**BASIC COMMANDS**

fuser -k 80/tcp //stop a port

docker rm $(docker ps -a -q)

sudo iptables -t nat -L -n

docker rmi $(docker images -q)

docker rmi -f $(docker images -q)

docker rmi $( docker images -q -f dangling=true)

docker volume ls -qf dangling=true | xargs -r docker volume rm //remove dangling volumes

docker pull busybox

docker images

docker run busybox

docker run busybox echo "hello from busybox"

docker ps

docker ps -a

docker run -it busybox sh

docker run --help

docker rm -v $(docker ps -a -q -f status=exited)

docker rmi

docker inspect -f '{{.Name}} - {{.NetworkSettings.IPAddress }}' $(docker ps -aq) //get ip address of host of all containers

docker stop $(docker ps -a -q) //stop all docker container

docker rm -v $(docker ps -a -q) //remove all docker container

docker rmi $(docker images | grep "^<none>" | awk '{print $3}') //removes all none images

docker logs -f bdb //first three letters of the container id - to view logs

docker stats CONTAINER ID displays the CPU, memory, and network I/O for each image.

docker inspect CONTAINER ID displays the configuration of the image.

docker info

docker volume ls

docker push lygado/docker-tomcat : push image to docker hub

docker network ls

**//login to running container**

docker exec -i -t <container-id> bash //debugging

sudo docker exec -i -t Nginx bash

cat /etc/hosts

Running env within your Nginx container

**LINKING MULTIPLE CONTAINERS**

docker run -p 9999:8080

--link otherContainerA --link otherContainerB

-v /Users/$USER/.m2/repository:/home/user/.m2/repository

toptal/pingpong

**To run apt-get update and install within the docker file, need to change :**

/etc/NetworkManager/NetworkManager.conf:

#dns=dnsmasq

sudo service network-manager restart

sudo service docker restart

**Sample Dockerfiles**

----JAVA----------

FROM java:8

COPY PingPong.java /

RUN javac PingPong.java

EXPOSE 8080

ENTRYPOINT ["java"]

CMD ["PingPong"]

docker build -t toptal/pingpong .

docker run -d -p 8081:8081 toptal/pingpong

curl http://localhost:8080/ping

docker build -t="dockerfile/java" github.com/dockerfile/java

--simple java program--

FROM java:8

WORKDIR /app

COPY src/HelloWorld.java /app/

RUN javac HelloWorld.java

CMD ["java", "HelloWorldApp"]

docker build -t helloworld .

docker run helloworld

---TOMCAT-----

docker run -d -p 8080:8080 tomcat //start tomcat

docker run -d tomcat

docker logs -f bdb //first three letters of the container id - to view logs

docker inspect -f '{{.Name}} - {{.NetworkSettings.IPAddress }}' $(docker ps -aq) //get ip address of host of all containers

curl 172.17.0.2:8080

---GOLANG-----

FROM golang:1.6

WORKDIR /app

COPY src/hello.go /app/

#Build file for hello

RUN go build hello.go

#COPY hello /app/

ENTRYPOINT ["./hello"]

docker build -t src/hello .

docker run /hello

---client server

FROM golang:1.6

WORKDIR /app

COPY src/server.go /app/

#Build file for hello

RUN go build server.go

#COPY server /app/

EXPOSE 3333

ENTRYPOINT ["./server"]

docker build -t goserverclient .

docker run -i -t -p 3333:3333 goserverclient

echo "Hello server" | nc <ip> 3333

git config --global --unset http.proxy

git config --global --unset https.proxy

three ways to configure networking for Docker containers that complement each other: Link, Port, and Pipework.

Link : env variables for the ports are generated for the docker container that can be used to connect automatically for other env variables

Port : Expose services to host port

Pipework :

GOLANG

FROM golang:1.6

# download and build protoc

RUN go get -u github.com/golang/protobuf/proto

# install dependencies (e.g. protoc-gen-go, grpc)

RUN go get -u github.com/golang/protobuf/protoc-gen-go

RUN go get -u google.golang.org/grpc

RUN go get -u golang.org/x/net/context

WORKDIR /app

COPY client.go /app/

COPY UUID.pb.go /app/protoFile/UUID.pb.go

COPY UUID.proto /app/protoFile/UUID.proto

#Build file

RUN go build client.go

#COPY client /app/

EXPOSE 50051

ENTRYPOINT ["./client"]

docker run --rm -v $PWD:/app -w /app treeder/go vendor

docker run --rm -v $PWD:/app -w /app treeder/go build

**DOCKER REGISTRY**

**Back up of mounted data :**

docker run --rm --volumes-from data-container -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /data-store

**Delete Image from private registry:**

Refer Feature Request - Delete.docx

**Change Folder permissions :**

chcon -Rt svirt\_sandbox\_file\_t /var/db

**To make a mounted volume read only : in docker compose file**

volumes:

- ./data:/data:ro

**To share a mounted volume across different containers: in docker compose file**

volumes\_from:

- registry //service name which contains the mounted volume

**External volumes** : access to volumes of containers running through different docker-compose yml file

Removing volumes when removing containers : docker rm -v jisto\_server

If volumes are not removed, the bridge for the service can’t be deleted

**SAMPLE configurations**

docker run -d -p 5000:5000 --name registry registry:2

docker pull ubuntu && docker tag ubuntu localhost:5000/ubuntu

docker push localhost:5000/ubuntu

docker pull localhost:5000/ubuntu

docker stop registry && docker rm -v registry

docker run -d -p 5000:5000 --restart=always --name registry registry:2

**Authentication and authorization**

entry /etc/hosts for non 127.0.1.1 - myregistrydomain.com

openssl req -newkey rsa:4096 -nodes -sha256 -keyout auth\_server/ssl/server.key -x509 -days 365 -out auth\_server/ssl/server.crt -subj /CN=myregistrydomain.com

sudo mkdir /etc/docker/certs.d/myregistrydomain.com:5000

sudo cp auth\_server/config/server.crt /etc/docker/certs.d/myregistrydomain.com:5000/ca.crt

copy certificate ca.crt to usr/local/share/ca-certificates/ca.crt --//for client side

sudo update-ca-certificates

docker run --entrypoint htpasswd registry:2 -Bbn admin admin > auth\_server/config/htpasswd

sudo service docker restart //service docker stop && service docker start

docker-compose.yml :

registry:

restart: always

image: registry:2

ports:

- 5000:5000

volumes:

- ./docker\_registry/data:/var/lib/registry

- ./auth\_server/ssl:/ssl

- ./auth\_server/config:/auth

environment:

REGISTRY\_HTTP\_TLS\_CERTIFICATE: /ssl/server.crt

REGISTRY\_HTTP\_TLS\_KEY: /ssl/server.key

REGISTRY\_HTTP\_TLS\_CLIENTCAS\_0: /ssl/ca.crt

REGISTRY\_STORAGE\_FILESYSTEM\_ROOTDIRECTORY: /var/lib/registry

REGISTRY\_AUTH\_HTPASSWD\_PATH: /auth/htpasswd

REGISTRY\_AUTH\_HTPASSWD\_REALM: Registry Realm

# extra\_hosts:

# - "myregistrydomain.com:127.0.0.1" //makes an entry in /etc/hosts in the container running this

sudo mkdir -p /etc/docker/certs.d/myregistrydomain.com:5000

sudo scp root@myregistrydomain.com:/ssl/ca.crt \

/etc/docker/certs.d/myregistrydomain.com:5000/ca.crt

**Running CURL with ssl certs**

curl --cacert auth\_server/ssl/ca.crt -X GET https://admin:admin@myregistrydomain.com:5000/v2/\_catalog

curl --cacert auth\_server/ssl/ca.crt -X GET https://admin:admin@myregistrydomain.com:5000/v2/busybox/tags/list

**#generating ssl certificate**

openssl req -newkey rsa:4096 -nodes -sha256 -keyout certs/domain.key -x509 -days 365 -out certs/domain.crt

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt

openssl x509 -outform der -in server.pem -out server.crt

docker run -d -p 5000:5000 --restart=always --name registry \

-v `pwd`/certs:/certs \

-e REGISTRY\_HTTP\_TLS\_CERTIFICATE=/certs/domain.crt \

-e REGISTRY\_HTTP\_TLS\_KEY=/certs/domain.key \

registry:2

**#authentication**

mkdir auth

docker run --entrypoint htpasswd registry:2 -Bbn testuser testpassword > auth/htpasswd

docker run --entrypoint htpasswd registry:2 -Bbn test 123 > htpasswd

docker run -d -p 5000:5000 --restart=always --name registry \

-v `pwd`/auth:/auth \

-e "REGISTRY\_AUTH=htpasswd" \

-e "REGISTRY\_AUTH\_HTPASSWD\_REALM=Registry Realm" \

-e REGISTRY\_AUTH\_HTPASSWD\_PATH=/auth/htpasswd \

-v `pwd`/certs:/certs \

-e REGISTRY\_HTTP\_TLS\_CERTIFICATE=/certs/domain.crt \

-e REGISTRY\_HTTP\_TLS\_KEY=/certs/domain.key \

registry:2

docker login myregistrydomain.com:5000

#docker run -d -p 5000:5000 --restart=always --name registry --dns=192.168.0.1 registry:2

docker tag hello-world myregistrydomain.com:5000/hello-world

docker push myregistrydomain.com:5000/hello-world

docker run -d -p 5000:5000 \

-e REGISTRY\_STORAGE\_FILESYSTEM\_ROOTDIRECTORY=/var/lib/registry \

-e REGISTRY\_AUTH=token \

-e REGISTRY\_AUTH\_TOKEN\_REALM=https://myregistrydomain.com:5001/auth \

-e REGISTRY\_AUTH\_TOKEN\_SERVICE="Docker registry" \

-e REGISTRY\_AUTH\_TOKEN\_ISSUER="Auth Service" \

-e REGISTRY\_AUTH\_TOKEN\_ROOTCERTBUNDLE=/ssl/server.pem \

-v /root/auth\_server/ssl:/ssl \

-v /root/docker\_registry/data:/var/lib/registry \

--restart=always \

--name registry registry:2

docker run -d --name docker\_auth -p 5001:5001 \

> -v `pwd`/auth\_server/config:/config:ro \

> -v /var/log/docker\_auth:/logs \

> --restart=always \

> -v `pwd`/auth\_server/ssl:/ssl cesanta/docker\_auth /config/auth\_config.yml

curl --cacert nginx/certs/devdockerCA.crt -X GET https://jeavioregistry.com:5043/internal/images?isJSON=true

curl --cacert nginx/certs/devdockerCA.crt -X GET https://jeavioregistry.com:5043/internal/images

**Not working :**

maintenance:

uploadpurging:

enabled: true

age: 168h

interval: 24h

dryrun: false

readonly:

enabled: false

REGISTRY\_STORAGE\_MAINTENANCE\_UPLOADPURGING\_ENABLED="true"