Understanding Containers

Shivam Jha   Email: shivamjhaonthecloud@gmail.com

Container

Build,Ship and Run anywhere

Why Containers ?

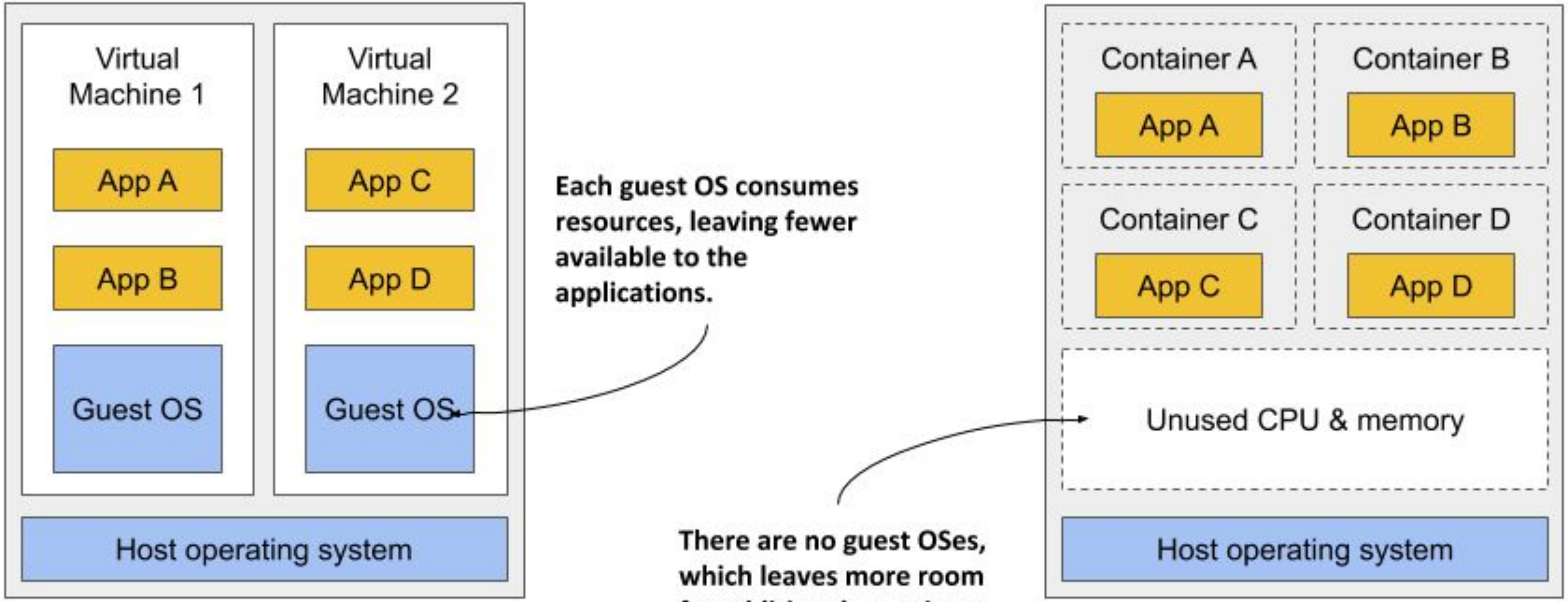
- When a system consists of a small number of applications, it’s okay to assign a dedicated VM’s to each application and run each in its own operating system. But as the microservices become smaller and their numbers start to grow, you may not be able to afford to give each one its own VM if you want to keep your hardware costs low and not waste resources

- Each VM typically needs to be individually configured and managed, which means that running higher numbers of VMs also results in higher staffing requirements and the need for a better, often more complicated automation system.

- Due to the shift to microservice architectures, where systems consist of hundreds of deployed application instances, an alternative to VMs was needed. Containers are that alternative.

https://docker-from-scratch.ivonet.nl/docs/about.html

Virtual Machines vs Container



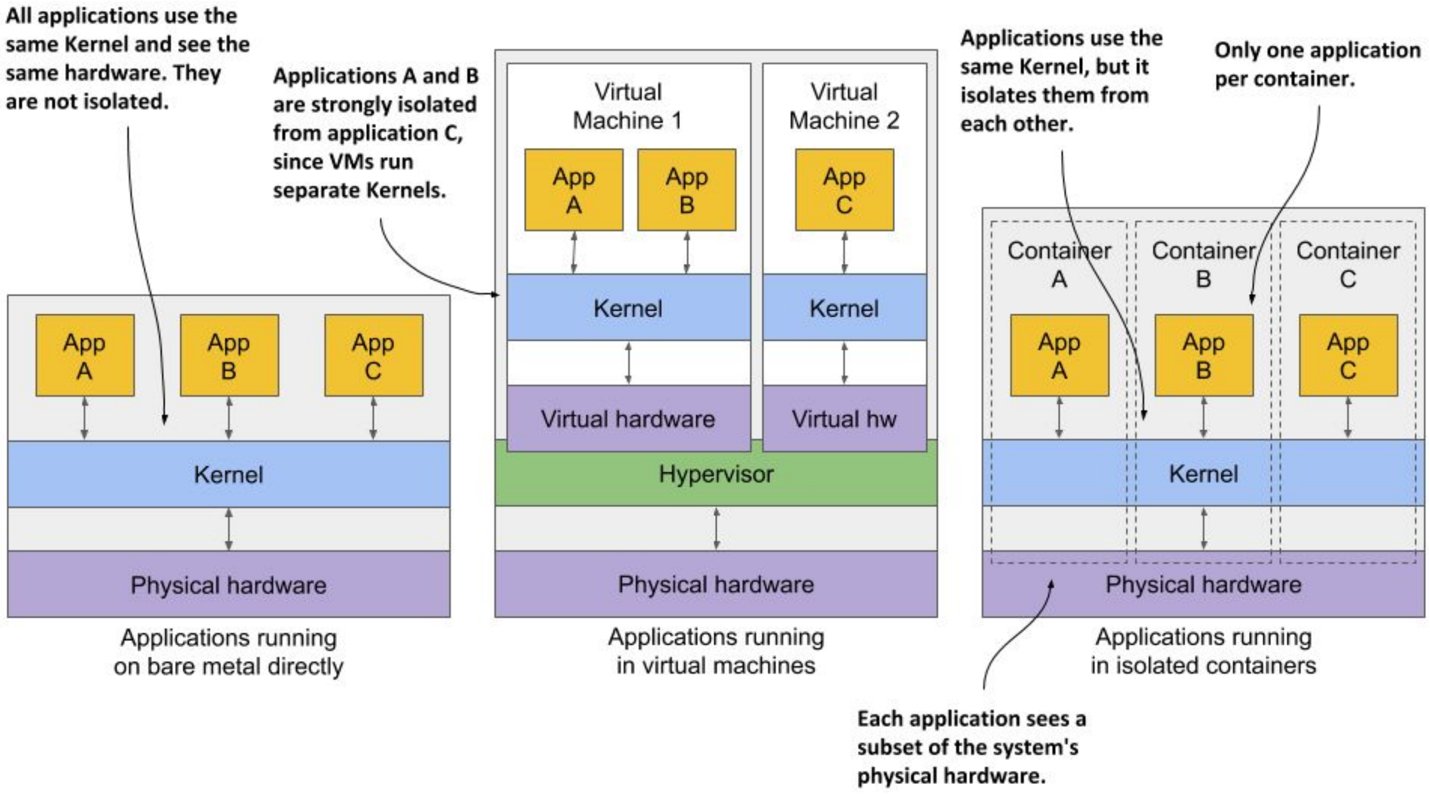
Each guest OS consumes resources, leaving fewer available to the applications.

There are no guest OSes, which leaves more room for additional containers

System using virtual machines
to isolate (groups of) applications

System using containers
to isolate individual applications

Bare Metal vs Virtual Machines vs Container



All applications use the same Kernel and see the same hardware. They are not isolated.

Applications A and B are strongly isolated from application C, since VMs run separate Kernels.

Applications use the same Kernel, but it isolates them from each other.

Only one application per container.

Applications running on bare metal directly

Applications running in virtual machines

Applications running in isolated containers

Each application sees a subset of the system's physical hardware.

Container drawbacks

- Containers all use the same kernel. This can clearly pose a security risk. If there’s a bug in the kernel, an application in one container might use it to read the memory of applications in other containers.

- Containers share memory space, whereas each VM uses its own chunk of memory. Therefore, if you don’t limit the amount of memory that a container can use, this could cause other containers to run out of memory or cause their data to be swapped out to disk.

THANK YOU…….

DO YOU HAVE ANY QUESTİONS ?

Docker Container

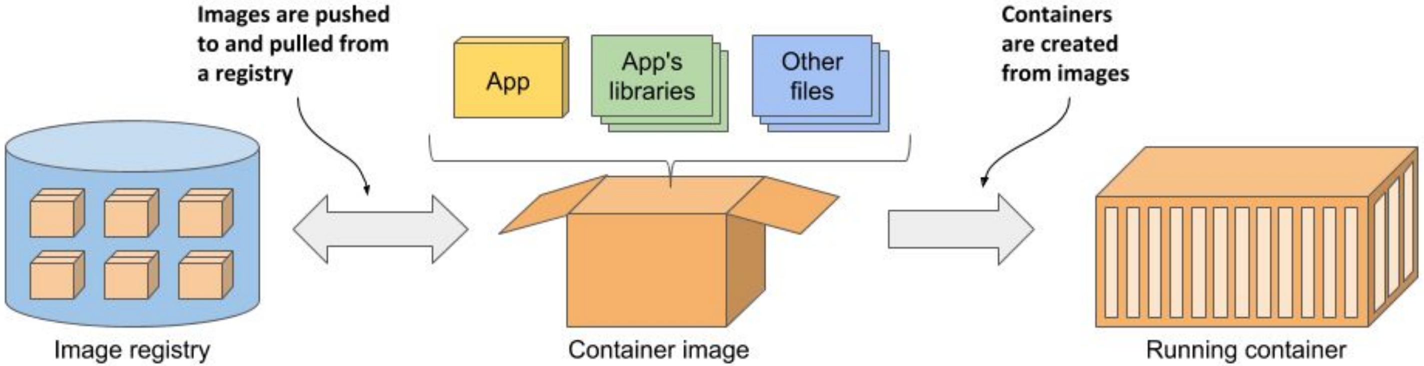
Build,Ship and Run anywhere

Overview

- Docker was the first container that made them easily portable across different computers.
- Docker is a platform for packaging, distributing and running applications.
- It simplified the process of packaging up the application and all its libraries and other dependencies - even the entire OS file system - into a simple, portable package that can be used to deploy the application on any computer running Docker.
- Docker allows you to distribute this package via a public repository to any other Docker-enabled computer.

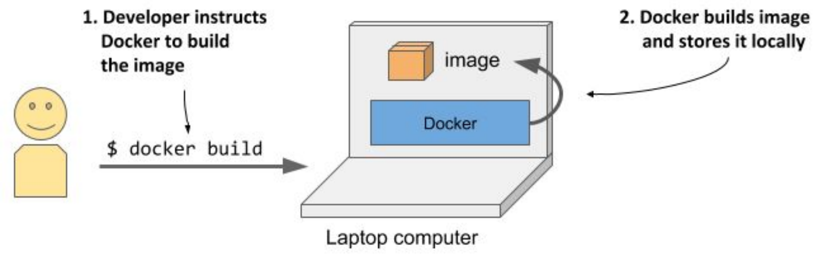
Docker Concepts

- Images- A container image is something you package your application and its environment into. Like a zip file or a tarball.
- Registries- A registry is a repository of container images that enables the exchange of images between different people and computers. After you build your image, you can either run it on the same computer, or push (upload) the image to a registry and then pull (download) it to another computer.
- Containers- A container is instantiated from a container image A running container is a normal process running in the host operating system, but its environment is isolated from that of the host and the environments of other processes.


Images are pushed to and pulled from a registry
App
App's libraries
Other files
Containers are created from images
Image registry
Container image
Running container

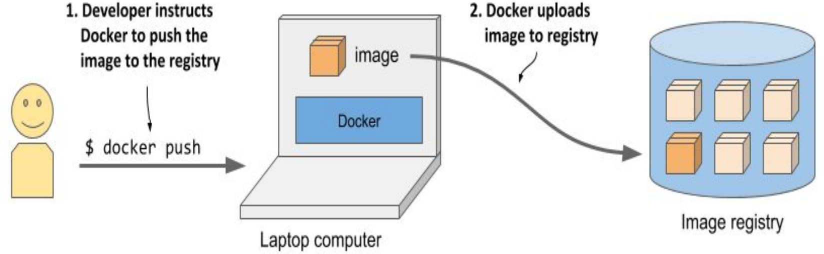
Build,Distribute and Run the Image

Three processes involves-



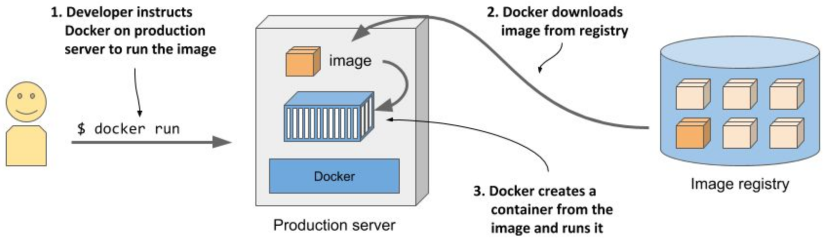
1. Developer instructs Docker to build the image
$ docker build
image
Docker
2. Docker builds image and stores it locally
Laptop computer

1 **Building a container image**



1. Developer instructs Docker to push the image to the registry
$ docker push
image
Docker
Laptop computer
2. Docker uploads image to registry
Image registry

2 **Uploading a container image to a registry**

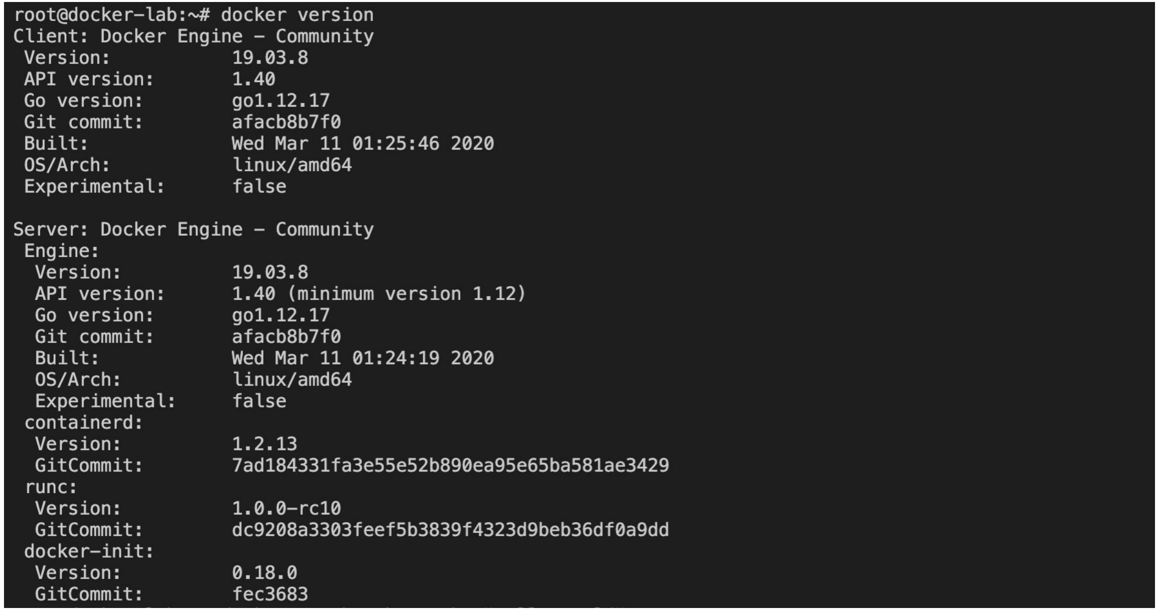


1. Developer instructs Docker on production server to run the image
$ docker run
image
Docker
Production server
2. Docker downloads image from registry
3. Docker creates a container from the image and runs it
Image registry

3 **Running a container on a different computer**

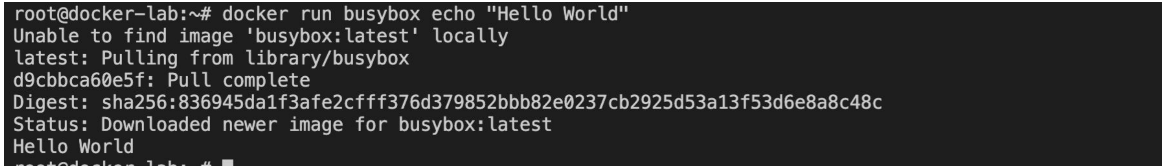
Hand-On

- Create an application, package it into a container image and run it.
- Installing docker and running a Hello World container



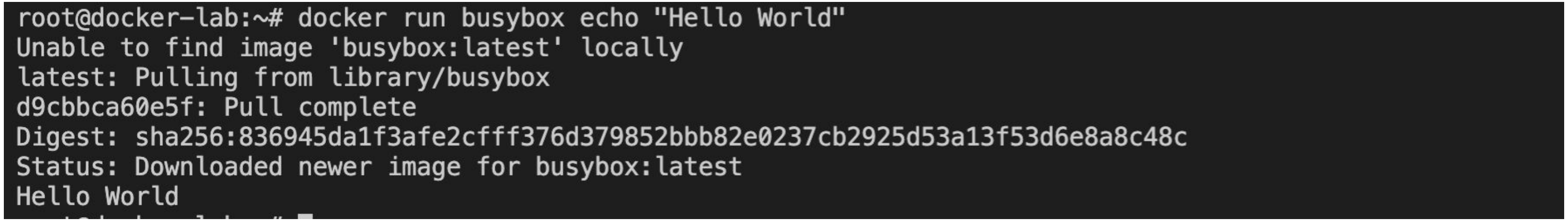
```
root@docker-lab:~# docker version
Client: Docker Engine - Community
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        afacb8b7f0
 Built:             Wed Mar 11 01:25:46 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
  Git commit:       afacb8b7f0
  Built:            Wed Mar 11 01:24:19 2020
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:        dc9208a3303feef5b3839f4323d9beb36df0a9dd
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```

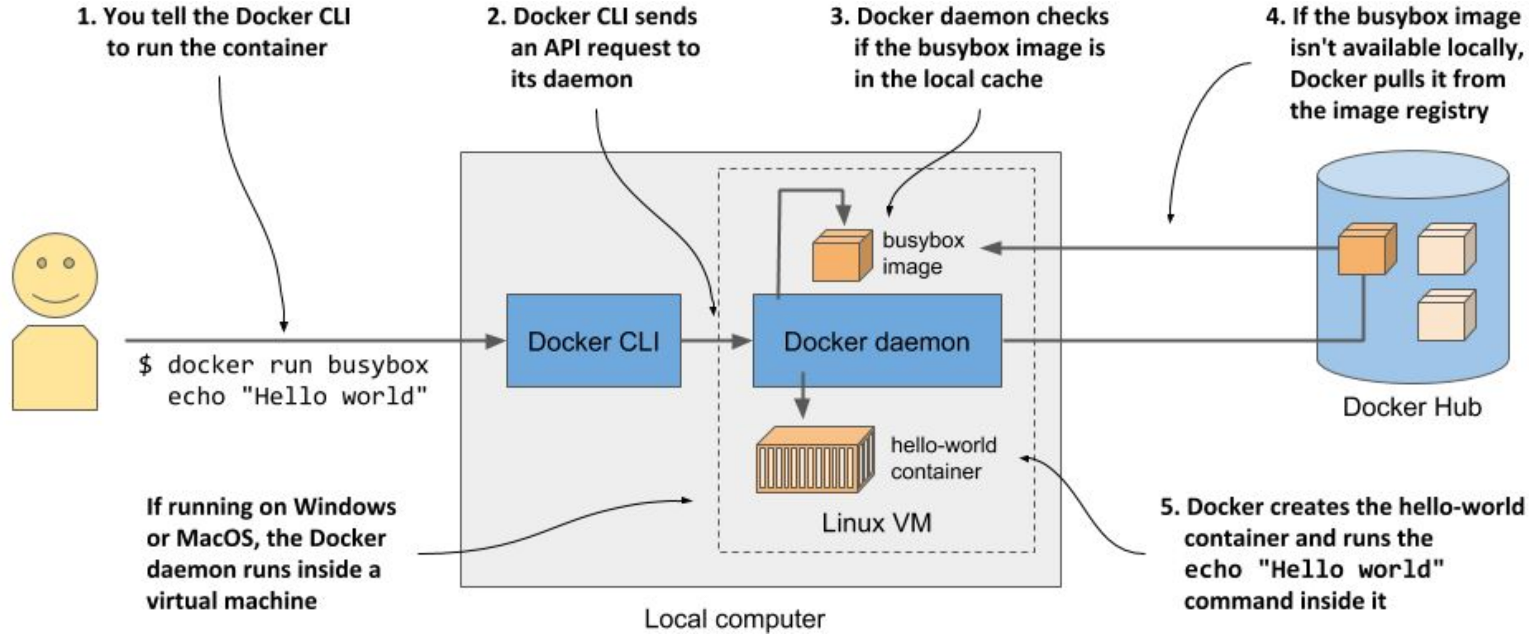


```
root@docker-lab:~# docker run busybox echo "Hello World"
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
d9cbbca60e5f: Pull complete
Digest: sha256:836945da1f3afe2cfff376d379852bbb82e0237cb2925d53a13f53d6e8a8c48c
Status: Downloaded newer image for busybox:latest
Hello World
root@docker-lab:~# █
```

Running “Hello World” image



```
root@docker-lab:~# docker run busybox echo "Hello World"
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
d9cbbca60e5f: Pull complete
Digest: sha256:836945da1f3afe2cfff376d379852bbb82e0237cb2925d53a13f53d6e8a8c48c
Status: Downloaded newer image for busybox:latest
Hello World
```



1. You tell the Docker CLI to run the container

2. Docker CLI sends an API request to its daemon

3. Docker daemon checks if the busybox image is in the local cache

4. If the busybox image isn't available locally, Docker pulls it from the image registry

`$ docker run busybox echo "Hello world"`

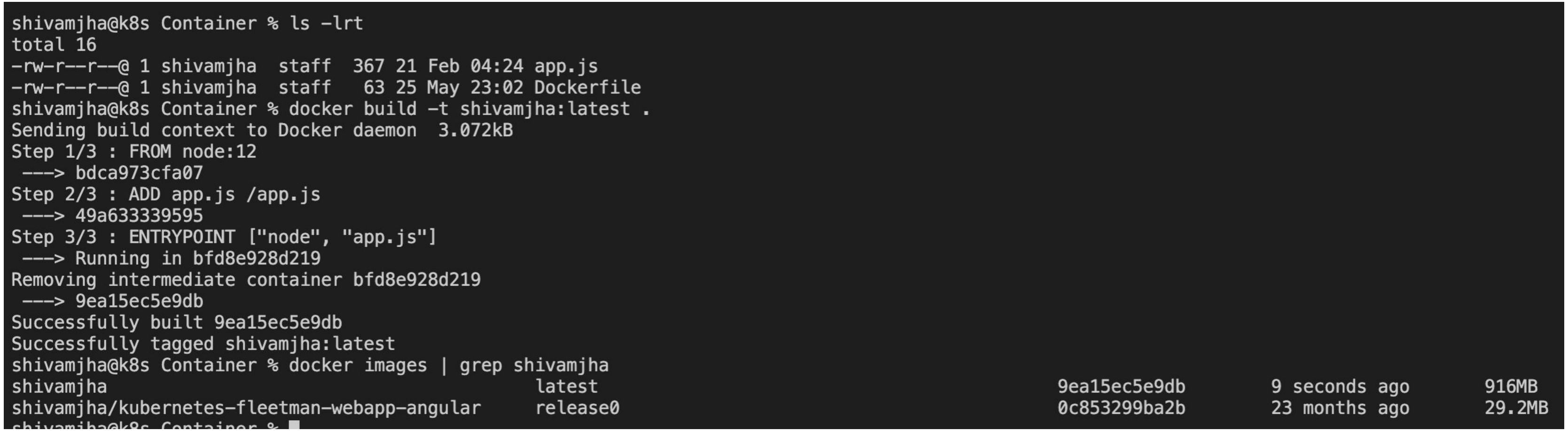
If running on Windows or MacOS, the Docker daemon runs inside a virtual machine

5. Docker creates the hello-world container and runs the `echo "Hello world"` command inside it

Lab- Create a containerized Web Application

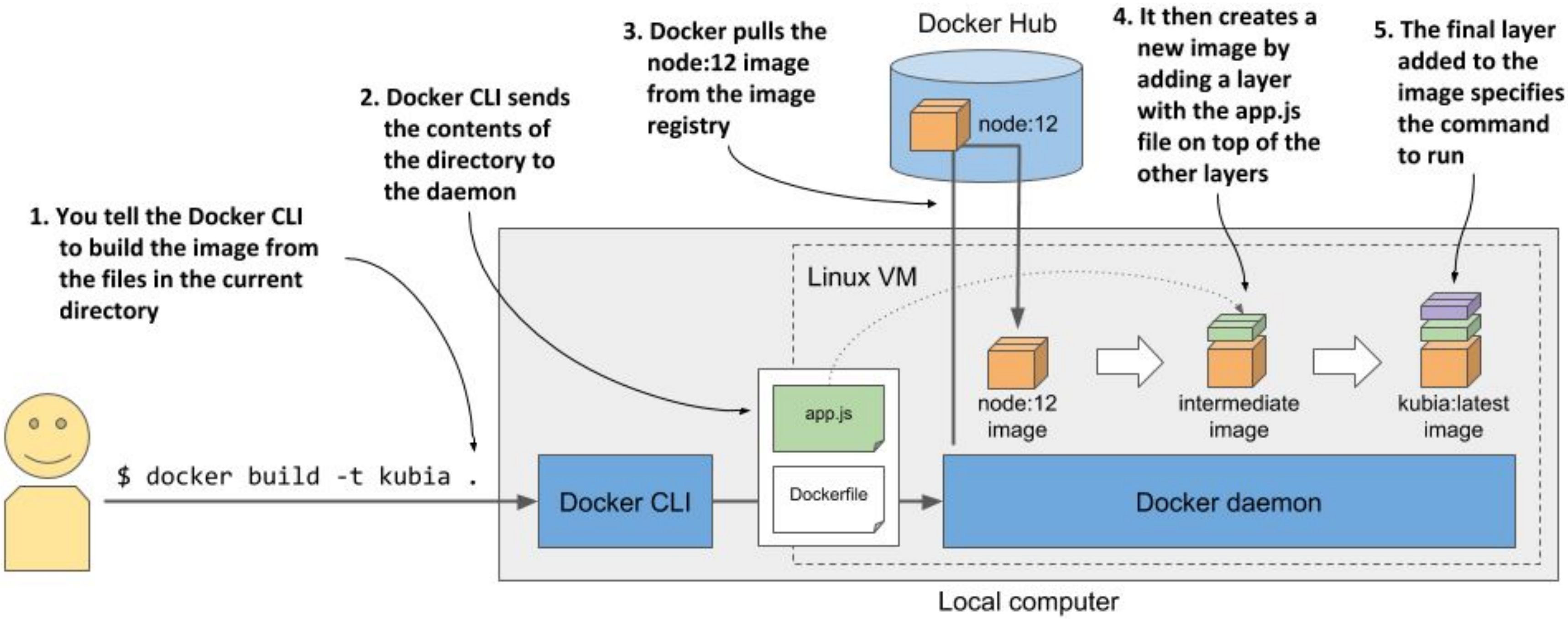
# Create a Node.js web application and package it into a container image.The application will accept HTTP requests and respond with the hostname of the computer it's running on.This way, you'll see that an app running in a container sees a different hostname and not that of the host computer, even though it runs on the host like any other process.

Lab- Create a containerized Web Application



```
shivamjha@k8s Container % ls -lrt
total 16
-rw-r--r--@ 1 shivamjha  staff  367 21 Feb 04:24 app.js
-rw-r--r--@ 1 shivamjha  staff   63 25 May 23:02 Dockerfile
shivamjha@k8s Container % docker build -t shivamjha:latest .
Sending build context to Docker daemon  3.072kB
Step 1/3 : FROM node:12
 ---> bdca973cfa07
Step 2/3 : ADD app.js /app.js
 ---> 49a633339595
Step 3/3 : ENTRYPOINT ["node", "app.js"]
 ---> Running in bfd8e928d219
Removing intermediate container bfd8e928d219
 ---> 9ea15ec5e9db
Successfully built 9ea15ec5e9db
Successfully tagged shivamjha:latest
shivamjha@k8s Container % docker images | grep shivamjha
shivamjha                                         latest                                           9ea15ec5e9db        9 seconds ago       916MB
shivamjha/kubernetes-fleetman-webapp-angular     release0                                         0c853299ba2b        23 months ago       29.2MB
shivamjha@k8s Container %
```

Lab- Create a containerized Web Application



1. You tell the Docker CLI to build the image from the files in the current directory

2. Docker CLI sends the contents of the directory to the daemon

3. Docker pulls the node:12 image from the image registry

4. It then creates a new image by adding a layer with the app.js file on top of the other layers

5. The final layer added to the image specifies the command to run

```
$ docker build -t kubia .
```

THANK YOU…….

DO YOU HAVE ANY QUESTİONS ?