5th Sept,2018

Docker: Concepts in Container Technologies

Containers : A Definition

A container is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it- code, runtime, system tools, system libraries, settings.

Benefits

- Resource efficient. Faster to boot.
- Portable. Consistent Environments
- Fast, consistent delivery of your applications
- Suited for DevOps, CI/CD and Microservices.
- Multiple applications with multiple dependencies
- Eliminates the “Works on my Machine” situation.

According to a recent study** by 451 Research, the adoption of application containers will grow by 40% annually through 2020.
Containers are facilitating rapid and agile development like never before.

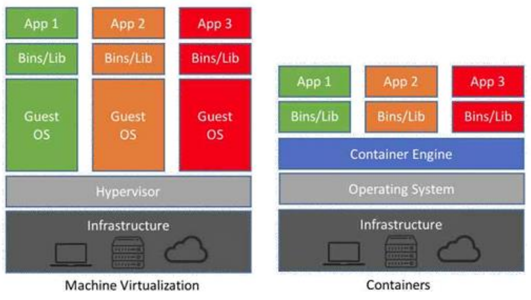
Virtual Machines (VMs) vs Containers

Virtual Machines :

- Virtualization software allows setting up one OS within another.
- The Guest OS' are isolated from the host and communicate with each other through a **Hypervisor** (A hypervisor or VMM(virtual machine monitor) is software, firmware or hardware that creates and runs VMs).
- It virtualizes the hardware layer. All OS resources available to apps.
- Established management tools & security tools and better-known security controls
- Added overhead in memory and storage footprint. Need to patch each VM OS.
- They are heavier to run and less portable.

Containers :

- **Container** is a isolated set of package, libraries and/or apps that are completely independent from its surroundings.
- Container technology virtualizes the Operating System.
- The containers on a machine share the OS kernel and often, binaries and libraries too, of the Host OS.
- Containers reduce OS management overhead. Only Host OS needs to be patched.
- They are light weight and more portable than VMs.



Popular VM Vendors
- VMware vSphere
- VirtualBox
- Xen
- Hyper-V
- KVM

Container Providers
- Docker
- Core OS rkt (*rocket*)
- Linux Containers (LXC)

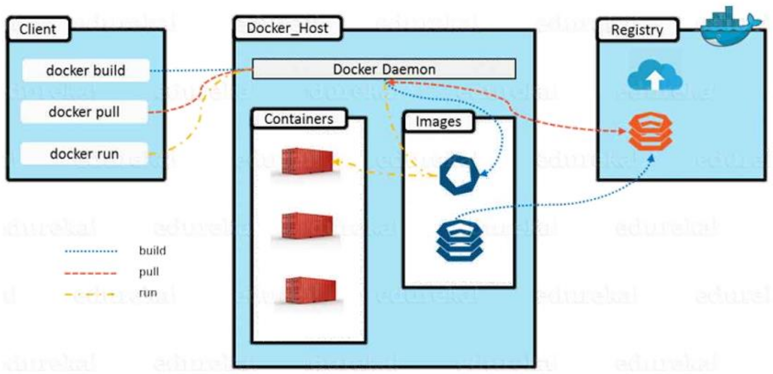
Virtual Box is the vendor and vagrant are the software to create different VM.

What is Docker?

- **Docker** is an application build and deployment tool.
- It is based on the idea of that you can package your code with dependencies into a deployable unit called a **container**.
- **Images** are an artifact, essentially a snapshot of the contents a container is meant to run
- For Windows, Docker Engine uses "Linux specific" kernel features, it needs to use a lightweight virtual machine to do its work.

Docker Architecture

- **Docker** is a client-server application where both the daemon and client can reside on the same system.
- Both of them talk to each other via sockets or through a RESTful (Represented State Transfer) API.
- Main components of Docker:
  - Daemon
  - Client
  - Docker.io Registry



Important Terms

Layer - a set of read-only files to provision the system

Image - a read-only layer that is the base of your container. It might have a parent image.

Container - a runnable instance of the image

Registry / Repository - Central place where images live

(Public – Docker Hub http://hub.docker.com or Private registry)

Docker Machine - a VM to run Docker containers (Linux does this natively)

Docker Compose - a utility to run multiple containers as a system

Install and Configure Docker

Source : https://docs.docker.com/install/linux/docker-ce/centos/#install-docker-ce

Obtain root privileges in the terminal and install nano editor

```
$ sudo su - root
```

All commands henceforth are run with elevated privileges.

```
$ yum install -y nano      (Optional step. You can use vim or vi editor too)
```

1. SET UP THE REPOSITORY

Install required packages.

```
$ yum install -y yum-utils device-mapper-persistent-data lvm2
```

Use the following command to set up the **stable** repository.

```
$ yum-config-manager --add-repo \ https://download.docker.com/linux/centos/docker-ce.repo
```

2. INSTALL DOCKER CE

```
$ yum install -y docker-ce
```

Configure Docker to start on boot

```
$ systemctl enable docker
```

Start Docker.

```
$ systemctl start docker
```

https://docs.docker.com/network/proxy/#configure-the-docker-client
https://docs.docker.com/config/daemon/systemd/#httphttps-proxy

Configure Docker to use a proxy server

```
$ mkdir -p /etc/systemd/system/docker.service.d
$ nano /etc/systemd/system/docker.service.d/http-proxy.conf
[Service]
Environment="HTTP_PROXY= http://www-proxy.us.oracle.com:80/"
Environment="HTTPS_PROXY= http://www-proxy.us.oracle.com:80/"
```

Reload the Docker Daemon and Restart the docker engine

```
$ systemctl daemon-reload
$ systemctl restart docker
```

Verify if Docker can connect to Docker registry

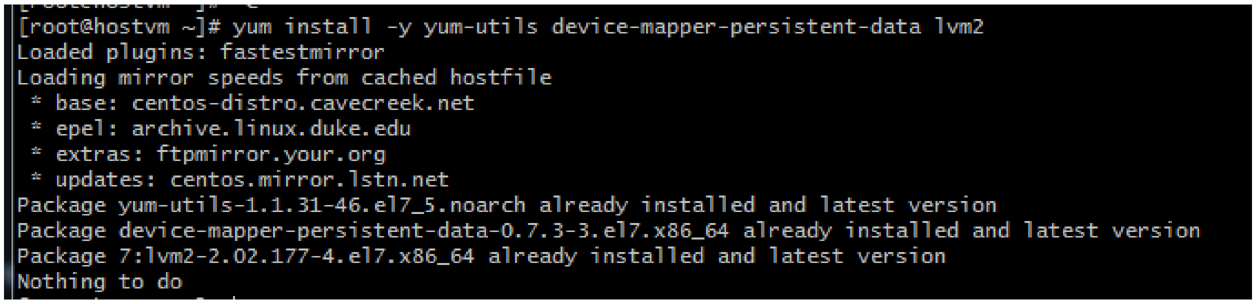
```
$ docker run hello-world
```

Source: https://docs.docker.com/install/linux/docker-ce/centos/#install-docker-ce

To install nano editor: pwd: /home/vagrant

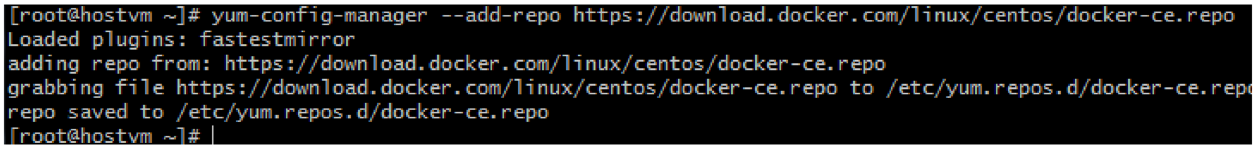
1. sudo su – root
2. `whoami` →`To check the current user`
3. yum install -y nano → To download code editor
4. yum install -y yum-utils device-mapper-persistent-data lvm2

   To enable the system for docker usage these are prerequisite package



```
[root@hostvm ~]# yum install -y yum-utils device-mapper-persistent-data lvm2
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos-distro.cavecreek.net
 * epel: archive.linux.duke.edu
 * extras: ftpmirror.your.org
 * updates: centos.mirror.lstn.net
Package yum-utils-1.1.31-46.el7_5.noarch already installed and latest version
Package device-mapper-persistent-data-0.7.3-3.el7.x86_64 already installed and latest version
Package 7:lvm2-2.02.177-4.el7.x86_64 already installed and latest version
Nothing to do
```

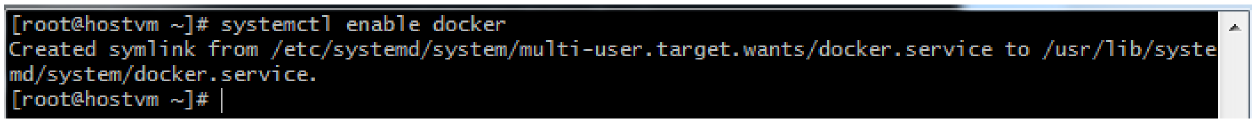
5. **yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo**



```
[root@hostvm ~]# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.rep
repo saved to /etc/yum.repos.d/docker-ce.repo
[root@hostvm ~]# 
```

6. **To open using nano**
   ```
   nano /etc/yum.repos.d/docker-ce.repo
   CTRL+X for exit
   ```
7. **`Installing Docker:`**
   ```
   yum install -y docker-ce
   ```
8. **Configure Docker to start on boot**

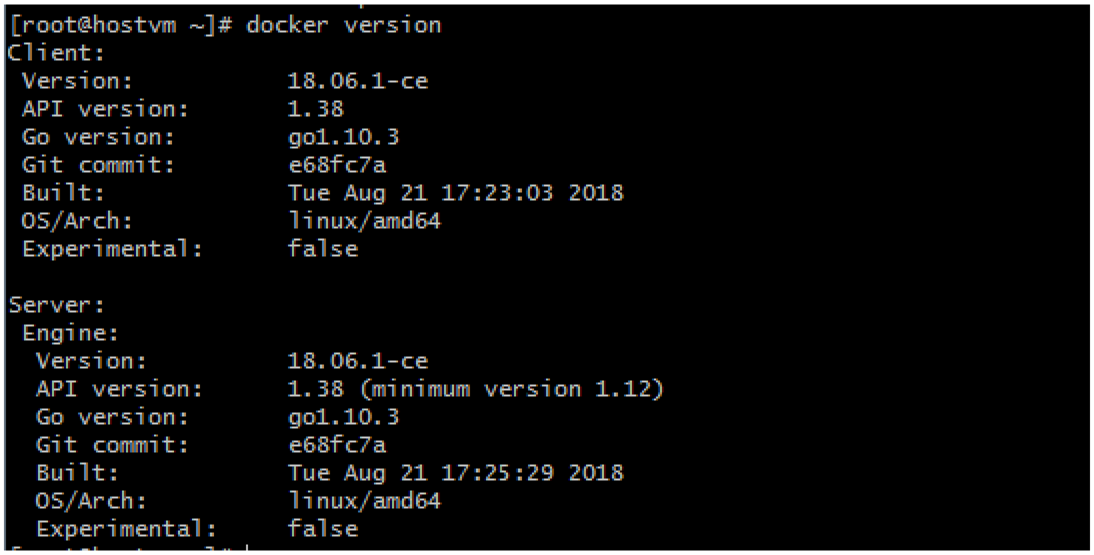
   systemctl enable docker



```
[root@hostvm ~]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/syste
md/system/docker.service.
[root@hostvm ~]# 
```

9. To check the status of docker:
   ```
   systemctl status docker
   ```
10. **To stop docker**

    Stop docker
11. **To start docker:**
    ```
    systemctl start docker
    ```
12. `To check docker version:`

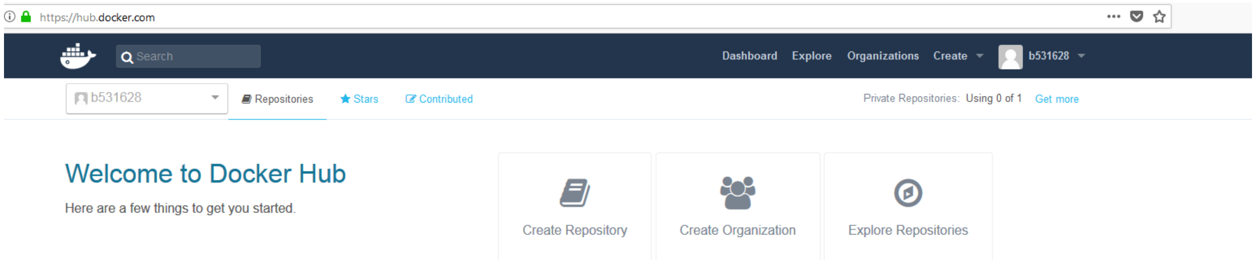


```
[root@hostvm ~]# docker version
Client:
 Version:           18.06.1-ce
 API version:       1.38
 Go version:        go1.10.3
 Git commit:        e68fc7a
 Built:             Tue Aug 21 17:23:03 2018
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.06.1-ce
  API version:      1.38 (minimum version 1.12)
  Go version:       go1.10.3
  Git commit:       e68fc7a
  Built:            Tue Aug 21 17:25:29 2018
  OS/Arch:          linux/amd64
  Experimental:     false
```

13. Create an account in docker hub:

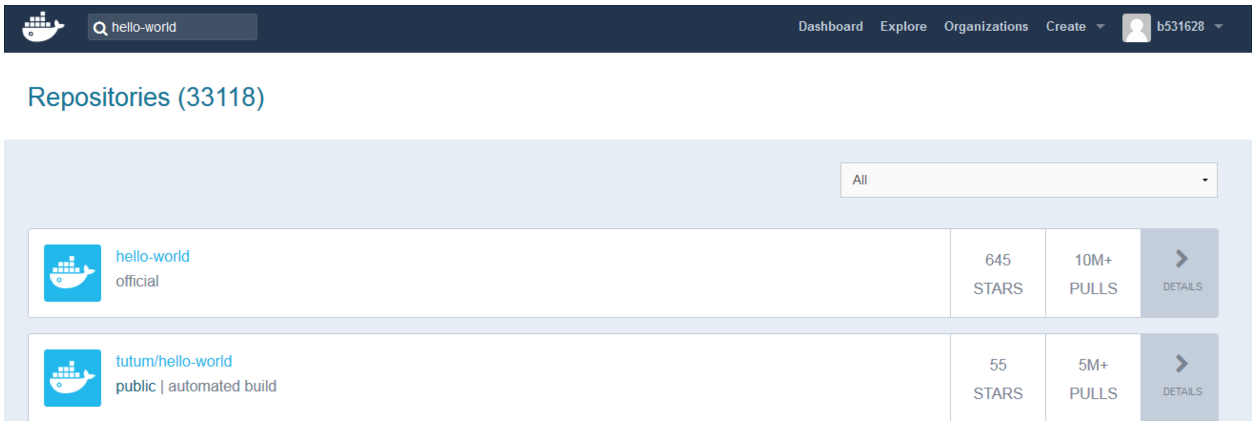
https://hub.docker.com/

User Id: b531628, **Password:** b531628
Email: Bishwajit.patel@oracle.com
2nd account: bispatel/bispatel
Email: bishwajit.patel@gmail.com



Welcome to Docker Hub

Here are a few things to get you started.



Repositories (33118)

hello-world official	645 STARS	10M+ PULLS
tutum/hello-world public | automated build	55 STARS	5M+ PULLS

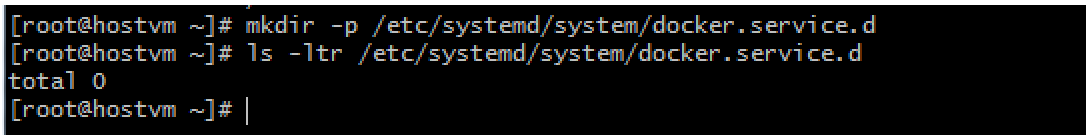
In the terminal:

docker search hello-world

14. Docker creates its owner internal network. So isolated layer can interact with internet using proxy

Setting up proxy for docker:

1. mkdir -p /etc/systemd/system/docker.service.d



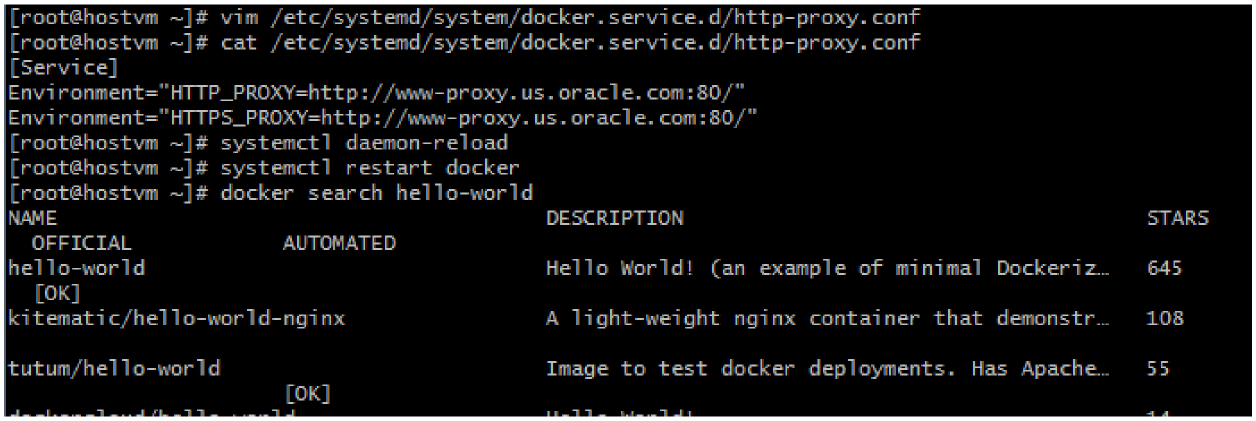
```
[root@hostvm ~]# mkdir -p /etc/systemd/system/docker.service.d
[root@hostvm ~]# ls -ltr /etc/systemd/system/docker.service.d
total 0
[root@hostvm ~]# |
```

2. ls -ltr /etc/systemd/system/docker.service.d
3. To create config file:

vim /etc/systemd/system/docker.service.d/http-proxy.conf

Content of the file:

[Service]
Environment="HTTP_PROXY=http://www-proxy.us.oracle.com:80/"
Environment="HTTPS_PROXY=http://www-proxy.us.oracle.com:80/"



```
[root@hostvm ~]# vim /etc/systemd/system/docker.service.d/http-proxy.conf
[root@hostvm ~]# cat /etc/systemd/system/docker.service.d/http-proxy.conf
[Service]
Environment="HTTP_PROXY=http://www-proxy.us.oracle.com:80/"
Environment="HTTPS_PROXY=http://www-proxy.us.oracle.com:80/"
[root@hostvm ~]# systemctl daemon-reload
[root@hostvm ~]# systemctl restart docker
[root@hostvm ~]# docker search hello-world
NAME                                       DESCRIPTION                                     STARS
  OFFICIAL              AUTOMATED
hello-world                                Hello World! (an example of minimal Dockeriz…   645
  [OK]
kitematic/hello-world-nginx                A light-weight nginx container that demonstr…   108

tutum/hello-world                          Image to test docker deployments. Has Apache…   55
                        [OK]
```

15. Reload docker so that proxy will be picked
systemctl daemon-reload

16. Restart Docker
systemctl restart docker

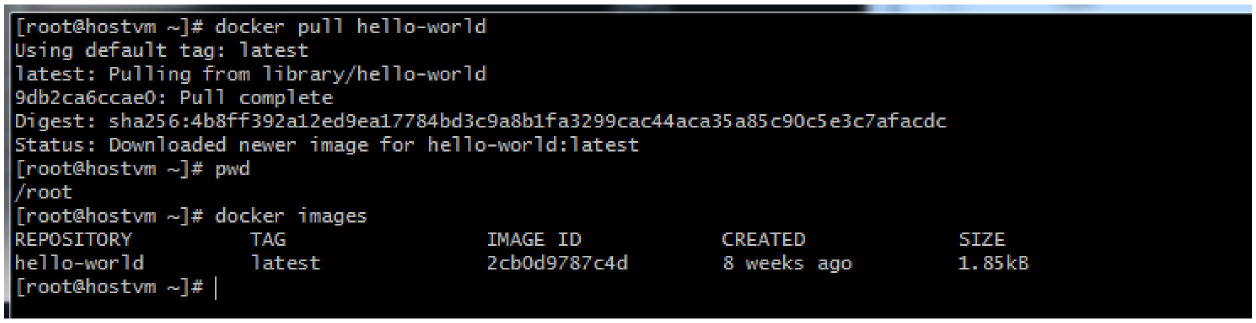
17. Seraching for a docker repository:
docker search hello-world

Official Image will have : image_name
Private Image: username/image_Name

18. Docker documentation:
https://docs.docker.com

19. To pull docker image from repository:
docker pull hello-world

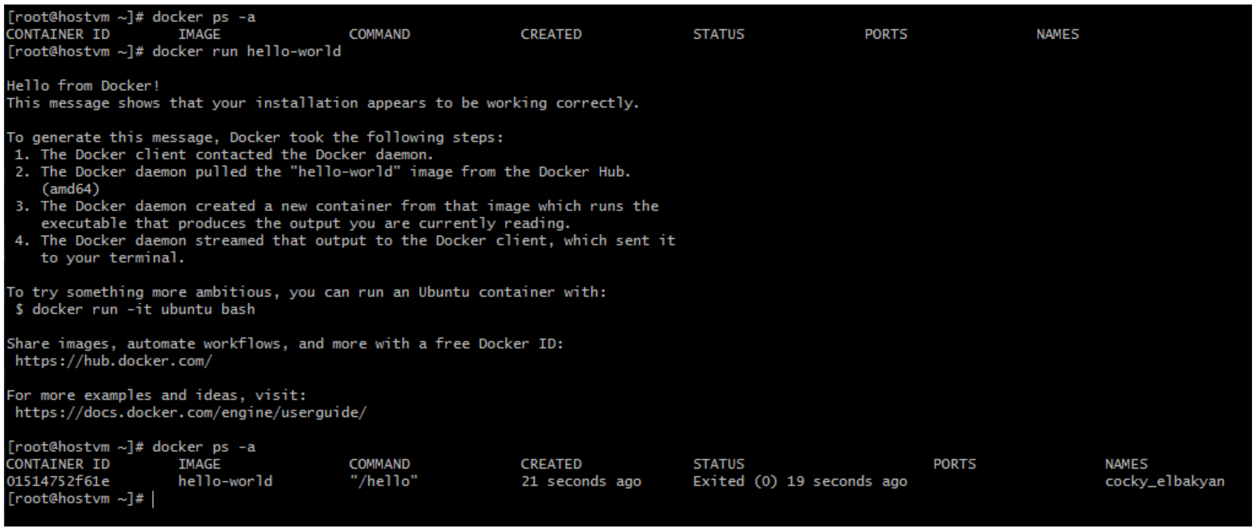
20. To check all the images in docker:
docker images



```
[root@hostvm ~]# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
9db2ca6ccae0: Pull complete
Digest: sha256:4b8ff392a12ed9ea17784bd3c9a8b1fa3299cac44aca35a85c90c5e3c7afacdc
Status: Downloaded newer image for hello-world:latest
[root@hostvm ~]# pwd
/root
[root@hostvm ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              2cb0d9787c4d        8 weeks ago         1.85kB
[root@hostvm ~]# |
```

21. To check if any container is running or not
`docker ps -a`

22. To run image as acontainer
`docker run hello-world`

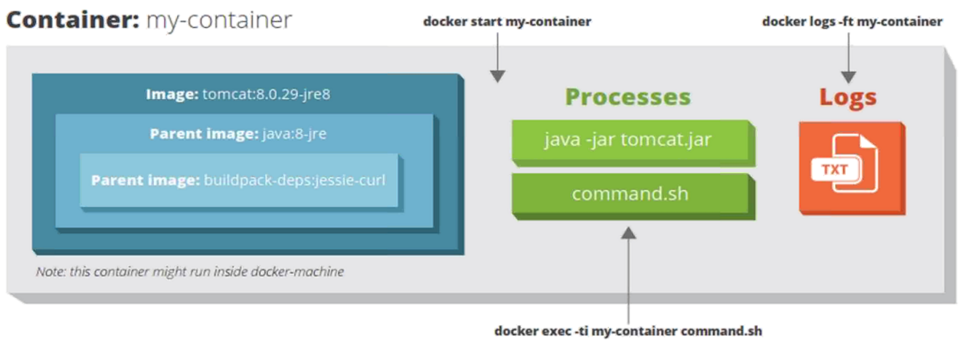


```
[root@hostvm ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@hostvm ~]# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

[root@hostvm ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
01514752f61e        hello-world         "/hello"            21 seconds ago      Exited (0) 19 seconds ago                       cocky_elbakyan
[root@hostvm ~]# |
```

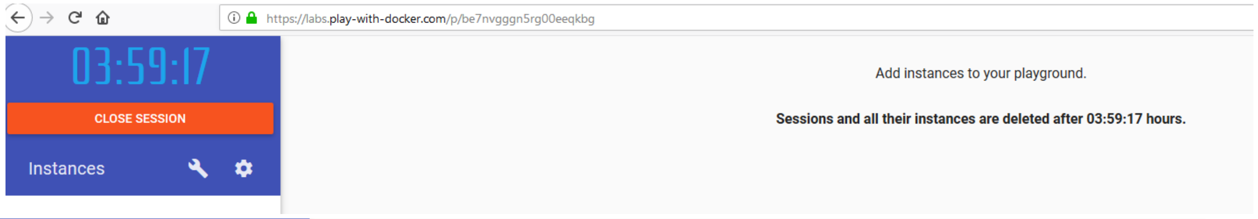
Docker Container

Containers allow the packaging of your application in a 'container image'.
- a base operating system, libraries, files and folders,
- environment variables, volume mount-points
- your application binaries.

An 'Image' is a template for the execution of a container —
- Multiple containers can be running from the same image, all sharing the same behavior
- Promotes the scaling and distribution of the application
- These images can be stored in a remote registry to ease the distribution.


Container: my-container
docker start my-container
docker logs -ft my-container
Image: tomcat:8.0.29-jre8
Parent image: java:8-jre
Parent image: buildpack-deps:jessie-curl
Processes
java -jar tomcat.jar
command.sh
Logs
TXT
Note: this container might run inside docker-machine
docker exec -ti my-container command.sh

23. Online Utility to play with docker:
https://labs.play-with-docker.com/



https://labs.play-with-docker.com/p/be7nvgggn5rg00eeqkbg

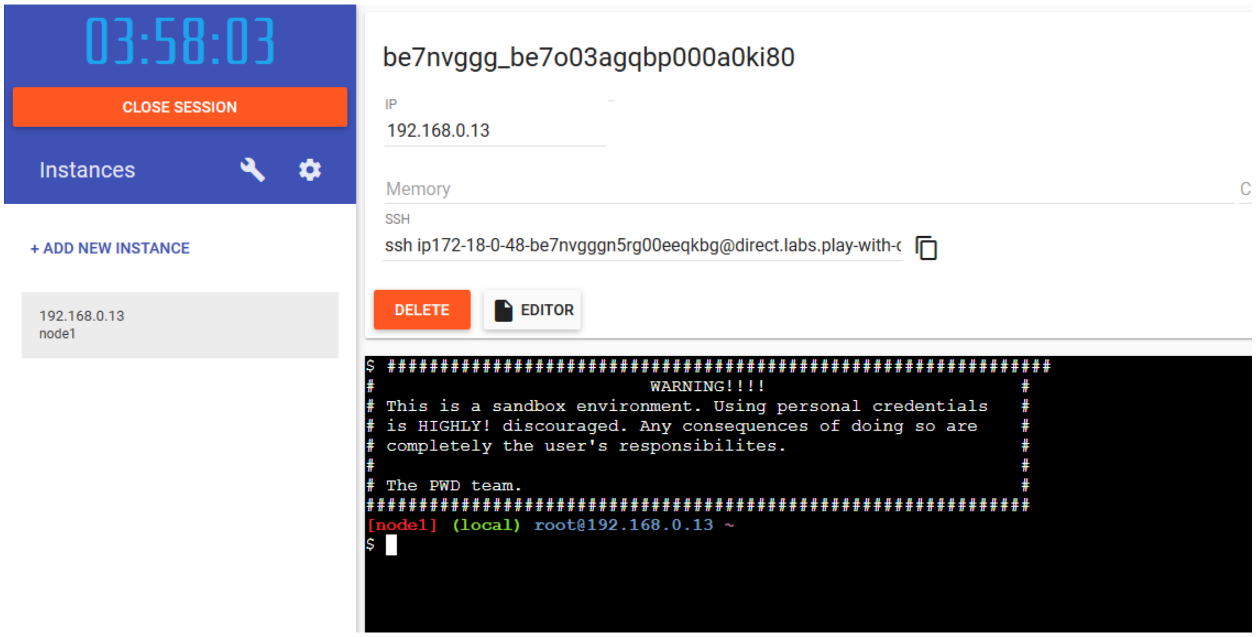
03:59:17

CLOSE SESSION

Instances

Add instances to your playground.

Sessions and all their instances are deleted after 03:59:17 hours.



03:58:03

CLOSE SESSION

Instances

+ ADD NEW INSTANCE

192.168.0.13
node1

be7nvggg_be7o03agqbp000a0ki80

IP
192.168.0.13

Memory

SSH
ssh ip172-18-0-48-be7nvggn5rg00eeqkbg@direct.labs.play-with-

DELETE

EDITOR

```
$ ##############################################################
#                          WARNING!!!!                          #
# This is a sandbox environment. Using personal credentials    #
# is HIGHLY! discouraged. Any consequences of doing so are     #
# completely the user's responsibilites.                       #
#                                                               #
# The PWD team.                                                 #
#################################################################
[node1] (local) root@192.168.0.13 ~
$ 
```