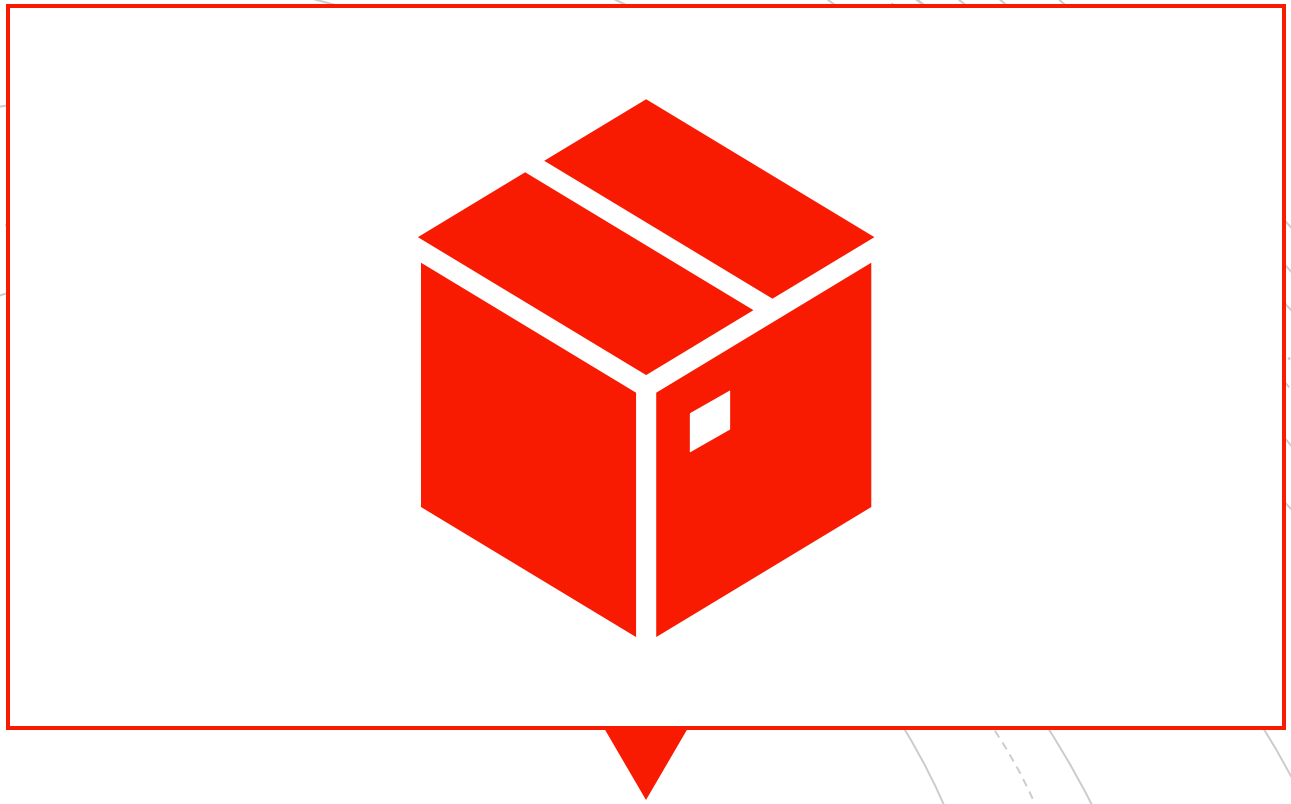
Docker

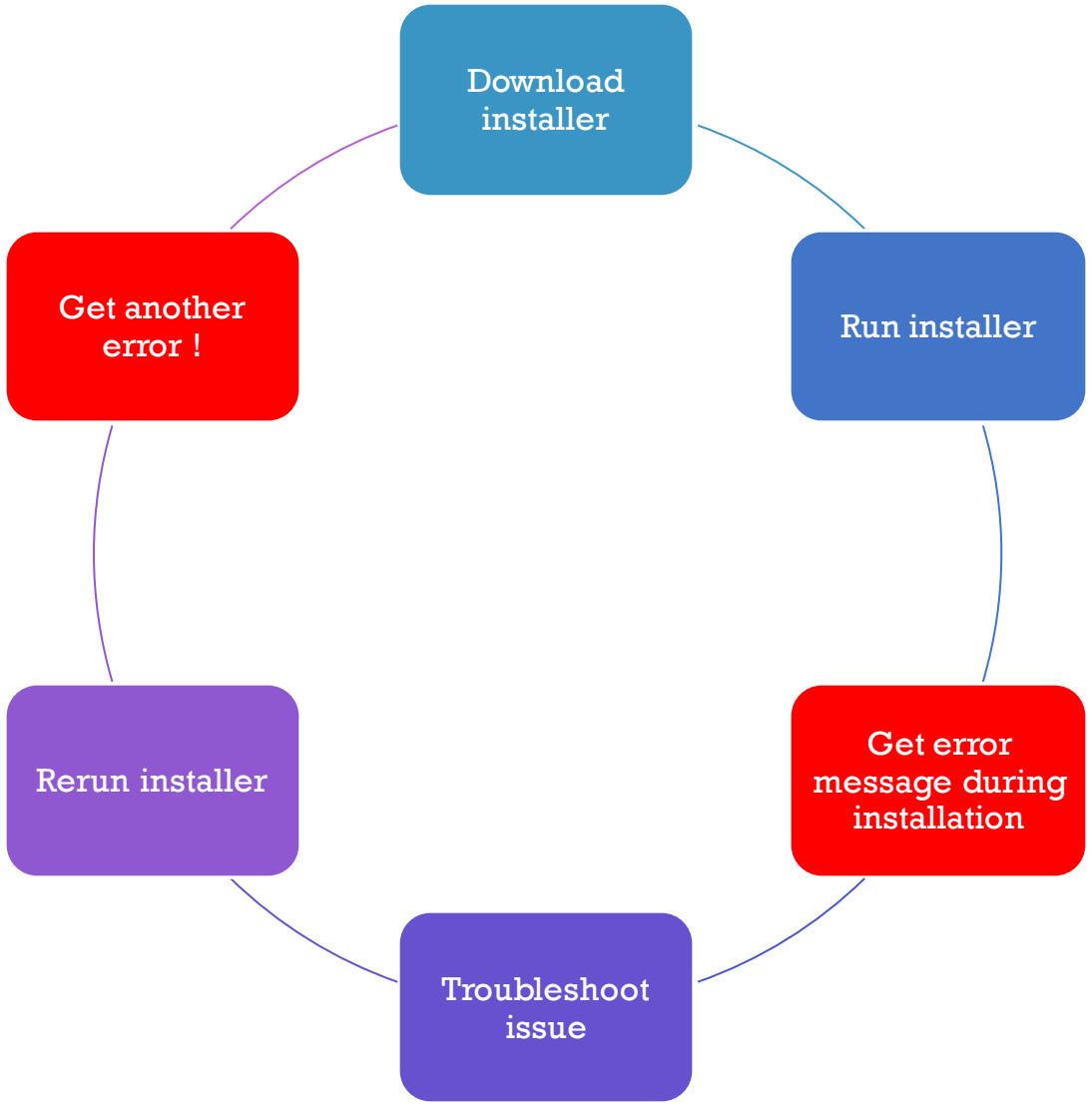
By: Abdelrehim Morsy

Agenda:

- Before Docker.
- containerization vs virtualization.
- What is Docker and Why?
- Docker Architecture.
- Container Lifecycle.
- Docker Volumes
- Docker Image.
- Docker Networks.

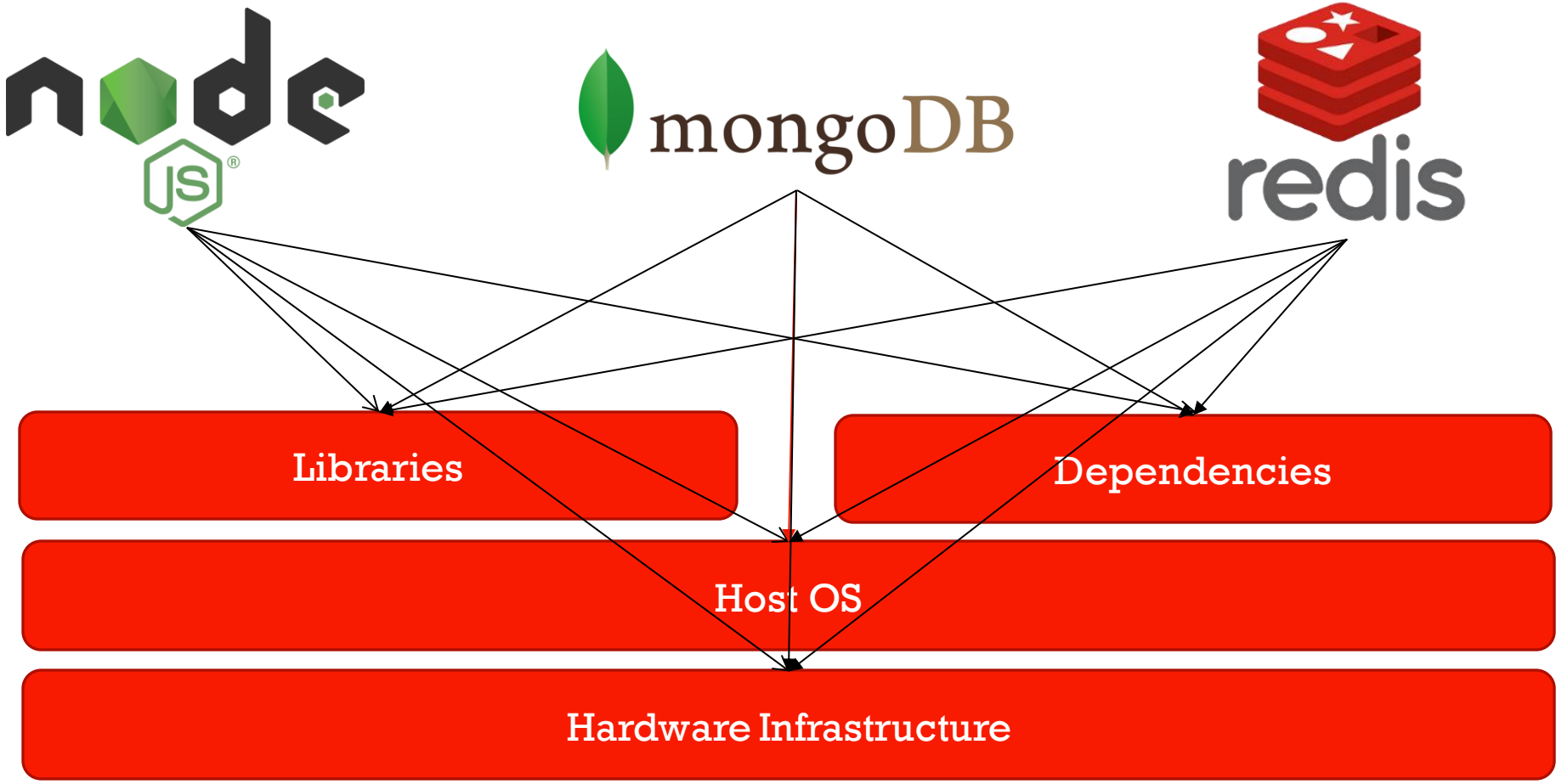


Before Docker

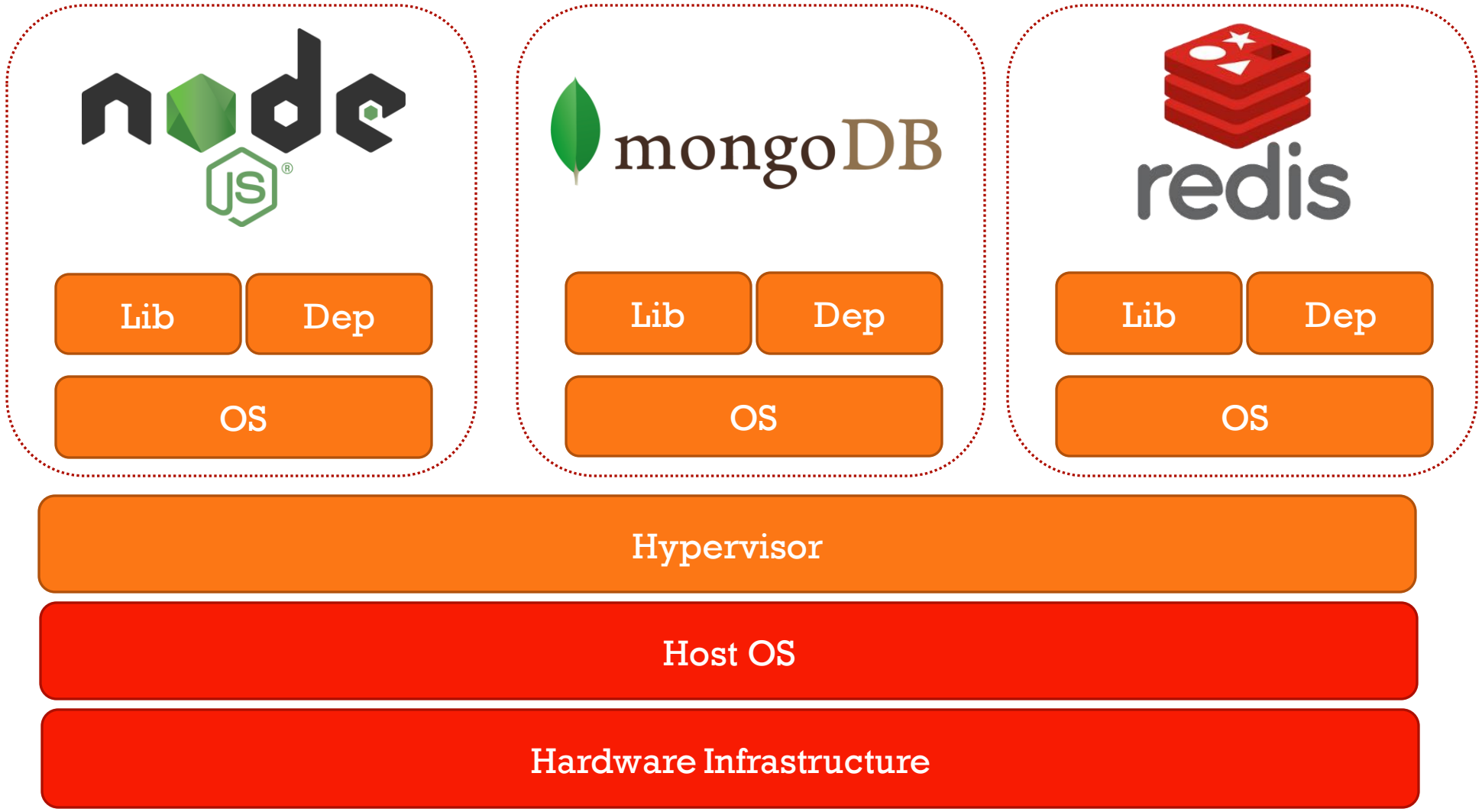


Installing software flow

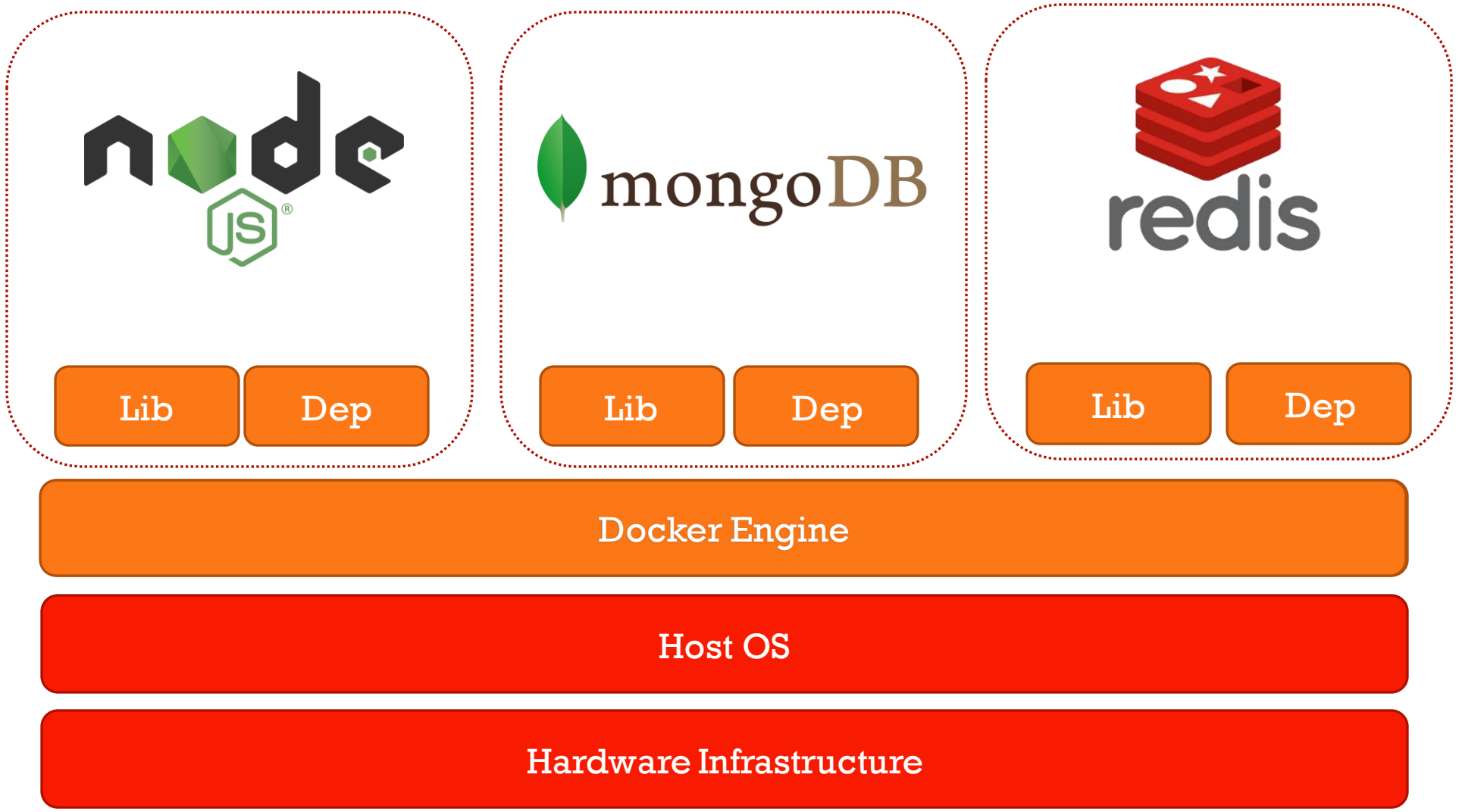
End-to-end application stack



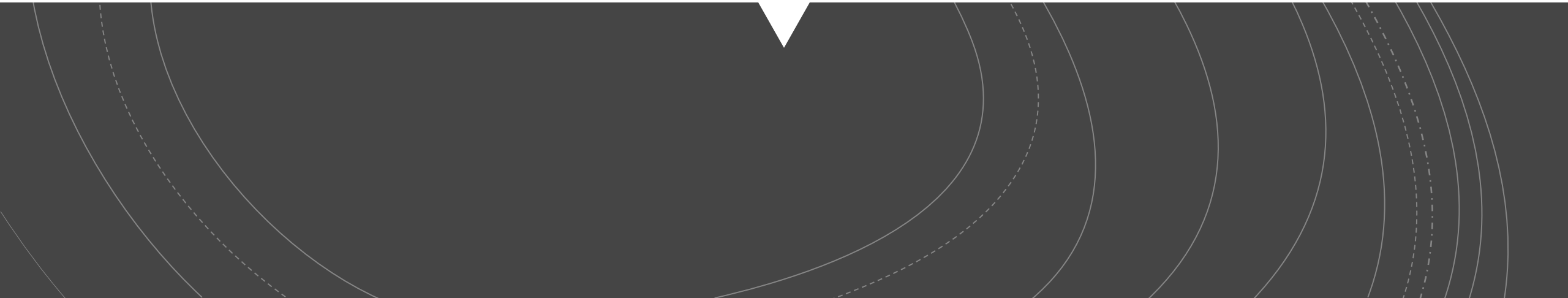
virtualization



containerization



Containerization vs Virtualization



containerization vs virtualization

UTILIZATION

SIZE

SPEED UP

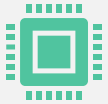
Docker

WHY USE DOCKER ?

WHAT IS DOCKER ?

Docker makes it really easy to install and run software without worrying about setup or dependencies.

Short setup time.



Different Dev/Test/Prod environments.

Why use Docker !

What is Docker ?

- Docker is an open platform or ecosystem around creating and running something called containers.
- Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.

Docker Client

Docker Daemon

Docker Compose

Docker Registery

Docker Objects

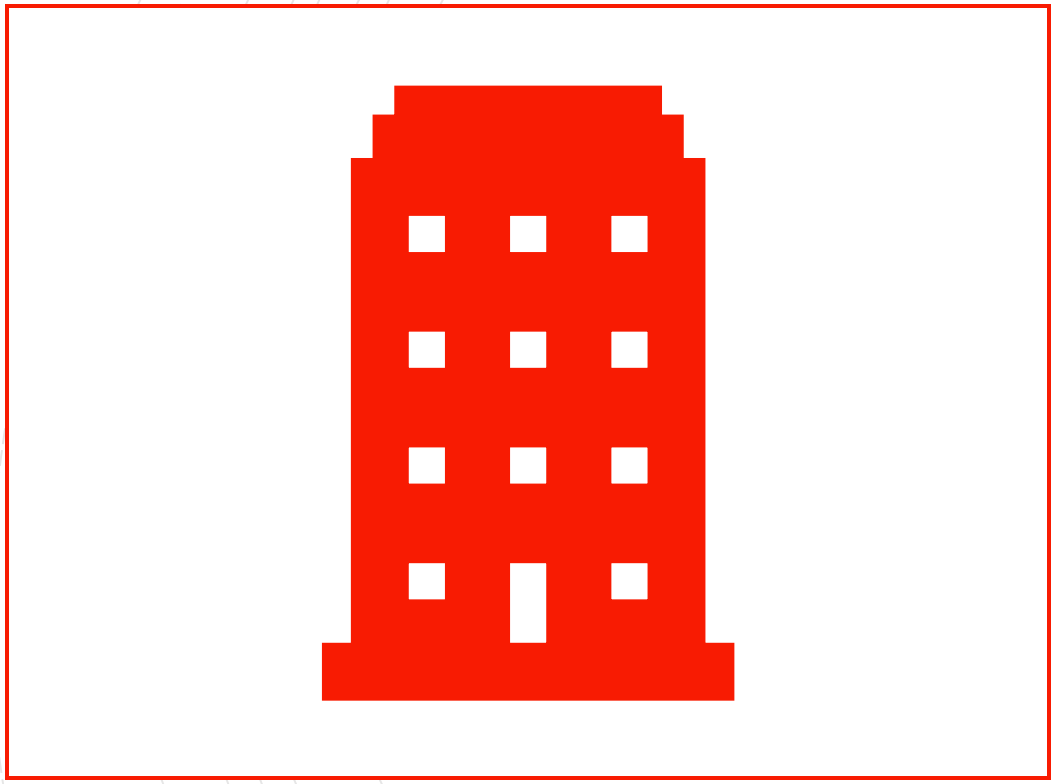
Image

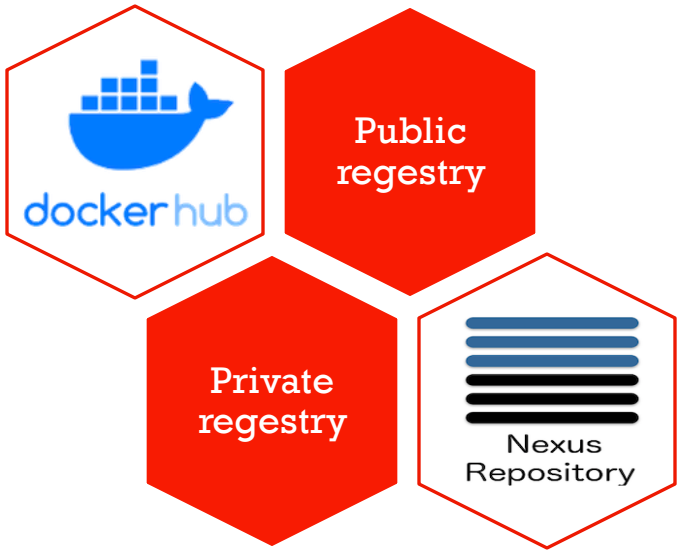
- Is a read-only template with instructions for creating a Docker container.

Container

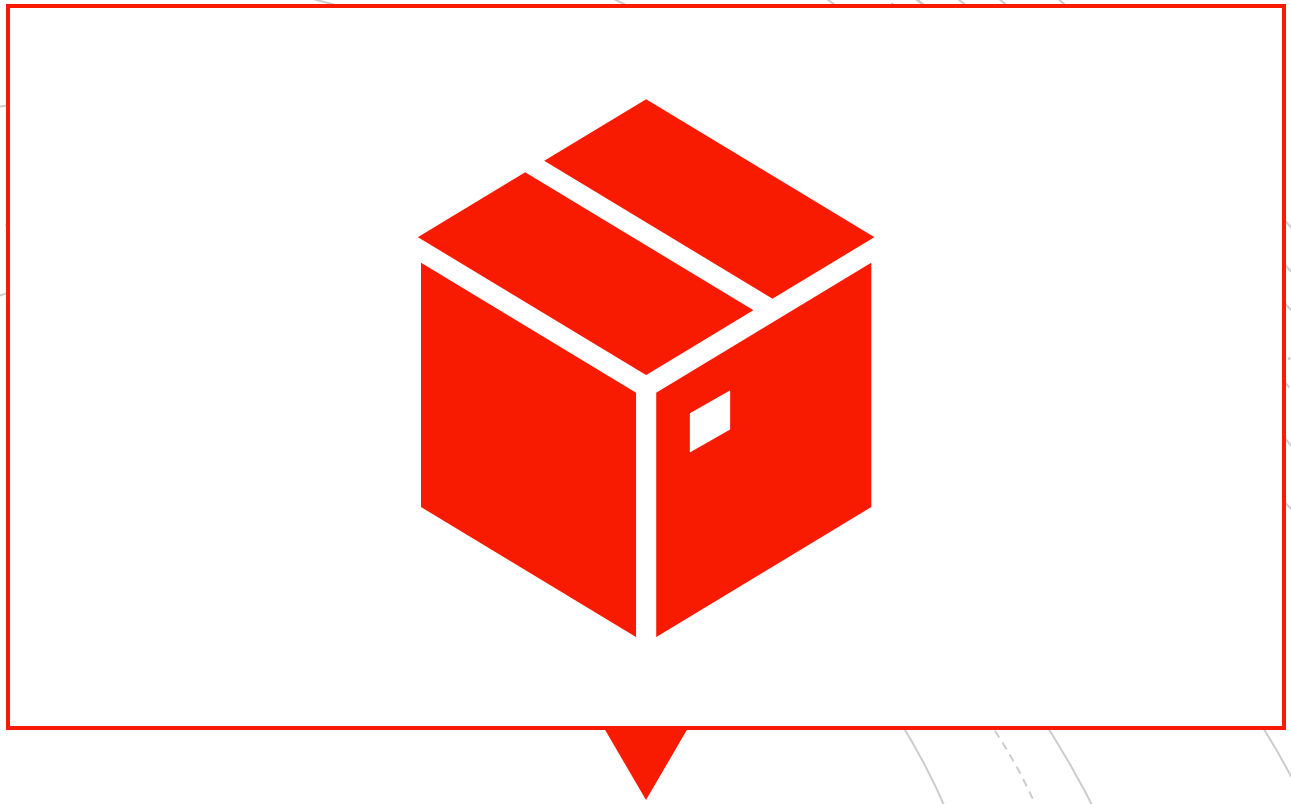
- Instance of an image. Runs a program.

Types of Docker regestries





Break



Container Lifecycle

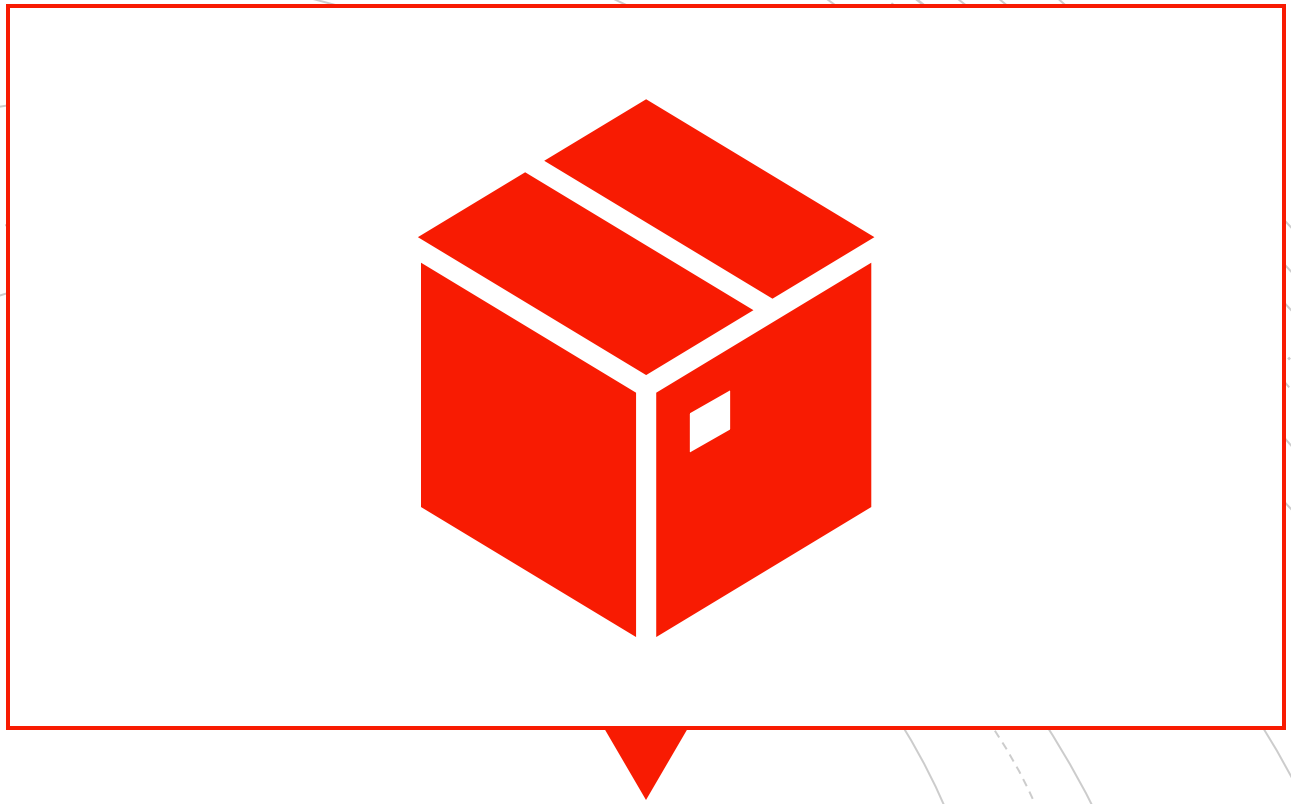
Container Lifecycle

- docker run:
  - Create: take file system snapshot and add as a process ready to run in hard drive.
  - Start: Run start up command
  - Exited: finished its job and exit.
- docker stop: terminate single but wait
- docker kill: force terminate
- docker prune: clear docker caches, images and stopped containers.

Manipulating Containers with the Docker Client

Docker Basic Commands

Run Container	•$ docker run <image-name>
Overriding default command	•$ docker run <image-name> <command!>
Attached and Detached	•$ docker run –d <image-name> •$ docker attach <container ID>
Interactive mode	•$ docker run -i <image-name>
Port mapping	•$ docker run –p host:container <image-name>
Volume mapping	•$ docker run –v host:container <image-name>
Inspect container	•$ docker inspect <container-name>
Stop container	• $ docker stop CONTAINER
List all images	•$ docker images -a
List all containers	•$ docker ps -a

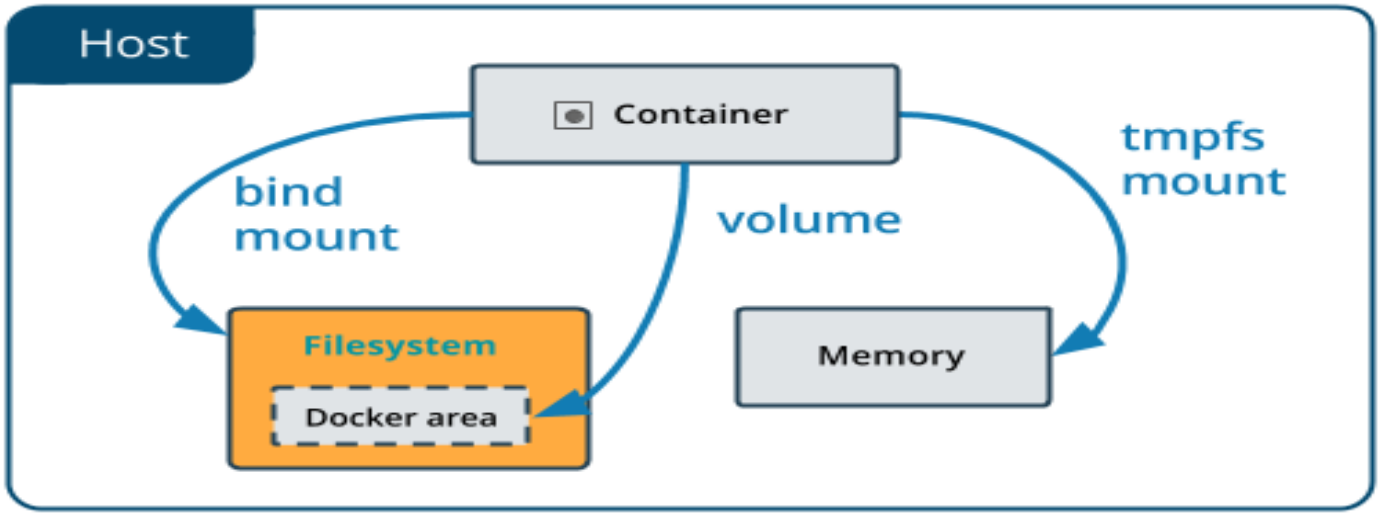


Docker Volume

Docker Volumes

- Mount:
  - a file or directory on the *host machine* is mounted into a container
  - The file or directory is referenced by its absolute path on the host machine

- Volume:
  - Anonymous VS Named
  - a new directory is created within Docker's storage directory on the host machine, and Docker manages that directory's contents. "/var/lib/docker/volumes"



Lab

Problem 1

- Run the container hello-world
- Check the container status
- Start the stopped container
- Remove the container
- Remove the image

Problem 2

- Run container centos or ubuntu in an interactive mode
- Run the following command in the container “echo docker ”
- Open a bash shell in the container and touch a file named hello-docker
- Stop the container and remove it. Write your comment about the file hello-docker
- Remove all stopped containers

Problem 3

- Run a container httpd with name apache and attach a volume to the container
  - Volume for containing static html file
- Remove the container
- Run a new container with the following:
  - Attach the volume that was attached to the previous container
  - Map port 80 to port 9898 on you host machine
  - Access the html files from your browser

Problem 4

- Run the image httpd again without attaching any volumes
- Add html static files to the container and make sure they are accessible
- Commit the container with image name my apache
- Create a dockerfile for ngnix and build the image from this dockerfile

Problem 5

- Create a volume called mysql_data, then deploy a MySQL database called app-database. Use the mysql latest image, and use the -e flag to set MYSQL_ROOT_PASSWORD to P4sSw0rd0!. Mount the mysql_data volume to /var/lib/mysql. The container should run in the background.

Thanks for attention