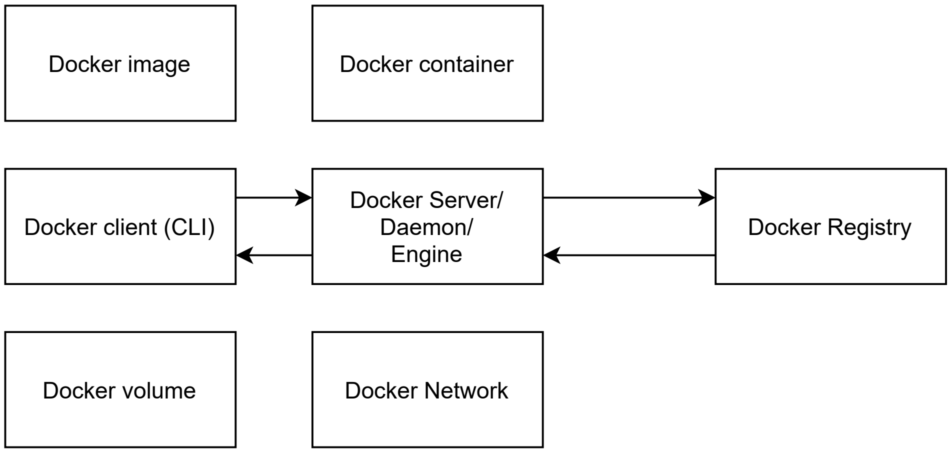
Developer machine

develop
test
build/package
containerize:
all platform/sdk deps
bundle as image
publish to a repo

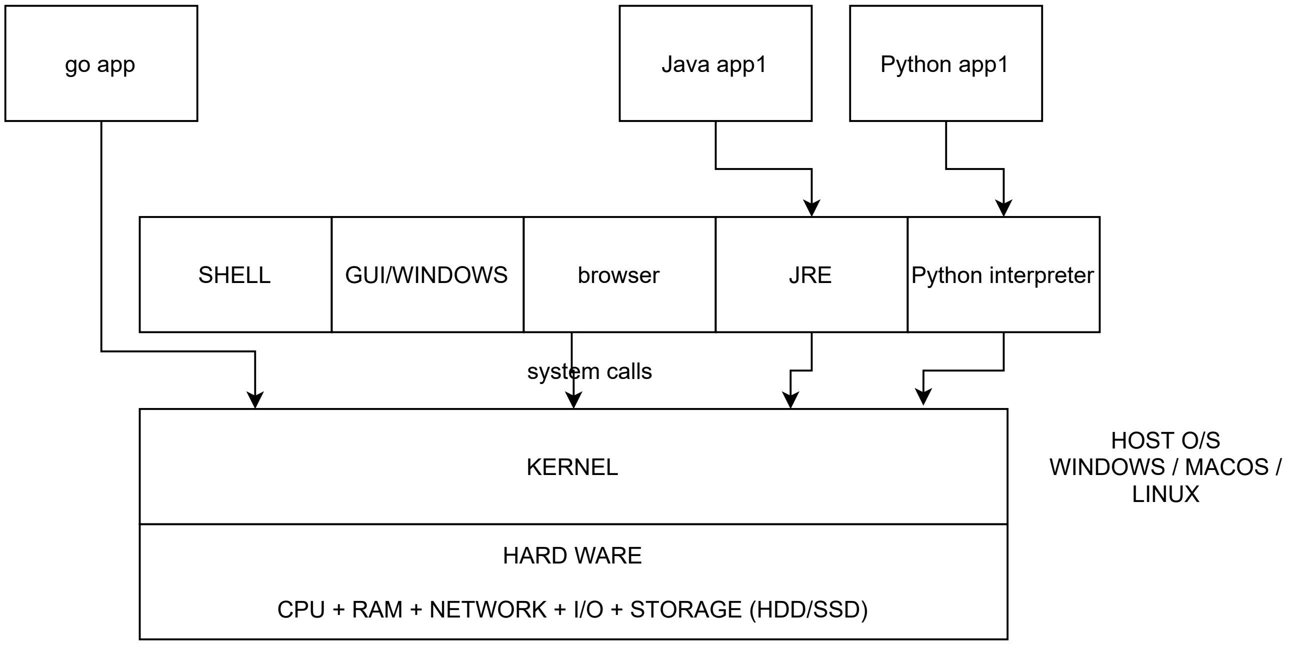
a new instance of that image can be created/deployed on local machines/cloud

Container orchestration:

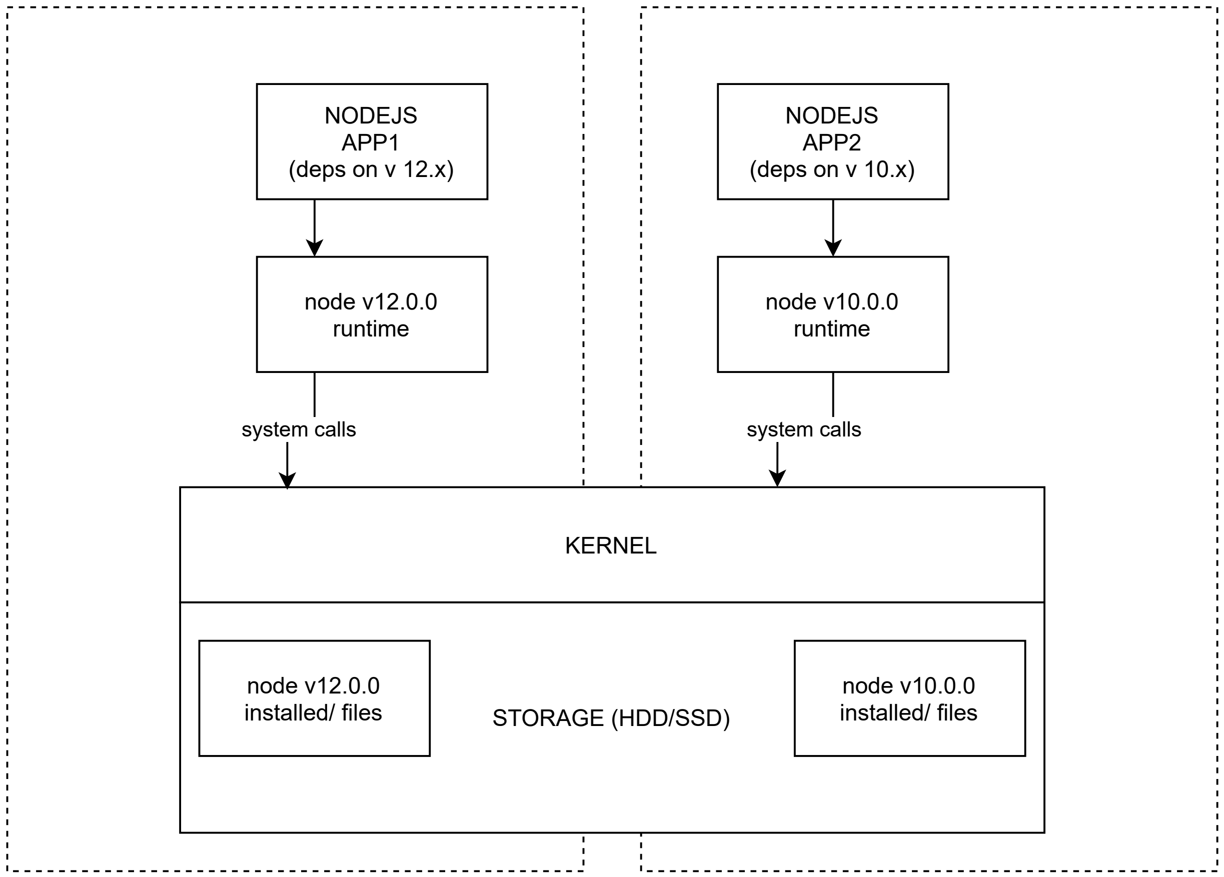
Docker swarm
Kubernetes

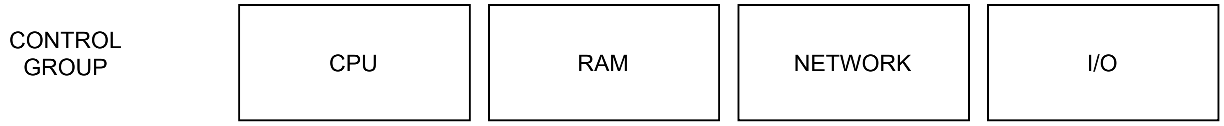


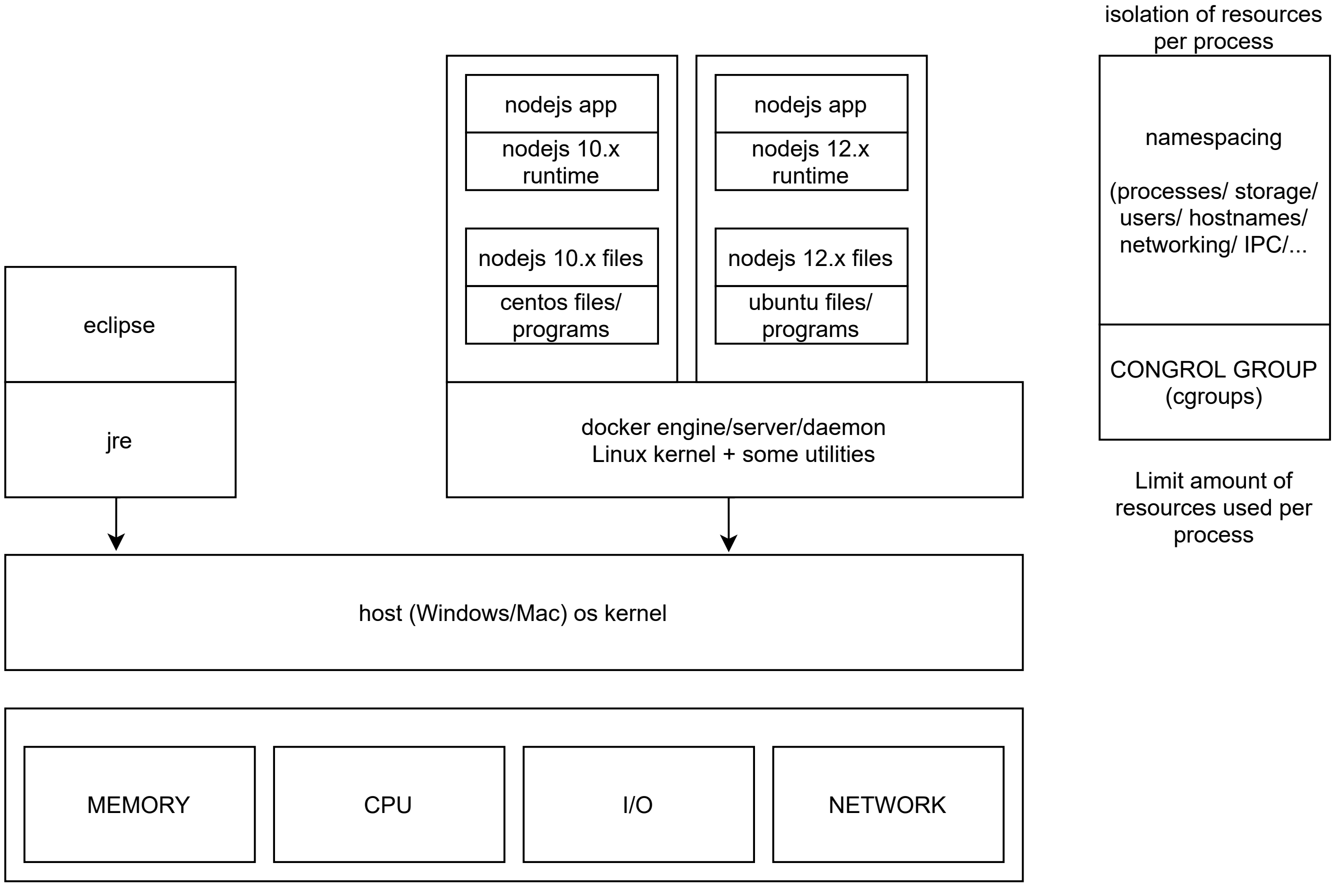
Docker image
Docker container
Docker client (CLI)
Docker Server/ Daemon/ Engine
Docker Registry
Docker volume
Docker Network



go app
Java app1
Python app1
SHELL
GUI/WINDOWS
browser
JRE
Python interpreter
system calls
KERNEL
HARD WARE
CPU + RAM + NETWORK + I/O + STORAGE (HDD/SSD)
HOST O/S
WINDOWS / MACOS / LINUX

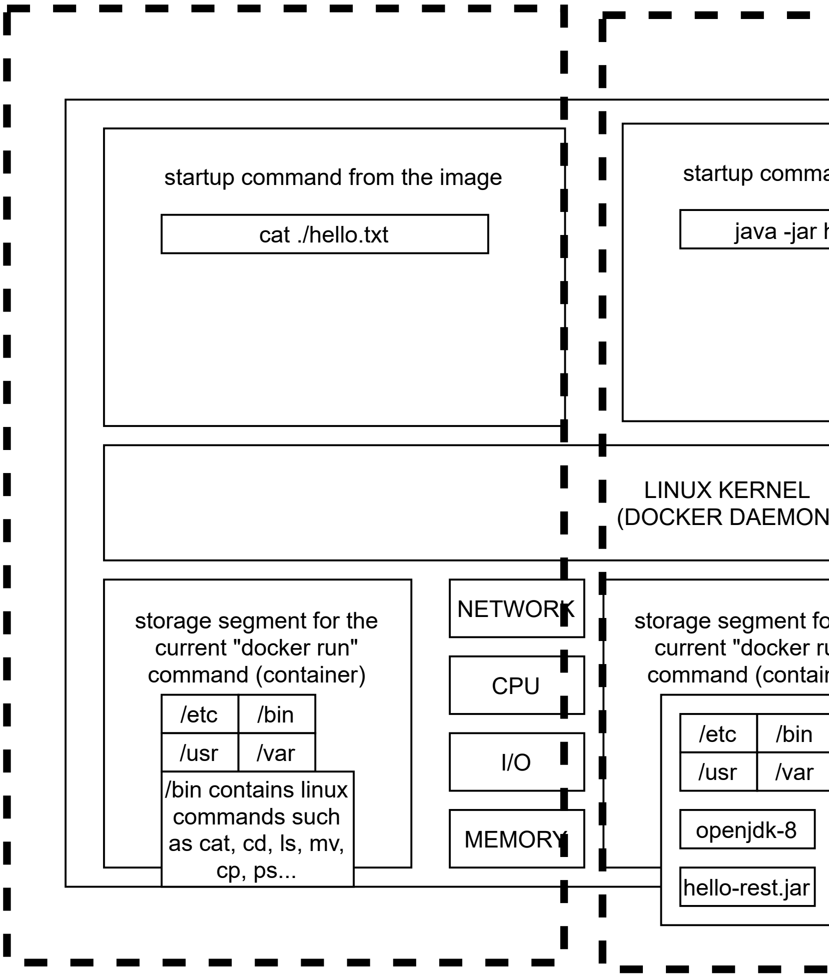

NODEJS
APP1
(deps on v 12.x)
NODEJS
APP2
(deps on v 10.x)
node v12.0.0
runtime
node v10.0.0
runtime
system calls
system calls
KERNEL
node v12.0.0
installed/ files
STORAGE (HDD/SSD)
node v10.0.0
installed/ files


CONTROL
GROUP
CPU
RAM
NETWORK
I/O


isolation of resources
per process
nodejs app
nodejs 10.x
runtime
nodejs 10.x files
centos files/
programs
nodejs app
nodejs 12.x
runtime
nodejs 12.x files
ubuntu files/
programs
namespacing
(processes/ storage/
users/ hostnames/
networking/ IPC/...
CONGROL GROUP
(cgroups)
eclipse
jre
docker engine/server/daemon
Linux kernel + some utilities
Limit amount of
resources used per
process
host (Windows/Mac) os kernel
MEMORY
CPU
I/O
NETWORK

docker image (hello-world)	
FS snapshot	starup command
hello	./hello

docker image learnwithvinod/whois	
FS snapshot	starup command
/etc /bin /usr /var /bin contains linux commands such as cat, cd, ls, mv, cp, ps...	cat ./vinod.txt



docker image learnwithvinod/hello-rest	
FS snapshot	starup command
/etc /bin /usr /var openjdk-8 hello-rest.jar	java -jar ./hell-rest.jar

docker image busybox	
FS snapshot	starup command
/etc /bin /usr /var	

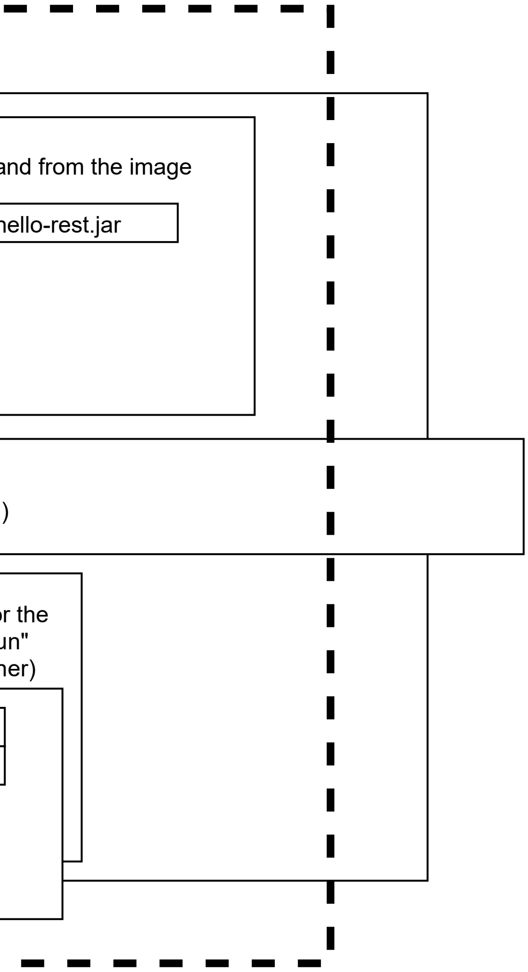
Few commands:

```
$ docker run busybox ping vinod.co
--> runs in foreground (has a unique container id)

$ docker ps
--> lists all the running containers

$ docker ps -a
--> lists both running as well as stopped container

$ docker stop <container-id-or-name>
--> graceful shutdown
$ docker kull <container-id-or-name>
--> force shutdown

$ docker images
$ docker image ls
--> lists all the pulled (downloaded) or built images

$ docker start <stopped-container-id-name>
--> start the container
```

docker run busybox < override-command >

More commands:

```
$ docker rm <container-id-name>
$ docker container rm <container-id-name>
--> removes the snapshot of the container

$ docker system prune
--> removes stopped containers/ unusednetworks/
dangling images/..
```


and from the image
hello-rest.jar
)
r the
un"
ner)

docker image busybox	
FS snapshot	starup command
/etc /bin /usr /var /bin contains linux commands such as cat, cd, ls, mv, cp, ps...	sh

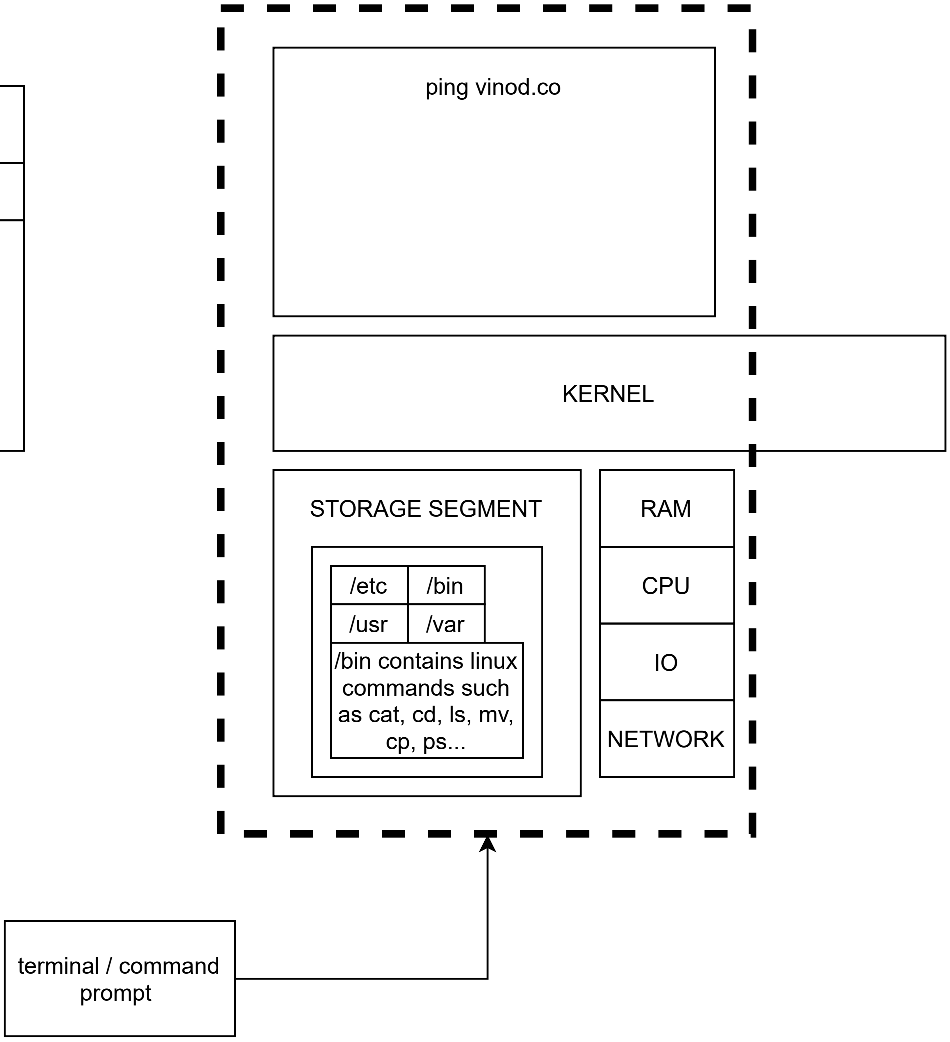
$ docker run busybox ping vinod.co

created a container with id: c39b47

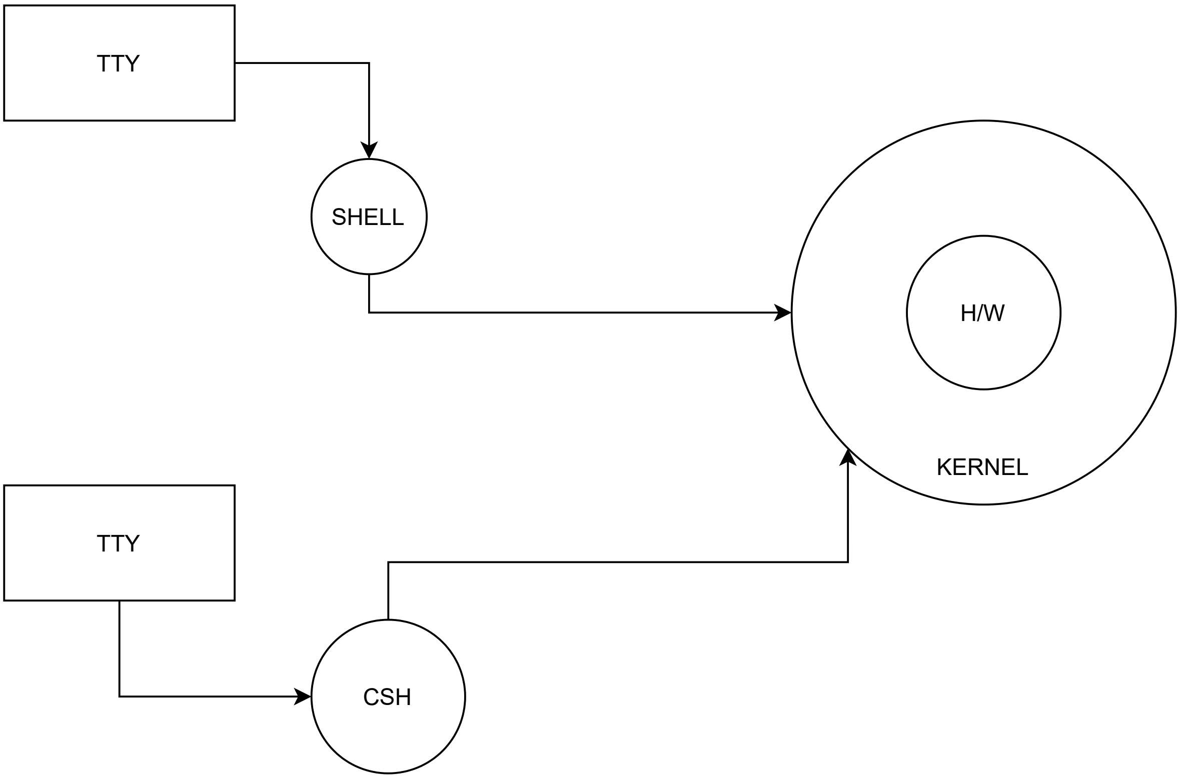
$ docker exec -it c39b47 sh

open a command interpreter (terminal)
to a running container

type exit to quit the shell

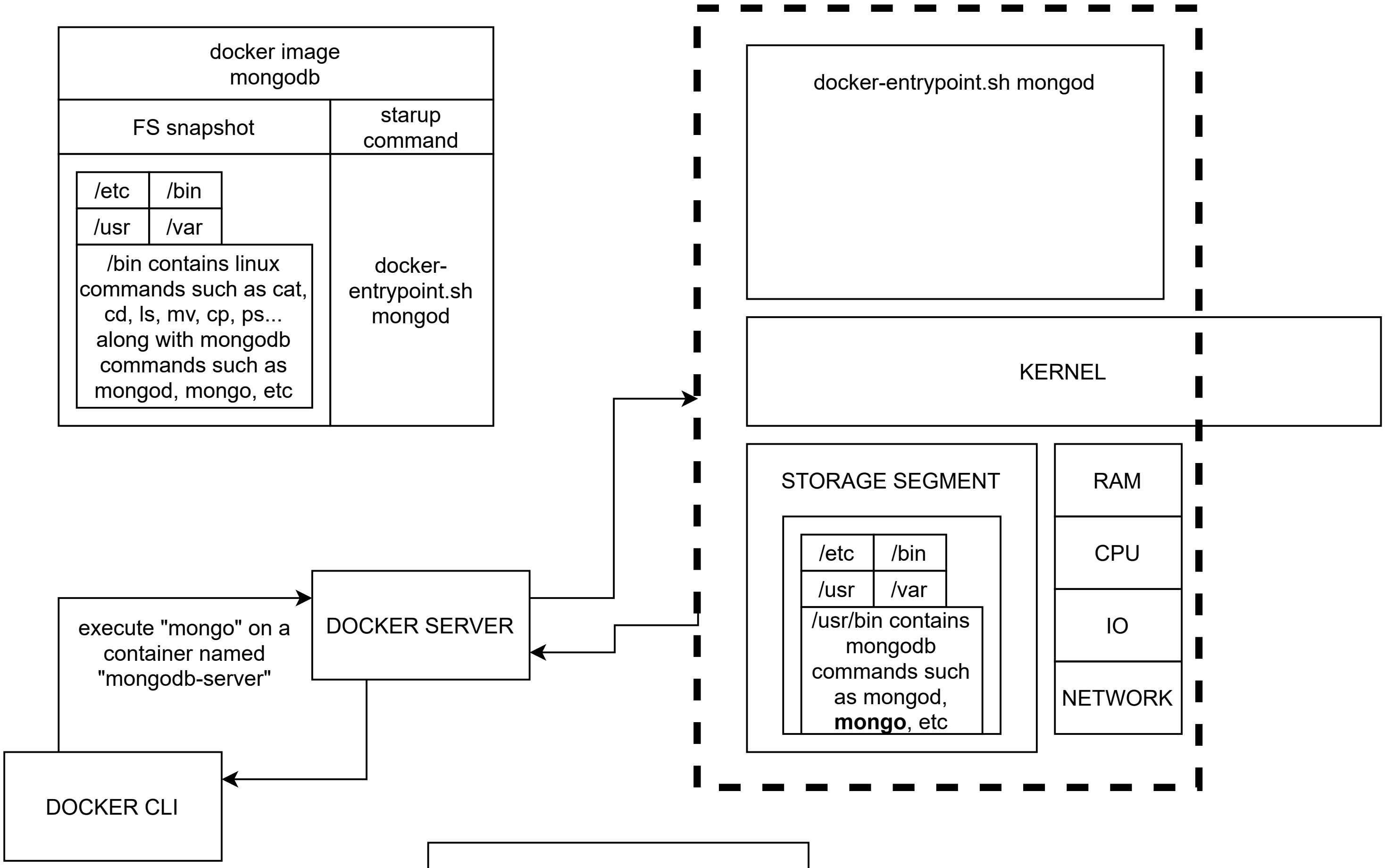


ping vinod.co
KERNEL
STORAGE SEGMENT
/etc
/bin
/usr
/var
/bin contains linux commands such as cat, cd, ls, mv, cp, ps...
RAM
CPU
IO
NETWORK
terminal / command prompt

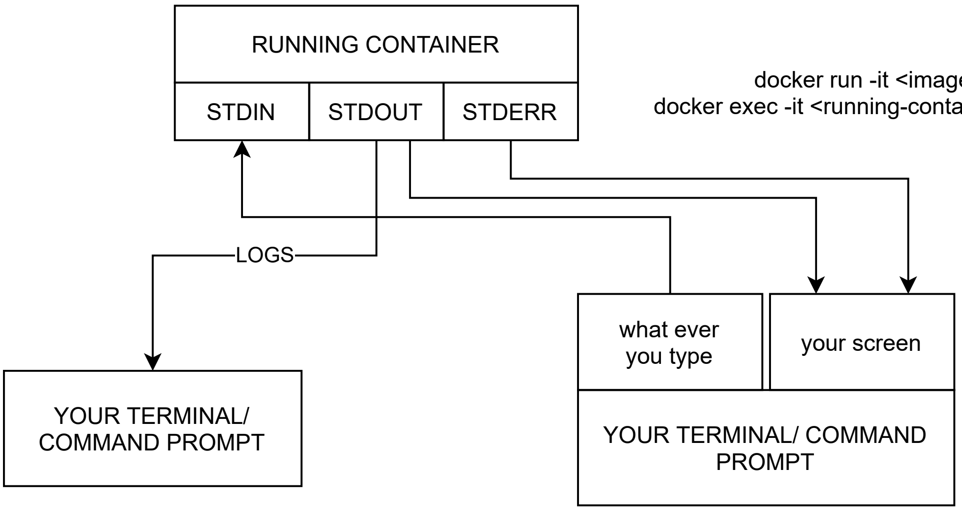


Notes:

```
$ docker run --rm <image-name>
--> removes the container when stopped

$ docker run --rm busybox ping -c 10 vinod.co


docker run <image-name>
is equivalent to
docker pull <image-name>
docker create <container-options> <image>
docker start <container-name-or-id>

docker run --detach --name mongodb-server mongo:latest
~
docker pull mongo:latest
docker create --name mongodb-server mongo:latest
docker start mongodb-server
```

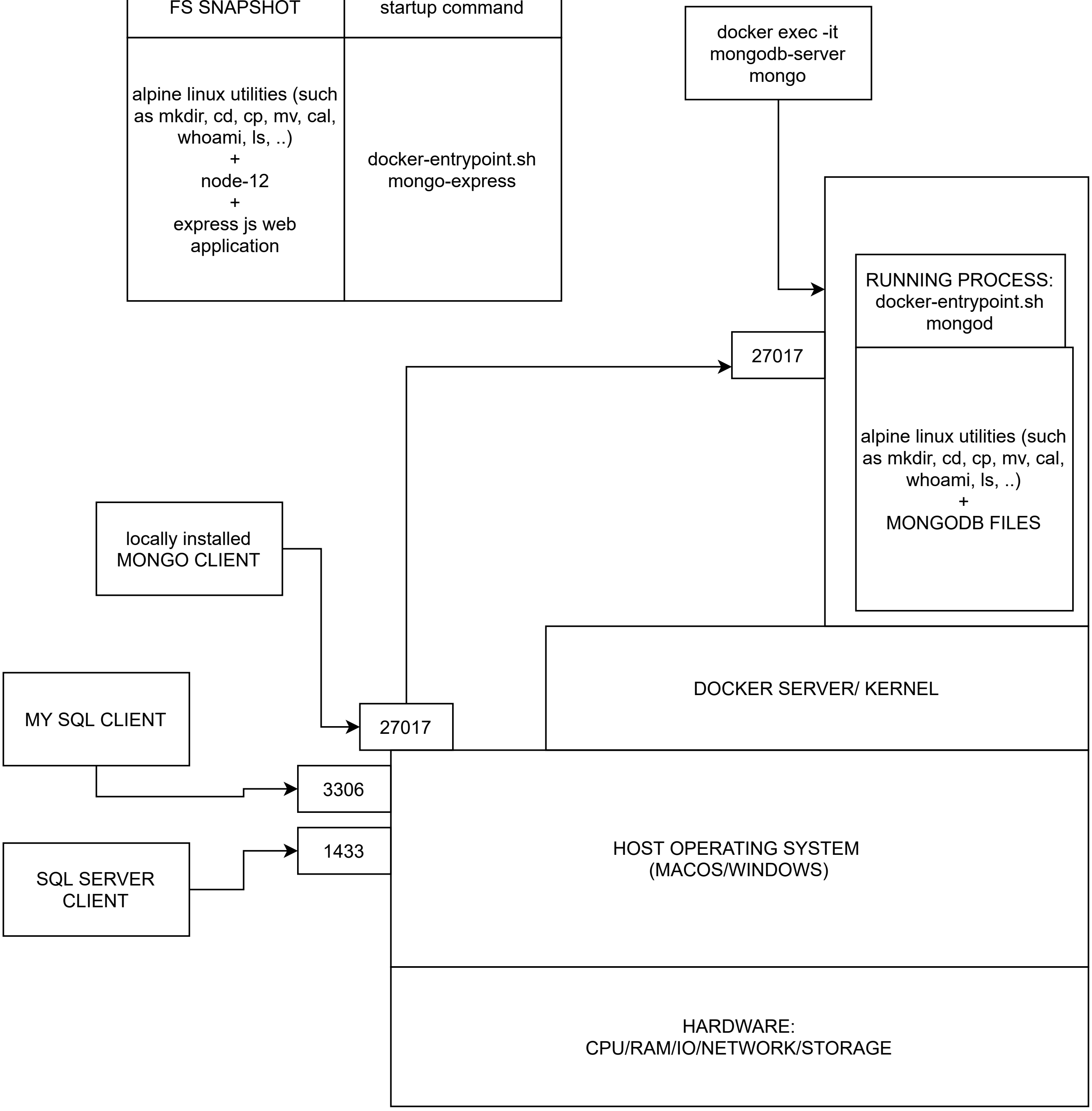


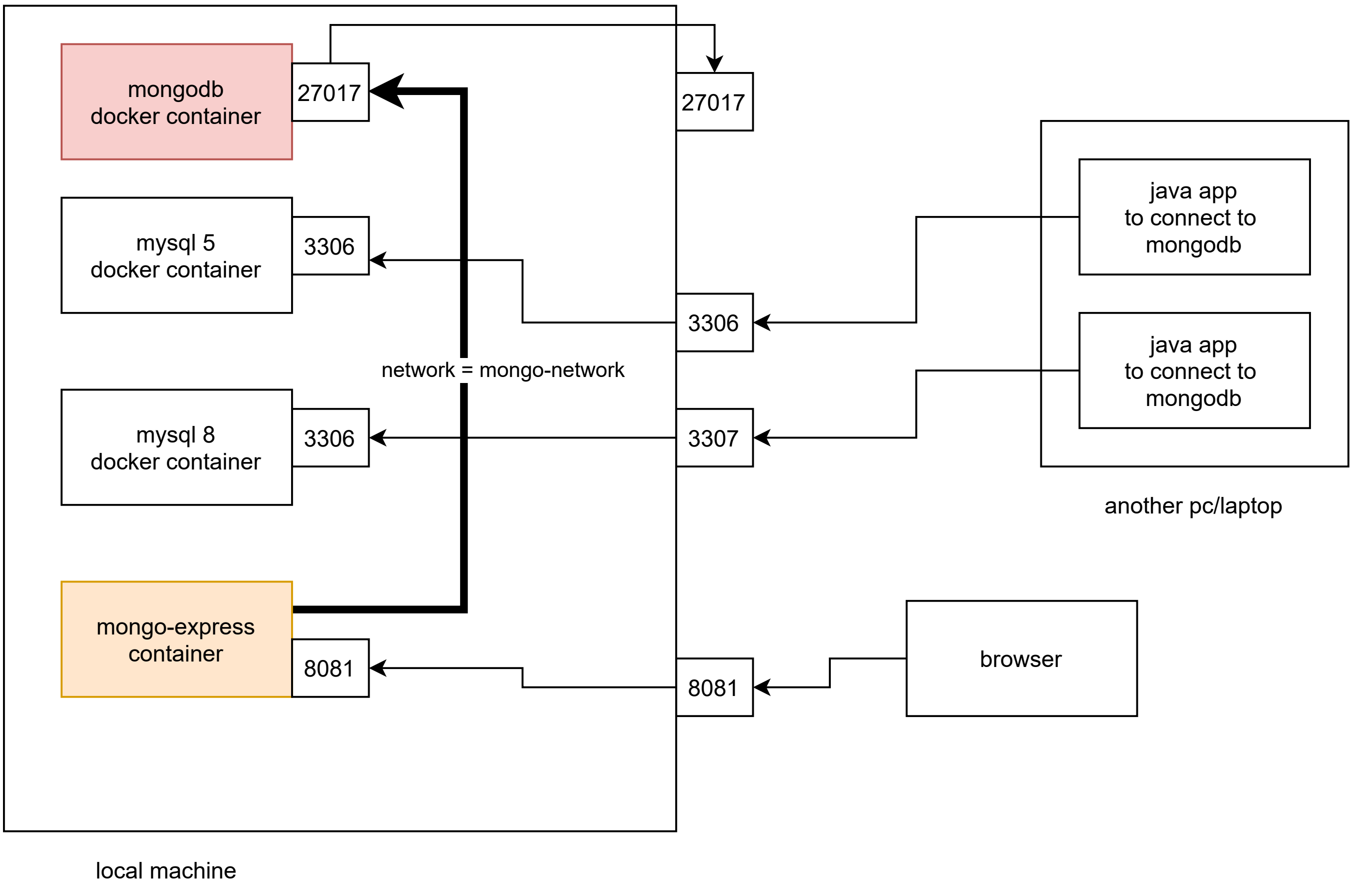
docker image
mongodb
FS snapshot
starup command
/etc
/bin
/usr
/var
/bin contains linux commands such as cat, cd, ls, mv, cp, ps... along with mongodb commands such as mongod, mongo, etc
docker-entrypoint.sh mongod
execute "mongo" on a container named "mongodb-server"
DOCKER SERVER
DOCKER CLI
docker-entrypoint.sh mongod
KERNEL
STORAGE SEGMENT
/etc
/bin
/usr
/var
/usr/bin contains mongodb commands such as mongod, mongo, etc
RAM
CPU
IO
NETWORK



```
docker run -it <image-name-id>
docker exec -it <running-container-id> <command>
```

mongo-express image	
FS SNAPSHOT	startup command
alpine linux utilities (such as mkdir, cd, cp, mv, cal, whoami, ls, ..) + node-12 + express js web application	docker-entrypoint.sh mongo-express


docker exec -it mongodb-server mongo
RUNNING PROCESS: docker-entrypoint.sh mongod
alpine linux utilities (such as mkdir, cd, cp, mv, cal, whoami, ls, ..)
+
MONGODB FILES
27017
locally installed MONGO CLIENT
DOCKER SERVER/ KERNEL
MY SQL CLIENT
27017
3306
1433
SQL SERVER CLIENT
HOST OPERATING SYSTEM (MACOS/WINDOWS)
HARDWARE: CPU/RAM/IO/NETWORK/STORAGE



Docker networking commands:

```
$ docker network ls
$ docker network rm <network-name-id>

$ docker network create <network-name>
$ docker network create mongo-network
```

To add a running to container to a network:

```
$ docker network connect <network-name> <container-id-or-name>
$ docker network connect mongo-network mongodb-server
```

To add a new container (run command) to a network:

```
$ docker run --network <network-id-name> -- ... .. ..
$ docker run --network mongo-network \
  -e ME_CONFIG_MONGODB_SERVER=mongodb-server \
  -p 8081:8081 \
  --name mongo-webclient mongo-express
```