Hands-On Docker and Kubernetes

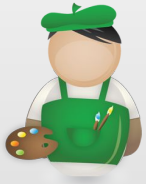
- Part1: Introduction to Docker and Installation, Docker commands
- Part2: Docker file and its related commands
- Part3: Volumes, network, docker compose

Satya Narayan Sahu

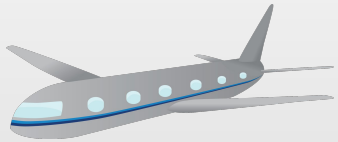
Dockerization :Part 1

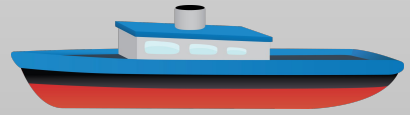
- Issue with normal shipping
- Problems before Docker
- How Docker solve the problem
- What is Docker?
- General work-flow of Docker
- Installation of Docker
- Running “Hello World” and “Centos” image with various commands

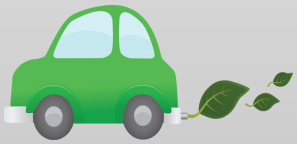
Issue with normal shipping

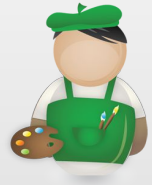


Different packaging for different transport mode

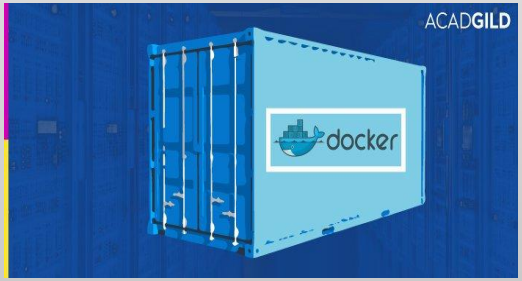








Container: Package once and ship anywhere with any transport mode



mindfiresolutions
BURN IGNORANCE

Satya Narayan Sahu

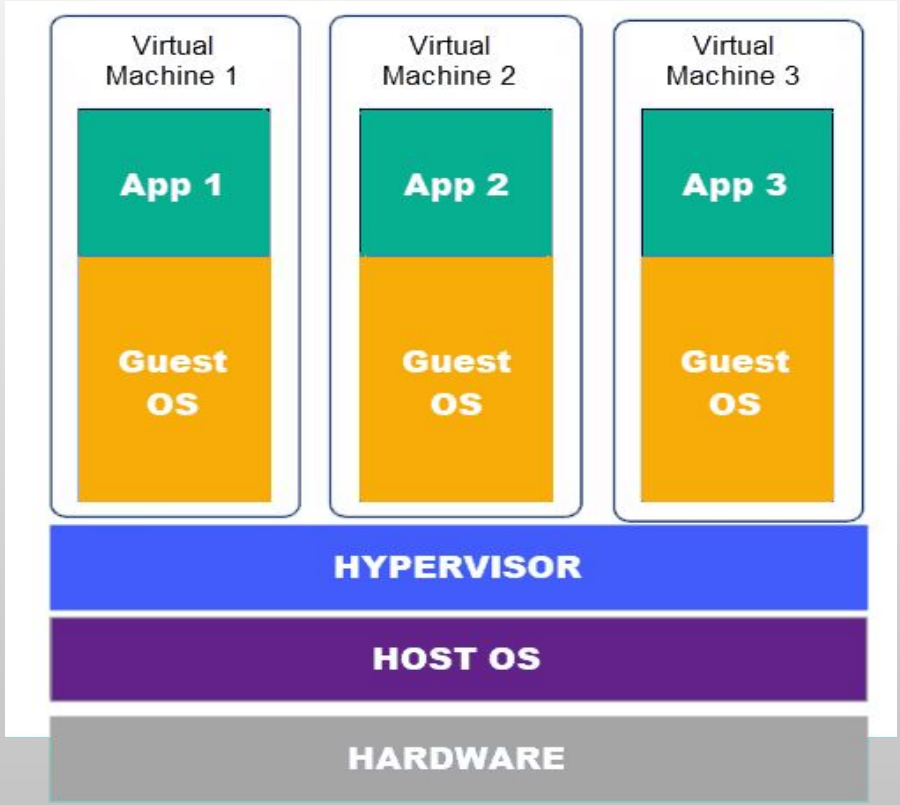
Problems before Docker

Application is running fine on developer system but not in production, because different computing environment

Developer

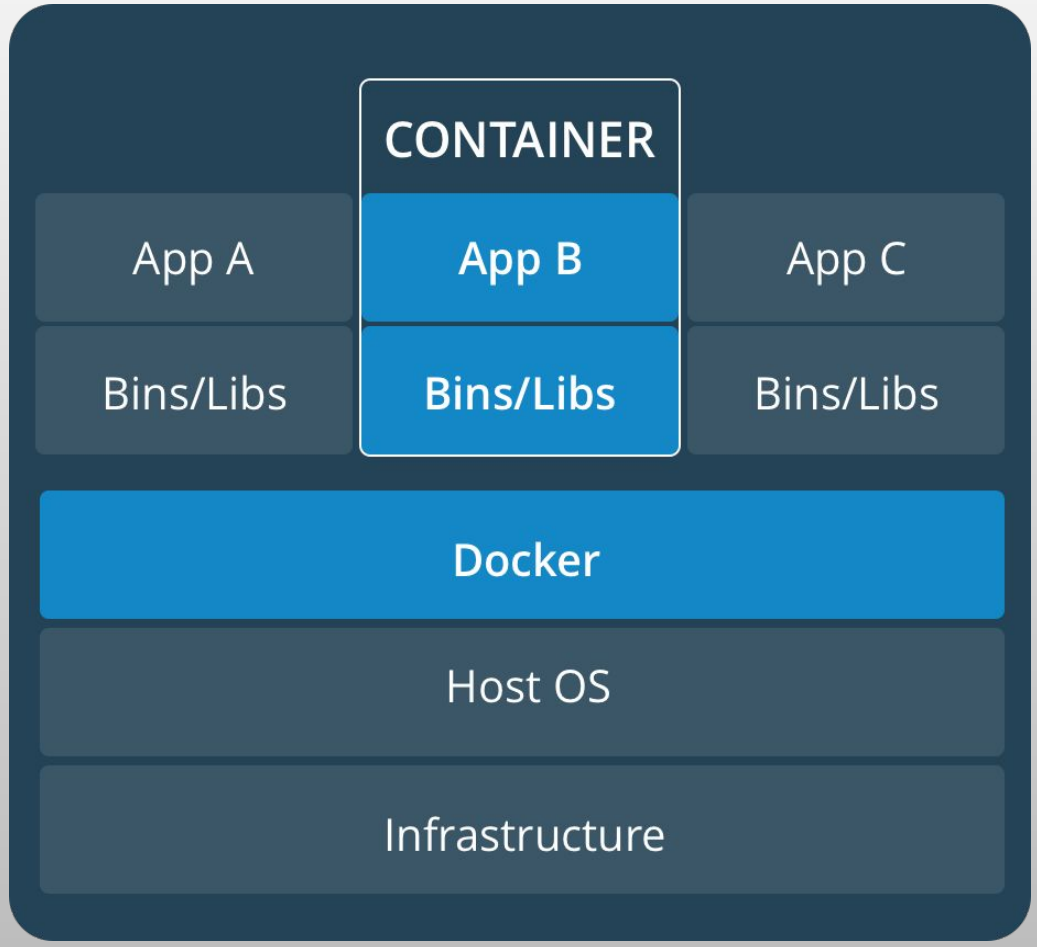
DevOps

Problems before Docker



- Each VM contains the dependency for one micro services.
- Disadvantage lots of wastage of resources like RAM, processor, disk space.
- What happen if we need 50 micro services?
- So VM is not an option make seance.

What is Docker?

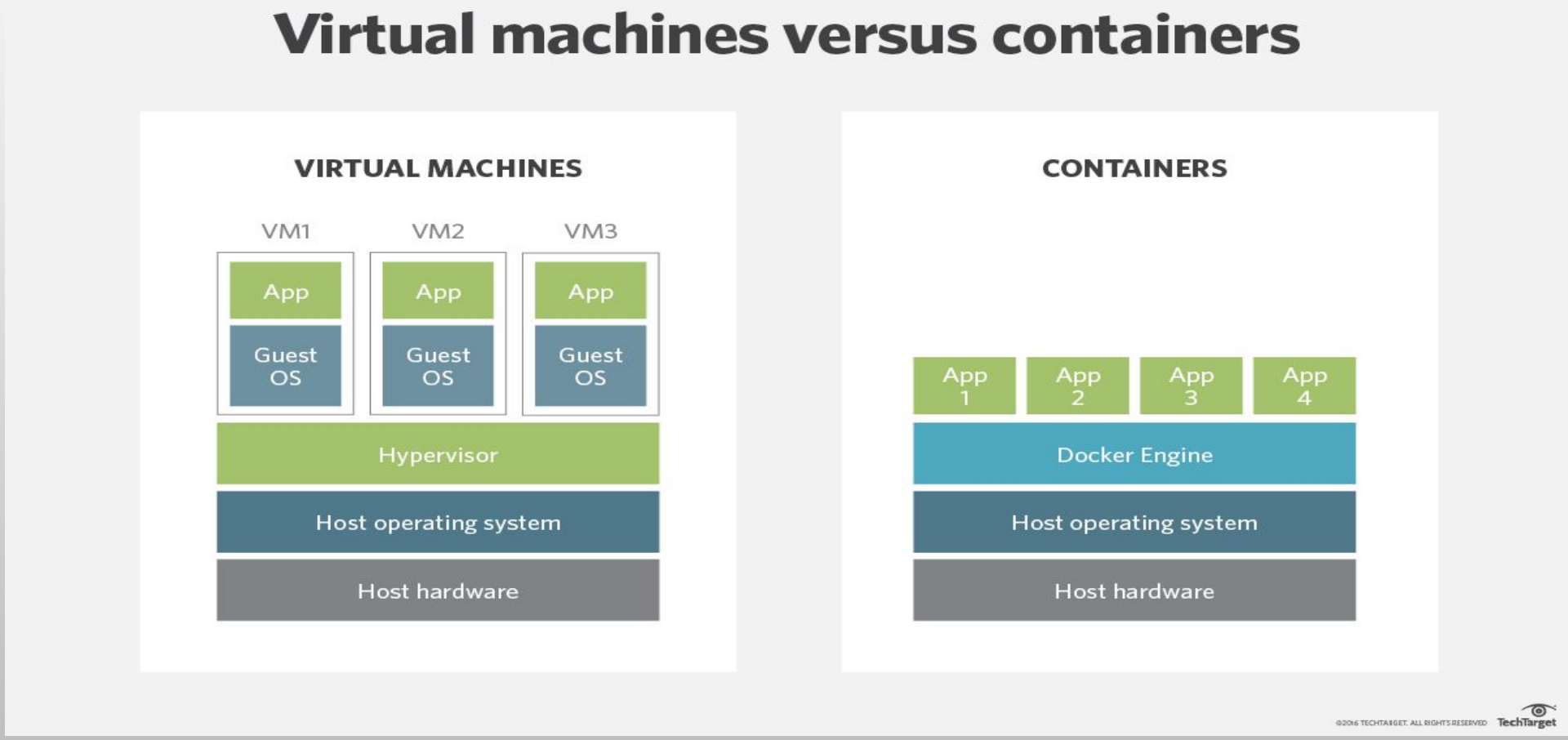


- Docker is the world most “Software container platform”.
- Docker is a tool designed to make it easier to create run and deploy application by using containers.
- Docker container are light-weight alternatives of VM.
- Fully resource utilization.

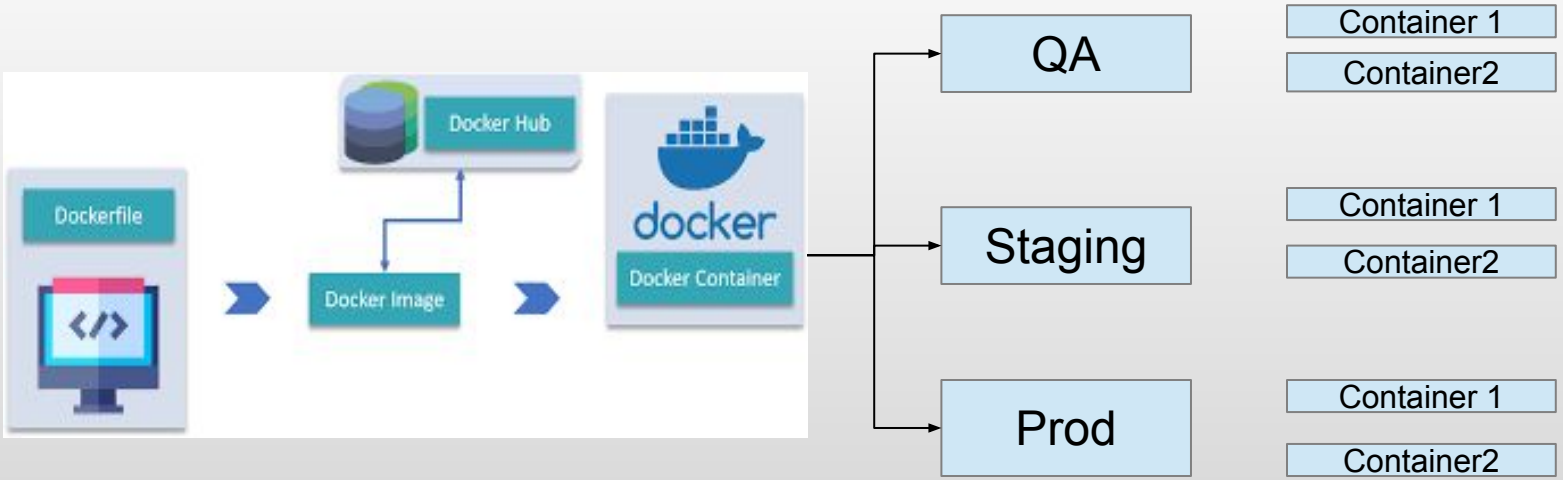
mindfiresolutions
BURN IGNORANCE

Satya Narayan Sahu

How Docker solve the problem



General workflow of Docker



- A developer write the code that defines an app requirements or dependency in an easy to docker file.
- Docker file produces docker image, so whatever dependency required for that application present in that docker image.
- Now image is uploaded to docker hub.
- From docker hub DaveOps people pull the image and create container.
- Advantages whatever dependency requisites present throughout the software development life cycle.

Installation of Docker for windows and linux

- https://docs.docker.com/docker-for-windows/install/
- https://runnable.com/docker/install-docker-on-linux

Running “Hello World” and “CentOS” image

Docker image pull

```
D:\>docker pull centos
Using default tag: latest
latest: Pulling from library/centos
8a29a15cefae: Pull complete
Digest: sha256:fe8d824220415eed5477b63addf40fb06c3b049404242b31982106ac204f6700
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest
```

Running Docker Image

```
D:\>docker run -it centos
[root@7f89bca5bc2b /]# exit
exit
```

Docker Commands (https://labs.play-with-docker.com/)

- **docker --version** -Version of docker Engine This command returns the version of docker installed
- **docker --help** -This command returns list of command available in docker along with flag
- **docker pull** -This command is used to pull images from the docker repository(hub.docker.com)

docker pull ubuntu:14.04

- **docker build** -This command is used to build an image from a specified docker file

docker build -t demo .

- **docker run** -This command is used to create a container from an image

docker run hello-world

- **docker exec**-This command is used to access the running container

docker run -d -it ubuntu

docker exec -it ef913eddd2bc bash **(-it interactive mode)**

Docker Commands

- **docker ps**-This command is used to list the running containers
- **docker images** -This command lists all the locally stored docker images
- **docker stop** -This command stops a running container
- **docker kill** -This command kills the container by stopping its execution immediately
- **docker rm** -This command is used to delete a stopped container
- **docker rmi** -This command is used to delete an image from local storage
- **docker commit** -This command creates a new image of an edited container on the local system
- docker commit [container id] satyanarayansahu/imagename

Docker Commands

- **docker login** -This command is used to login to the docker hub repository
- **docker push** -This command is used to push an image to the docker hub repository

docker tag demo satyanarayansahu/demo:version
docker push satyanarayansahu/demo

- **docker container -** Manage containers(start, run, kill, stop, rm)

docker container start|run|stop

- **docker compose -**Docker compose commands

Assignment

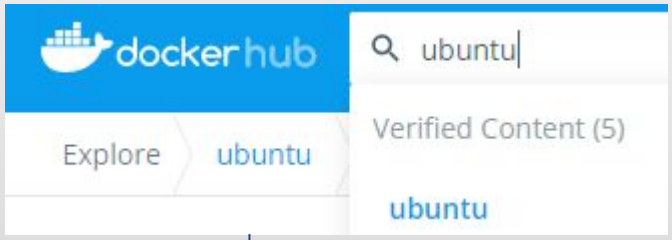
-Pull ubuntu image
-Customize ubuntu image with ur own image name(tag) with version
-login to docker hub and push the image
-Now pull the image again and run that
-Try to work with different version

Docker File

- What is Dockerfile
- Syntax of Dockerfile
- Play with Dockerfile

What is docker file?

How I can build my custom Image?



Base Image

Create Own Dockerfile

Set of Instruction

- FROM
- RUN
- CMD
- ENTRYPOINT
- EXPOSE
- ARG
- ENV
- WORKDIR

mindfiresolutions
BURN IGNORANCE

Satya Narayan Sahu

Syntax DockerFile

Dockerfile is a text file without any extension

COMMANDS + ARGUMENTS

Example:

```
#Print hello world
RUN echo “Hello world”
```

Syntax DockerFile

FROM The FROM instruction initializes a new build stage and sets the Base Image for subsequent instructions

```
#Fetch base image
FROM ubuntu

#Fetch base image
ARG  CODE_VERSION=latest
FROM ubuntu:${CODE_VERSION}
```

Syntax DockerFile

RUN — The RUN instruction will execute any commands in a new layer on top of the current image and commit the results.

RUN <command>
RUN ["executable", "param1", "param2"]

#Fetch base image
RUN apt-get update

#Run command
RUN ["/bin/bash", "-c", "echo hello"]

Syntax DockerFile

CMD

The main purpose of a CMD is to provide defaults for an executing container. There can only be one CMD instruction in a Dockerfile. If you list more than one CMD then only the last CMD will take effect.

CMD ["executable","param1","param2"] (exec form, this is the preferred form)
CMD ["param1","param2"] (as default parameters to ENTRYPOINT)
CMD command param1 param2

#cmd
CMD “echo” “hello world”

#Run command
CMD ["java", "-jar", "/demo.jar"]

Syntax DockerFile

ENTRYPOINT An ENTRYPOINT allows you to configure a container that will run as an executable.

```
ENTRYPOINT ["executable", "param1", "param2"]
ENTRYPOINT command param1 param2

# Run the jar file
ENTRYPOINT ["java", "-jar", "/demo.jar"]

#Arguments can be overide
CMD “Hello World”
ENTRYPOINT echo
echo "
```

Override the entrypoint instruction during run time :**docker run --entrypoint**

Syntax DockerFile

Understand how CMD and ENTRYPOINT interact

Both CMD and ENTRYPOINT instructions define what command gets executed when running a container. There are few rules that describe their co-operation.

Dockerfile should specify at least one of CMD or ENTRYPOINT commands.

ENTRYPOINT should be defined when using the container as an executable.

CMD should be used as a way of defining default arguments for an ENTRYPOINT command or for executing an ad-hoc command in a container.

CMD will be overridden when running the container with alternative arguments.

Syntax DockerFile

ADD

Copy the content from source to host

ADD source destination

```
# Add files
ADD /myfolder /myfolder

# The application's jar file
ARG JAR_FILE=target/demo.jar
# Add the application's jar to the container
ADD ${JAR_FILE} demo.jar
```

Syntax DockerFile

ENV

The ENV instruction sets the environment variable <key> to the value <value>. This value will be in the environment for all subsequent instructions in the build stage and can be replaced inline in many as well.

ENV <key> <value>
ENV <key>=<value> ...

#Get env variable
ENV NAME satya

Syntax DockerFile

WORKDIR

The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile
The WORKDIR instruction can be used multiple times in a Dockerfile

```
WORKDIR /path/to/workdir

ENV DIRPATH /path
WORKDIR $DIRPATH/$DIRNAME
RUN pwd
```

Syntax DockerFile

ARG

The ARG instruction defines a variable that users can pass at build-time to the builder with the docker build command using the --build-arg <varname>=<value> flag

ARG <name>[=<default value>]

```
FROM ubuntu
ARG CONT_IMG_VER
ENV CONT_IMG_VER v1.0.0
RUN echo $CONT_IMG_VER
```

```
$ docker build --build-arg CONT_IMG_VER=v2.0.1 .
```

Syntax DockerFile

EXPOSE

The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime.
By default, EXPOSE assumes TCP. You can also specify UDP:

EXPOSE <port> [<port>/<protocol>...]

```
# Make port 8080 available to the world outside this container
EXPOSE 8080
```

Syntax DockerFile

MAINTAINER

The MAINTAINER instruction sets the Author field of the generated images.

MAINTAINER <name>

#Set Auther name
MAINTAINER satya

Syntax DockerFile

VOLUME — The VOLUME instruction creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers.

```
VOLUME ["/data"]
```

```
FROM ubuntu
RUN mkdir /myvol
RUN echo "hello world" > /myvol/greeting
VOLUME /myvol
```

Syntax DockerFile

-What the OS we are going to use?
-We need to install JRE
-Keep the jar file
-Run jar by using java -jar

Now We will do the same by using dockerfile