**Go in Docker**

**Introduction to Docker**

**Initial concepts**

1. **Virtualisation**
2. **Hypervisor**
3. **Virtual Machines vs Linux Containers**

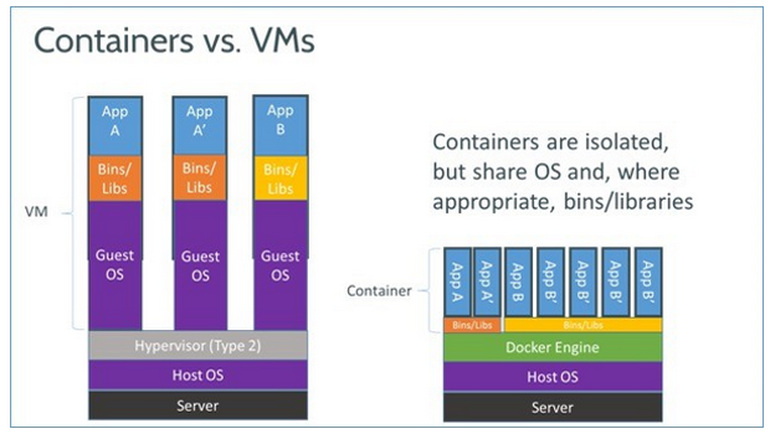
**Virtualisation: emulate a real devices in software**

**Hypervisor: Type-1 hypervisors (bare metal) Type-2 hypervisors(Virtual Box, VMWare, ..)**

**creates and runs the virtual machine**

**Virtual Machine: a virtual computer**

**Linux Containers: Linux Operating-system-level virtualization allows for multiple isolated namespace instances, instead of just one, cgroups allows resource management features to limit the impact of one container's activities on the other containers.**



**Summary**

* **Containers virtualize at the operating system level, Hypervisors virtualize at the hardware level.**
* **Hypervisors abstract the operating system from hardware, containers abstract the application from the operation system.**
* **Hypervisors consumes storage space for each instance. Containers use a single storage space plus are smaller for each layer and thus are much more efficient.**
* **Containers can boot much quicker than virtual machines.**

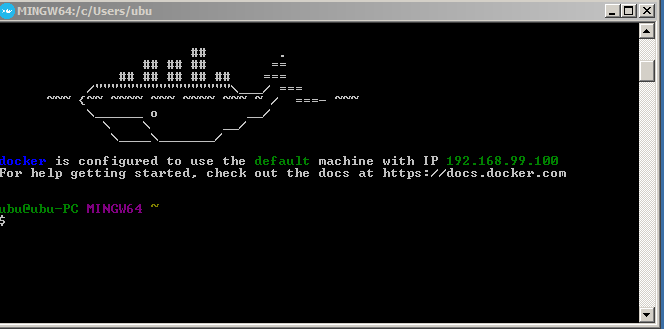
# **Docker**

* **Framework for container virtualization.**
* **Containers are Linux instances that hold applications.**
* **Docker is molded on the concept of shipping containers to present a standardised way of packaging/shipping**

1. install docker for your opperating system https://docs.docker.com/

windows looks like this is everything works out

suggest if windows, remove any previous installations of virtualbox and let docker install it for you.



**Docker so we will only look at the minimal set of docker commands you need to be productive**

// show all images

**docker images**

// remove an image using its id

**docker rmi 02be6f2a9681**

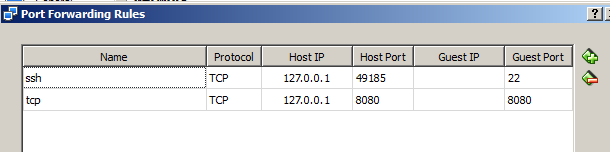
//shows running and stopped containers

**docker ps -a**

// remove all containers running or not

**docker rm $(docker ps -aq)**

**docker run -d -p 8080:8080 -ti google/golang /bin/bash**



**Create and run a docker container from the google/golang image**

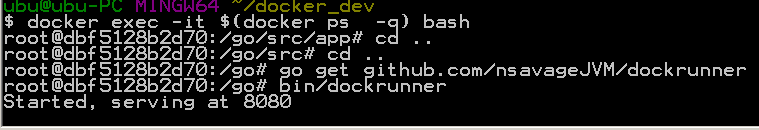
**docker run -d -p 8080:8080 -ti google/golang /bin/bash**

**docker exec -it $(docker ps -q) bash**

**go get github.com/nsavageJVM/dockrunner**

The go get command is a variant of the go build command that allows fetching and building remote dependencies. You can start the resulting executable with:

**bin/dockrunner**



**Find IP**

**docker exec -it $(docker ps -q) bash**

**ip add | grep global**

**docker exec -it loving\_nobel bash**

**docker run -d -p 8080:8080 -ti google/golang**

**docker exec -it $(docker ps -q) bash**

**ip add | grep global**

**docker run -d -p 8080:8080 -t -i --name google/golang /bin/bash**

**docker start small\_jepsen**

**docker stop small\_jepsen**

**First dockerfile**

**# golang image where workspace (GOPATH) configured at /go.**

**FROM golang:latest**

**WORKDIR /go/src/code**

**# Copy the local package files to the container’s workspace.**

**COPY code/server\_test.go /go/src/code/**

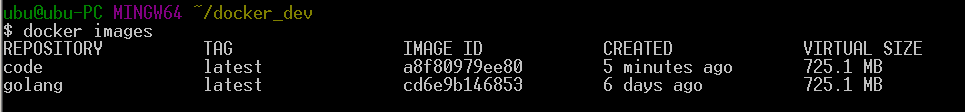
**# http server listens on port 8080.**

**EXPOSE 8080**

**Build the image for the container**

**docker build -t code .**

**docker images**



**run a interactive container**

**docker run -d -p 8080:8080 -t -i --name code**

**docker run -i -t --entrypoint /bin/bash <imageID>**

**docker run -i -t --entrypoint /bin/bash code**

**go install github.com/golang/code**