Contents

[Tutorials 2](#_Toc522430988)

[Docker 2](#_Toc522430989)

[Components of Docker 2](#_Toc522430990)

[Docker Image 2](#_Toc522430991)

[Docker container 2](#_Toc522430992)

[Docker registry 2](#_Toc522430993)

[How to install Ubuntu in Oracle virtual box 3](#_Toc522430994)

[Installing docker on Ubuntu 3](#_Toc522430995)

[Quick Ubuntu commands 3](#_Toc522430996)

[Installing docker images 4](#_Toc522430997)

[Difference between VM and Docker 4](#_Toc522430998)

[Quick Docker commands and explanation 5](#_Toc522430999)

[Create docker images 6](#_Toc522431000)

[Docker understanding 6](#_Toc522431001)

[How to create a docker image and post it to docker registry 7](#_Toc522431002)

[Docker experience with Jelastic 7](#_Toc522431003)

[Installing postgresql with plv8 extension 2.3.3 in jelastic 8](#_Toc522431004)

# Tutorials

<https://www.tutorialspoint.com/docker>

Don’t follow the installations from this url.

# Docker

Docker is a container. It makes use of host operating system. You can consider it as an application containing other applications. It uses a process of the operating system.

Say for example you require Ubunto with tomcat and java vm. You can take the base docker image as Ubuntu from Docker hub which is a public library of base images. Upon this base image you can deploy Tomcat with JVM. You save that as docker image. You can use this Docker image multiple times in production, staging and dev as Docker container.

You can also develop your applications in form of Docker containers and run anywhere. Official site is <https://www.docker.com/>

# Components of Docker

* **Docker for Mac** − It allows one to run Docker containers on the Mac OS.
* **Docker for Linux** − It allows one to run Docker containers on the Linux OS.
* **Docker for Windows** − To run Docker containers on the Windows OS.
* **Docker Engine** − Used for building Docker images and creating Docker containers.
* **Docker Hub** − Registry which is used to host various Docker images.
* **Docker Compose** − This is used to define applications using multiple Docker containers.

# Docker Image

It is the basic template. It usually contains OS and several applications. In docker hub you can find many such images with verities of OS and softwares.  <https://hub.docker.com/>.

# Docker container

This is an implementation of image which can be read or written into. Docker uses union-file-system; any changes which are made to the container are added to new layer above base image. Docker container is an instance of Docker image.

# Docker registry

This is repository of Docker images. There is public and private one. Public repository is called Docker hub. We can push and pull our own images in public repository.

# How to install Ubuntu in Oracle virtual box

Download virtual box. Download Ubuntu ISO image. Start Virtual Box, settings->storage point the optical device to ISO image. This will install Ubuntu in virtual box. Set the processor core to 4 or maximum. Increase the memory to 2GB otherwise it will be very slow.

# Installing docker on Ubuntu

Docker can be installed from Ubuntu repository or Docker repository. Installing from Ubuntu repository is simple. Do it like sudo apt install docker.io. But the Ubuntu repositories do not contain the latest Docker. The latest is available in docker repositories.

Installing from docker hub is somewhat complex and I installed it following the site <https://docs.docker.com/install/linux/docker-ce>/ubuntu

Uninstall docker

sudo apt-get purge docker-ce

You need to remove the dependencies also.

sudo rm -rf /var/lib/docker

docker info and docker version commands can tell about docker installation.

# Quick Ubuntu commands

Update

sudo apt-get update

Multi line install

sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

software-properties-common

Remove

sudo apt-get remove docker docker-engine docker.io

Vim editor: esc :wq to write and quit

The right Ctrl button is called HOST. Some keys work in combination of HOST.

CTRL+H 🡪 close down

CTRL+P 🡪 pause

CTRL+F 🡪 Full screen

Unlock the ubuntu screen: Press a key

CTRL+C 🡪 Scale mode come out

# Installing docker images

sudo docker pull Jenkins

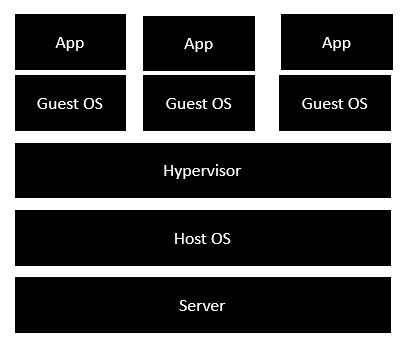
To run an image, say hello world

Sudo docker run hello-world

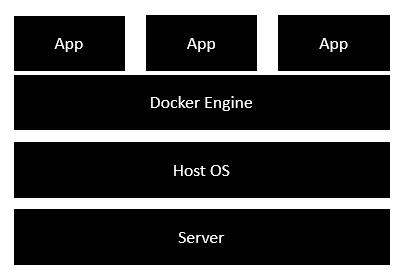
If the image in not there in the machine it will be pulled from docker hub and run.

# Difference between VM and Docker

Conventional VM



Docker



# Quick Docker commands and explanation

Docker run command downloads the image if it is not already there.

docker run hello-world

Creates an instance of image hello-world.

sudo docker run centos –it /bin/bash

Creates an instance of image centos and runs it in interactive mode (-it). /bin/bash is to run bash when the centos is up and running.

docker images

Displays all docker images in system.

sudo docker rmi 7a86f8ffcb25

Removes docker image with image id as above. You can get the image id from docker images command. Sometimes you may have to put -f switch to force deletion of image.

docker inspect centos

Provides detailed information of the centos image.

docker ps

Shows all running containers

Docker ps -a shows all containers whether running or stopped.

docker stop ContainerID

sudo docker stop 9f215ed0b0d3

stops a container

docker rm ContainerID

removes a container.

docker stats ContainerID

Shows CPU and memory usage of container

docker pause ContainerID

Pause a container

docker unpause ContainerID

unpause container

docker kill ContainerID

Kill a container

service docker stop

Stop the docker daemon

service docker start

start a docker daemon

sudo docker run –it centos /bin/bash

Above command starts a centos container in interactive mode. Now whatever you type on console goes to the container. To come out of interactive session in Ubunto on Oracle virtual box press left CTRL P + Q. Remember right CTRL+P means pause. Right CTRL+P is host command for the virtual box.

If you come out of the container then the container would be destroyed. To enter and exit a container cleanly use nsenter command.

# Create docker images

You can create docker images by using docker files. A docker file is a text file with instructions as how to make a docker image. It should be strictly named as Dockerfile.

# Docker understanding

Once you install docker in any operating system you become concerned with only the docker commands and not with OS. Docker commands run same in all OS’s. Basically, docker runs in Linux. So, in another OS’s docker simulates running on Linux. Some example:

**$ docker image pull nginx**

**$ docker container run -d --name nginx-test -p 8080:80 nginx**

1st command: pulls nginx.

2nd command: runs nginx docker image as a container in background / detached mode with name as nginx-test, your machine port as 8080 mapped to container port of 80. You can now browse at port 8080 as <http://localhost:8080>. The command and results are same in all OS’s.

**$ docker container stop nginx-test**

**$ docker container rm nginx-test**

Above commands stop and remove nginx container.

# How to create a docker image and post it to docker registry

Method 1: from Dockerfile: Successfully tested

* Create a Dockerfile in a new folder and write your commands into it using nano.

FROM postgres

EXPOSE 5432

* Cd to new folder
* sudo docker build -t capitalch/xpostgres:latest . (check the dot at last)
* sudo docker push capitalch/xpostgres:latest
* To remove the image

Sudo docker rmi -f capitalch/xpostgres:latest

Method 2: from existing image

* Create a docker image

Sudo docker pull postgres

* Create a tag out of this image. Get the image-id by ‘sudo docker images’, copy on screen the image id.

Sudo docker tag <image-id> capitalch/xpostgres1:latest

# Docker experience with Jelastic

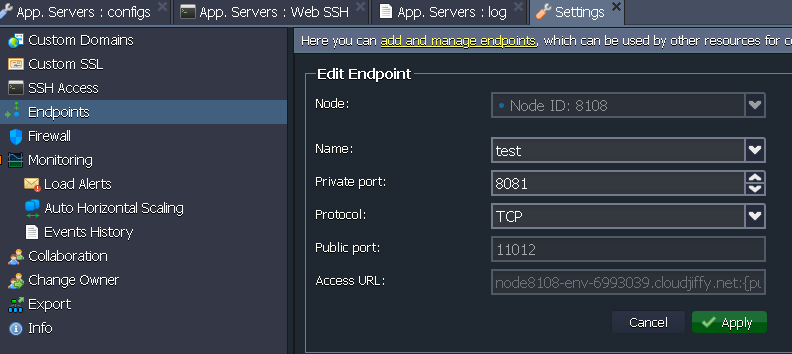
I found JElastic a good company for providing docker hosting. Purchased account in app.cloudjiffy.com with capitalch… and …123.

With OTB node.js and postgresql it ran fine. Then I tried with custom node.js docker from docker hub. Install the new node.js environment from docker hub.

1. **Steps to create a custom node.js app from docker hub in cloudjiffy.com**

* Opened configs and went to Root folder. Created a new folder with name node and uploaded all node.js files in this folder.
* Used SSH🡪 cd /home/node. Originally it was in Root folder by default when you login through SSH. You are logged in as root user with all rights available. This is equivalent to SUDO in UBUNTU.

**node server**

* My application of hellow world node application was using port 8080. When I gave default url of environment created it worked fine.
* **Now I wanted to run** app in port 8081. I changed the app for using port 8081 and uploaded it to /home/node folder. Then I created an endpoint through **settings icon in environment** row. I filled up following:
* ****

The public port and Access URL was auto filled by cloudjiffy.

The application worked

## Steps to create a new postgresql in cloudjiffy from docker hub:

* Select docker tab in a new environment
* Select postgres and create from docker hub. Use new IP.
* In pgadmin create new server, give new ip, use user as postgres and password as received in email.

It worked. I was able to connect the postgresql database.

## Installing postgresql with plv8 extension 2.3.3 in jelastic

I used [waldo2188](https://hub.docker.com/u/waldo2188/)/[postgres-plv8](https://hub.docker.com/r/waldo2188/postgres-plv8/)

* For that I searched with name waldo2188 in docker registry and added the image and created the environment. Make sure that you create a public IP.
* I got a mail from cloudjiffy giving the password for root user.
* I created a new server in pgadmin giving the public ip and password.
* In database I gave sql command:
  + create extension if not exists plv8
  + select plv8\_version();

plv8 version 2.3.3 was available of April 2018 which supported ES6.

Next I am going to use [**clkao**](https://github.com/clkao)**/**[docker-postgres-plv8](https://github.com/clkao/docker-postgres-plv8)

Clkao did not work. It tries to install plv8 extension from make utility which compiles the source files. In several Ubutu installations of 18.04 it failed after an hour or so. Then I tried the dockerfile of waldo2188/postgres-plv8. I made some very minor changes in the Dockerfile and build the image from Dockerfile through command sudo docker build -t capitalch/xpostgres:latest . (check the dot at last) .

After few hours I was able to create the build 2.3.4 of plv8 in docker in postgresql and I uploaded the new docker image through docker command sudo docker push capitalch/xpostgres:latest. Thereafter I created the environment in cloudjiffy and used the said docker image to create postgres. Then in local machine I used the pgAdmin and connected to the postgres database in cloudjiffy. Then I gave command create extension if not exists plv8; select plv8\_version(); It all worked fine.

## Automated build of docker registry

I create an automated build in docker registry from my GitHub account **xpostgres-plv8**. When I pushed in GitHub the automated build started. It ran for 2 hours and ended with error. I saw the log file abruptly stopped. Then I created the same account in cloud.docker.com instead of hub.docker.com. There I created and automated build and manually triggered it. It ran for 2 + hours and a build was created successfully. It was automatically ported to hub.docker.com in capitalch/xpostgres-plv8.

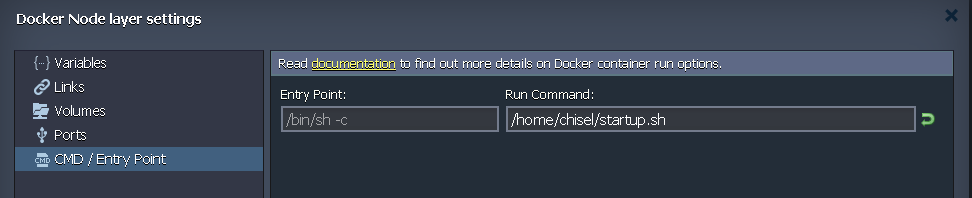
## Installing your multiple node applications in node docker image

Created a docker environment named chisel from a node docker image. I realized that node docker image was quite flexible. You would get extensive logs in run.log file.

### Running a script at startup

Method 1: simplest: just mention in the entry point

* Created a file startup.sh in a new folder in /home/chisel. Content of file is ‘echo starting up’.
* In CMD entry point I wrote ‘/home/chisel/startup.sh’
* Restarted the server and checked log file run.log. It showed message ‘starting up’. Cool.



**Method 2**: crontab -e and then putting the script through editor did not work. It was able to start the /home/startup.sh but node server was never started. It appears that the script is to run at the end but it does not work at end.

**Method 3**: /etc/rc.local

I put /home/startup.sh in the above file and it worked. You can see the log in syslog.log and not in run.log. rc.local runs at end hence multiple node.js servers are activated from startup.sh. cool!!!

**Method 4**: using systemd: file in /etc/systemd/system/start.service

Systemd is new and it is responsible for starting / stopping / restarting programs for you

Following file worked but I was only able to start single node.js server using this approach

[Unit]

Description=startup service for multiple node applications

#Requires=After=mysql.service # Requires the mysql service to run first

[Service]

Type=oneshot

WorkingDirectory=/home/cosmos

ExecStart=/usr/local/bin/node server.js

WorkingDirectory=/home/sundry-sites/capital-seo

ExecStart=/usr/local/bin/node local.js

StandardOutput=syslog

StandardError=syslog

SyslogIdentifier=nodejs-example

[Install]

WantedBy=multi-user.target

Use this url for system approach

<https://www.axllent.org/docs/view/nodejs-service-with-systemd/>

Using systemd approach you cannot call another sh file which opens many node servers. You can only run one node server from system approach

Note: always remember to keep the line

#!/bin/bash

as the first line of your .sh script file.

### Install a node.js application in node.js docker image

* I zipped the cosmos application required files in cosmos-node.zip file and uploaded in a new directory named ‘cosmos’ in home directory.
* Using command prompt I unzipped the .zip file (unzip cosmos-node.zip).
* Npm install in cmd prompt after cd /home/cosmos
* ‘Node server’ starts the application. If you put default port as 80 then in browser chisel.cloudjiffy.net/api/tear I am able to use tear which is part of cosmos. If any other port say 8080 or 3000 is there then you have to map the port with endpoint. The new url with mapped endpoint works.



* In linux if you give a command with ‘&’ at last, the command is executed as service and your terminal is free. ‘ps’ command shows all running processes. You can kill a process by ‘kill process id’. Process id can be found from ps command. Here ps command will show that node server is running.
* If you restart cloudjiffy server then you need to again run the ‘node server.js’ to get it up again. So to avoid giving command ‘node server.js’ every time when server is start up you need to put the startup script for node server as explained before.
* I saw ‘top’ command is more useful than ‘ps’ command. ‘ps’ does not show all the processes but ‘top’ shows all processes.
* If startup script is not in home folder then you have to give it the execute permission by ‘chmod +x /../../startup.sh.

### Create several node.js applications in one cloud server and control them from a startup script

* I created a project sundry-sites in GitHub. I put two node servers 1) capital-seo, 2 service-karao-seo in that. The two projects only have compiled code and not the source files. I zipped the relevant files in sundry-sites.zip
* Created a folder sundry-sites in \home and uploaded the sundry-sites.zip in folder sundry-sites and gave command ‘unzip sundry-folders’. All files with hierarchy are now there.
* We don’t need node\_modules since files are compiled files. We need to just execute ‘node local.js’ in respective directories. We will take care of that through the startup.sh script.
* I had already provided a string ‘/home/startup.sh’ in the cmd / entry point through the interface. That means it would execute the startup.sh when reboot is done. I modified the startup.sh as follows:

cd /home/cosmos && node server &

cd /home/sundry-sites/capital-seo && node local &

cd /home/sundry-sites/service-karao-seo && node local &

See the’&’ symbol at last which will not block the thread in Linux. I needed to do ‘cd’ to reach to the respective directories.

Cool!!!