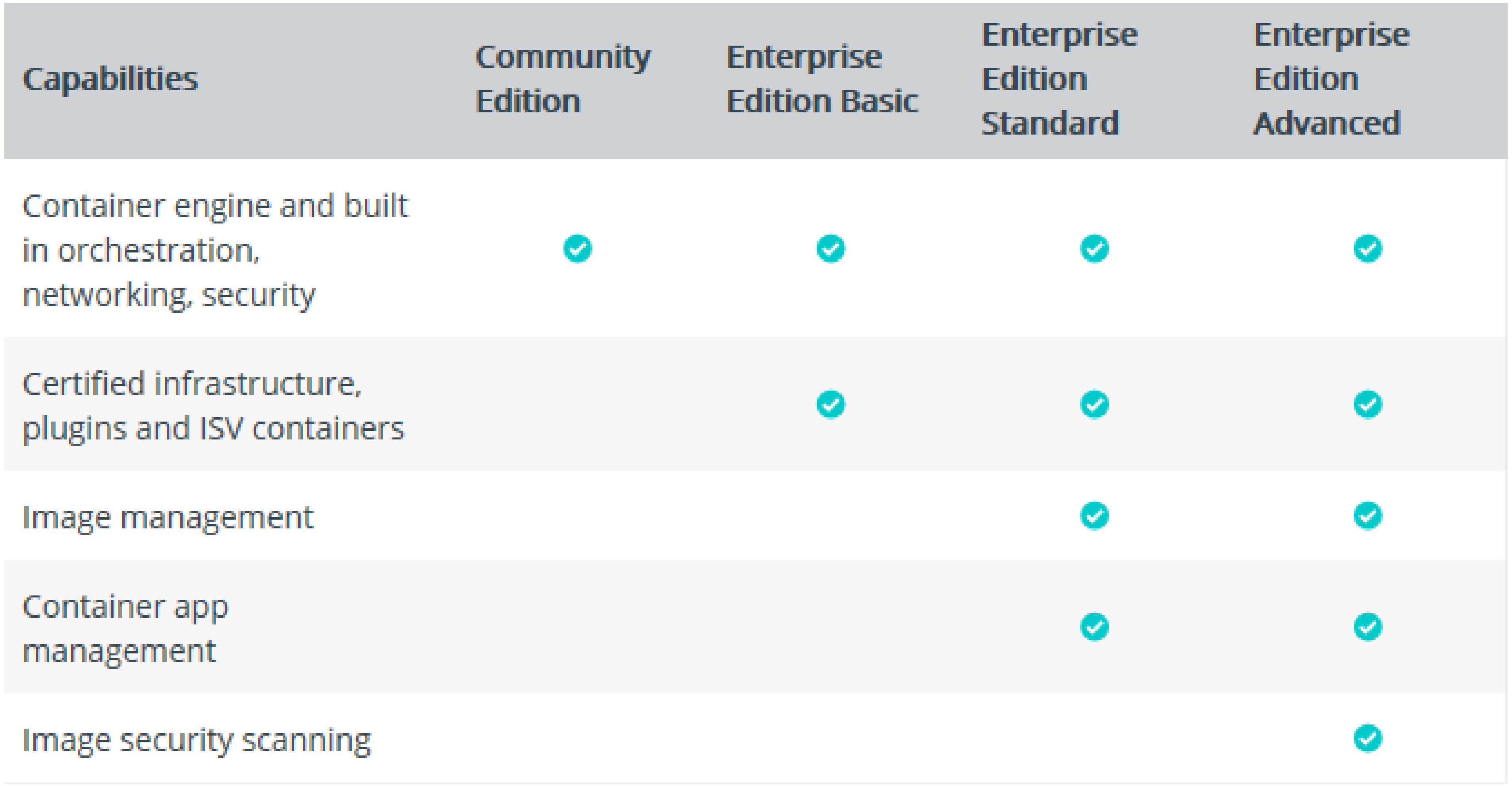


Architecture



SAR INFOTECH
Making Intellectuals Meet Excellence

Docker Software



Capabilities	Community Edition	Enterprise Edition Basic	Enterprise Edition Standard	Enterprise Edition Advanced
Container engine and built in orchestration, networking, security	✓	✓	✓	✓
Certified infrastructure, plugins and ISV containers		✓	✓	✓
Image management			✓	✓
Container app management			✓	✓
Image security scanning				✓

Day 2

- Docker file
- Registry
- Copy on Write Strategy
- Volumes
- Environment Files

SAR INFOTECH
Making Intellectuals Meet Excellence

Docker file

- **Docker** can build images automatically by reading the instructions from a **Docker file** . A **Docker file** is a text document that contains all the commands a user could call on the command line to assemble an image.

- The build is run by the Docker daemon, not by the CLI. The first thing a build process does is send the entire context (**recursively**) to the daemon.

Preferences of Dockerfile

- .dockerignore
  - */tmp*
  - !readme.tmp
  - *.md
- Dockerfile
  - No Root directory
  - User defined Directory (Preferred)

Commands In Docker

- ARG
- # comment
- # escape=` (default \)
- MAINTAINER
- FROM image:tag as newIMg
- WORKDIR
- USER

ENV and Value

- ENV fname /bar
- ENV descpn="This is Text File available"
- ENV abc=hello
- ENV abc=bye def=$abc
- ENV ghi =$abc

  (Predict values of ghi and abc)

- {}, $()
  - $(fname)
  - ${fname}
  - WORKDIR $fname

RUN command

- 2 Forms
- Form 1 – Shell Form Run <Command>
  - RUN /bin/bash –c ‘echo $hello’
- Form 2 – Exec Form Run [“Exe”,”Param1”,”Param2”] RUN[“/bin/bash”,”-c”,”echo hello”]
- RUN pwd

SAR INFOTECH
Making Intellectuals Meet Excellence

CMD Command

- Executable Form
  - CMD[“exe”,”Parm1”,”Parm2”]
- Shell Form
  - CMD Command Param1 Param2

- Default to ENTRYPOINT
  - CMD[“Param1”,”PARAM2”]

  - CMD [“echo”,”$HOME”]
  - CMD [“sh”,”-c”,”echo $HOME”]

  - CMD echo This is a line of text | wc –
  - CMD [“/bin/wc”,”-c”,”echo this is a line of text”]

ENTRYPOINT

- --entrypoint or in Dockerfile
- Two Forms
  - ENTRYPOINT Command , PARAM1
  - ENTRYPOINT [“Cmd”, “PARAM1”]

  - Entry point treats Container as Executable
  - Default values for entry point can come via CMD
  - CMD can be overridden

COPY / ADD

Copy files from a specific location into a Docker image.

COPY takes in a *src* and *destination*. It only lets you copy in a local file or directory from your host (the machine building the Docker image) into the Docker image itself.

ADD lets you do that too, but it also supports 2 other sources. First, you can use a URL instead of a local file / directory. Secondly, you can extract a tar file from the source directly into the destination.

SHELL

The SHELL instruction is particularly useful on Windows where there are two commonly used and quite different native shells: cmd and PowerShell, as well as alternate shells sh.

The SHELL instruction can appear multiple times. Each SHELL instruction overrides all previous SHELL instructions, and affects all subsequent instructions.

```
# Executed as cmd /S /C echo default
RUN echo default
# Power shell
SHELL ["powershell", "-command"]
RUN Write-Host hello

# Executed as cmd /S /C echo hello
SHELL ["cmd", "/S", "/C"]
RUN echo hello
```

SAR INFOTECH
Making Intellectuals Meet Excellence

Docker build

- Build an image from Dockerfile
- --tag –t
- --target
- -no-cache
- --compress
- --cpu-period
- --cpu-quota
- --network
- --memory

BUILD and RUN

Lets Create Container

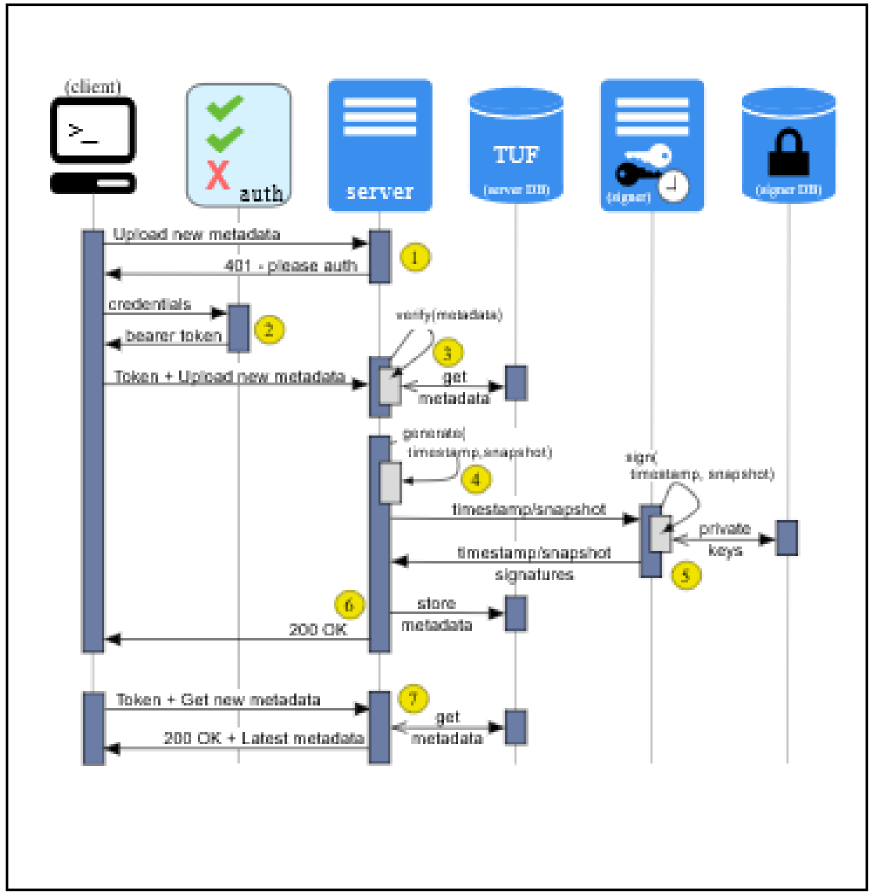
- Step 1 – Creation of Directory , cd? Why Not Root Directory?
- Step 2 - Create Docker File Creation
- Step 3 – Build docker through
  docker build –t <name _of _image> . (default current directory) – f <filename> .
- Step 4 docker build –t < > .
- Step 5 docker run <>

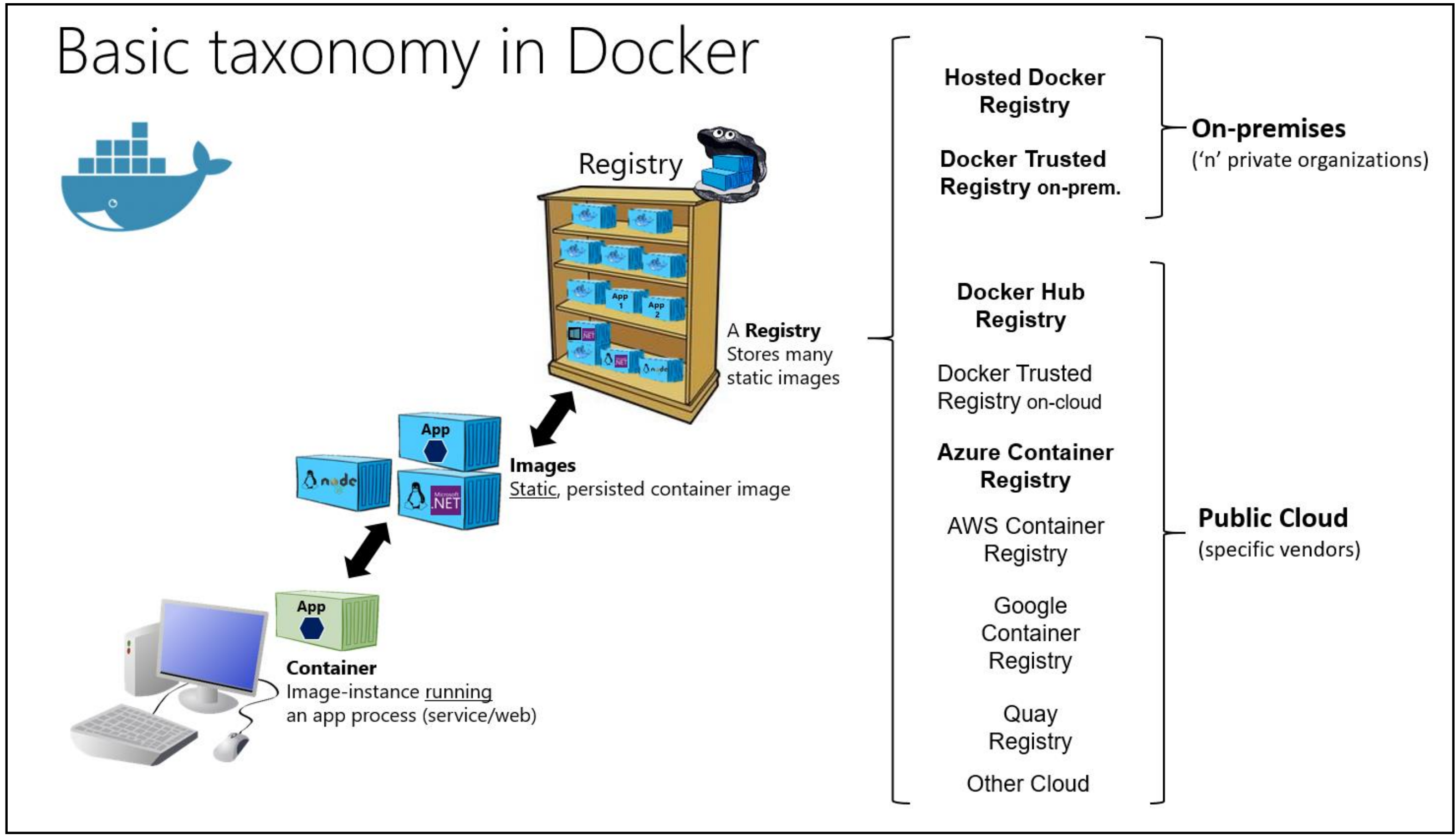
REGISTRY

- A Registry is a hosted service containing repositories of images which responds to the Registry API.
- The Registry is a stateless, highly scalable server side application that stores and lets you distribute Docker images.
- use the Registry if you want to:
  - tightly control where your images are being stored
  - fully own your images distribution pipeline
  - integrate image storage and distribution tightly into your in-house development workflow

NOTARY

- Signed Images




Basic taxonomy in Docker
Registry
A **Registry** Stores many static images
App
App 1
App 2
node
.NET
Images
Static, persisted container image
Container
Image-instance running an app process (service/web)
Hosted Docker Registry
Docker Trusted Registry on-prem.
On-premises
('n' private organizations)
Docker Hub Registry
Docker Trusted Registry on-cloud
Azure Container Registry
AWS Container Registry
Google Container Registry
Quay Registry
Other Cloud
Public Cloud
(specific vendors)

Registry Example

- Step 1

Docker search –no-trunc –s1000 registry
docker run –d –p 5000:5000 –restart=always --name registry registry:2

Step 2
Docker pull ubuntu:16.04

Step 3
Docker tag ubuntu:16.04 localhost:5000/tag2registry
Docker push localhost:5000/tag2registry

Step 4
Docker image remove ubuntu:16.04
Docker image remove localhost:5000/tag2registry
Docker pull localhost:5000/tag2registry

SAR INFOTECH
Making Intellectuals Meet Excellence

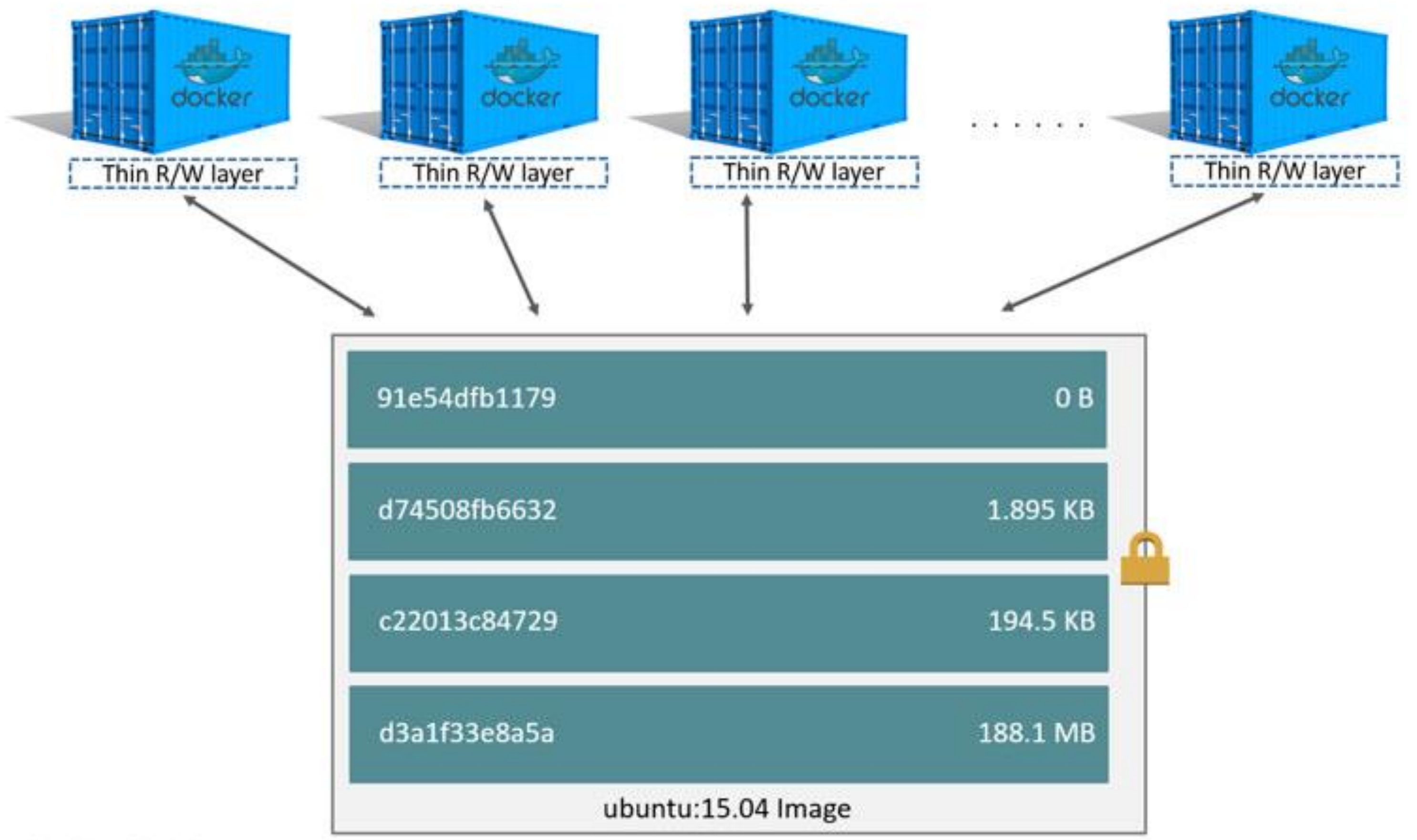
Check for registry online

- http://192.168.99.100:5000/v2/_catalog
- Docker image ls

- docker run -d -e REGISTRY_HTTP_ADDR=0.0.0.0:5001 -p 5001:5001 --name testreg1 registry:2

SAR INFOTECH
Making Intellectuals Meet Excellence

C-O-W Strategy



Docker ps –a

Size
Virtual Size (Thin Layer + SIZE)

Docker history <container>

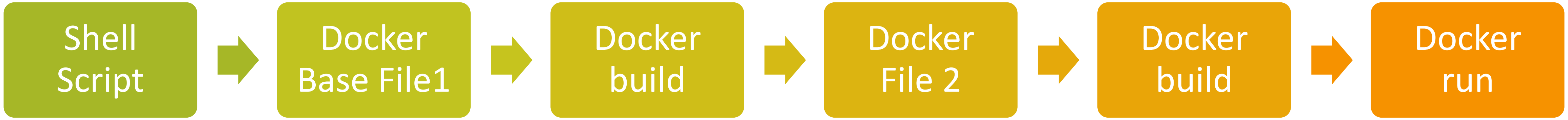
Copy on Write Registry

- The first time another layer needs to modify the file (when building the image or running the container), the file is copied into that layer and modified. This minimizes I/O and the size of each of the subsequent layers

Layered Docker

- Nested Docker file
- Intermediate containers
- Dangling Images
- Storage Drivers
  - AUFS, OVERLAY, OVERLAY2 (File)
  - BTRFS, DEVICEMAPPER,ZFS (Block)
  - Docker inspect <image>
  - Docker info
  - Docker container <> (Layers ?)

Now Steps for Layered Approach



#! /bin/sh

FROM
COPY

FROM <BF>
RUN
CMD

Play with Layers

- RUN and CMD – Changes in Dockerbase
- RUN base images and Final Images
- Docker history

Environment Variables

- Docker lets you store data such as configuration settings, encryption keys, and external resource addresses in environment variables. Docker Cloud makes it easy to define, share, and update the environment variables for your services.

- ENV
- -e
- --env-file

ENV in Dockerfile

- FROM ubuntu:latest
- ENV avar DHANANJAYAN

- Docker build –t mynewimg .
- Docker run –it mynewimg /bin/bash

  # echo $avar

  Lets try this → Docker run –it mynewimg echo $avar
  What result ?

OVERRIDING ENV with -e

- Docker run –it

docker run -it -e avar=DJ imglist/myimg1.0 /bin/bash

# echo $avar

- Docker exec –it avar=DHANAN <container ID> /bin/bash
- Echo $avar

- Docker exec –it avar=DHANAN <container ID> echo /bin $avar
  What happens here ?

SAR INFOTECH
Making Intellectuals Meet Excellence

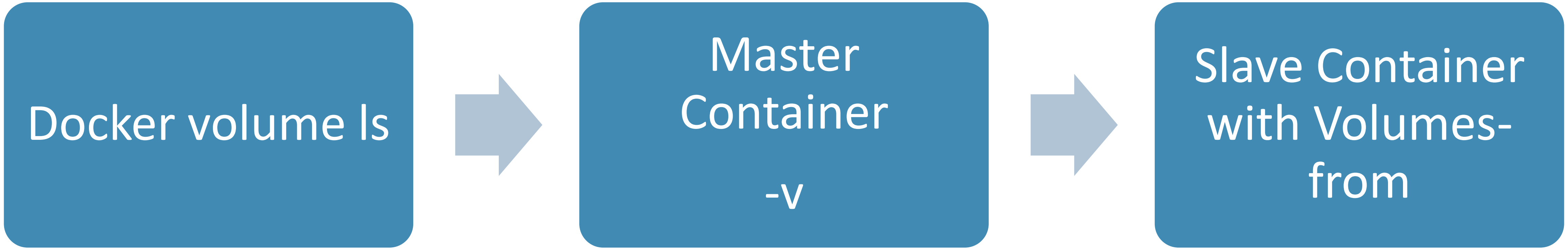
ENVIRONMENT FILE

- Envfile.txt
  - VAR1=VALUE
  - VAR2=VALUE
- docker run -it --env-file=envfile.txt ubuntu bash
- # echo $VAR1
- docker start
- docker exec -it -e CLIENT=ORA1 7f0d bash
  - echo $CLIENT

VOLUMES

- A volume is a specially-designated directory within one or more containers that bypasses the Union File System. Volumes are designed to persist data, independent of the container's life cycle.
- Docker therefore never automatically delete volumes when you remove a container, nor will it "garbage collect" volumes that are no longer referenced by a container
- Types
  - Host (FS)
  - Named (Disk)
  - Anonymous
- Driver
  - Flocker
  - local

Volume Creation – Type 1



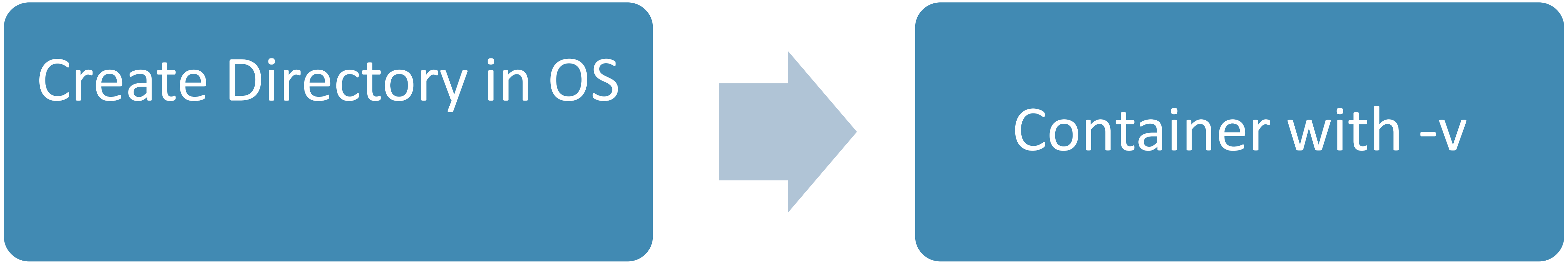
Docker run –it –name mastervol-data – v **bddata:**/bddata ubuntu /bin/bash

# cd data
Create files and directories.

Docker inspect bddata

Docker run –it –name slave_from_master –volumes-from mastervol-data ubuntu /bin/bash

Volume Creation – Type 2



```
docker run -dit --name devmtest1 -v ~/data:/app ubuntu /bin/bash
```