Day 1

Lab Setup
Introductions to Docker
First Step with Docker

Modules

- Lab Access / Azure Setup / Lab Scripts
- Chapter 1: Introducing Docker, Docker Compose, First steps with Docker

Lab Setup

Lab Setup

- Labs will be hosted in Microsoft Azure
- All Virtual Machines will be created in Microsoft Azure
- You Need a windows or Mac or Linux Desktop or Laptop
- Please TURN OFF any VPN Service before accessing MS Azure

Lab Requirements – 1

- A Microsoft Based email address (live , Hotmail , outlook)
- DO NOT USE any EXISTING ACCOUNT that been associated with Azure Services or M365 Services
- STRICTLY Microsoft based Account –
  - NO COMPANY ACCOUNT, NO GMAIL , NO YAHOO… etc
- If you don't have, Please create one now (its FREE)

Lab Requirements – 2

- A GitHub Account
- Open a GitHub account using the previously created Microsoft Account
- If you don't have, Please create one now (its FREE)
- You may skip this part , but don't ask Steven to pass you any code manually , He won't , his code for this training available at github

Lab Requirements – 3

- A Docker HUB account
- Open a Docker Hub account using the previously created Microsoft Account
- If you don’t have, Please create one now (its FREE)
- This account is for uploading your container images

Lab Requirements – 3

- Activate your Microsoft Azure Pass
- Azure Pass will come with $100 use pass with 30days validity
- Azure Pass have Limitation – 10 vcpu / Account Limit
- DO NOT create anything on Azure
  - All Creation Script will be given as we progress
- Once Azure is Activated, Login and Activate Cloud Shell

Lab Requirements – 4

- Download and Install List -
- Putty for SSH Client (Windows)
- PuttyGen for converting OLD PEM format to putty format (Windows)
- Visual Studio Code (Windows and Linux)
- GitHub Desktop (Windows)

Introducing Docker & Docker Compose

Chapter 1

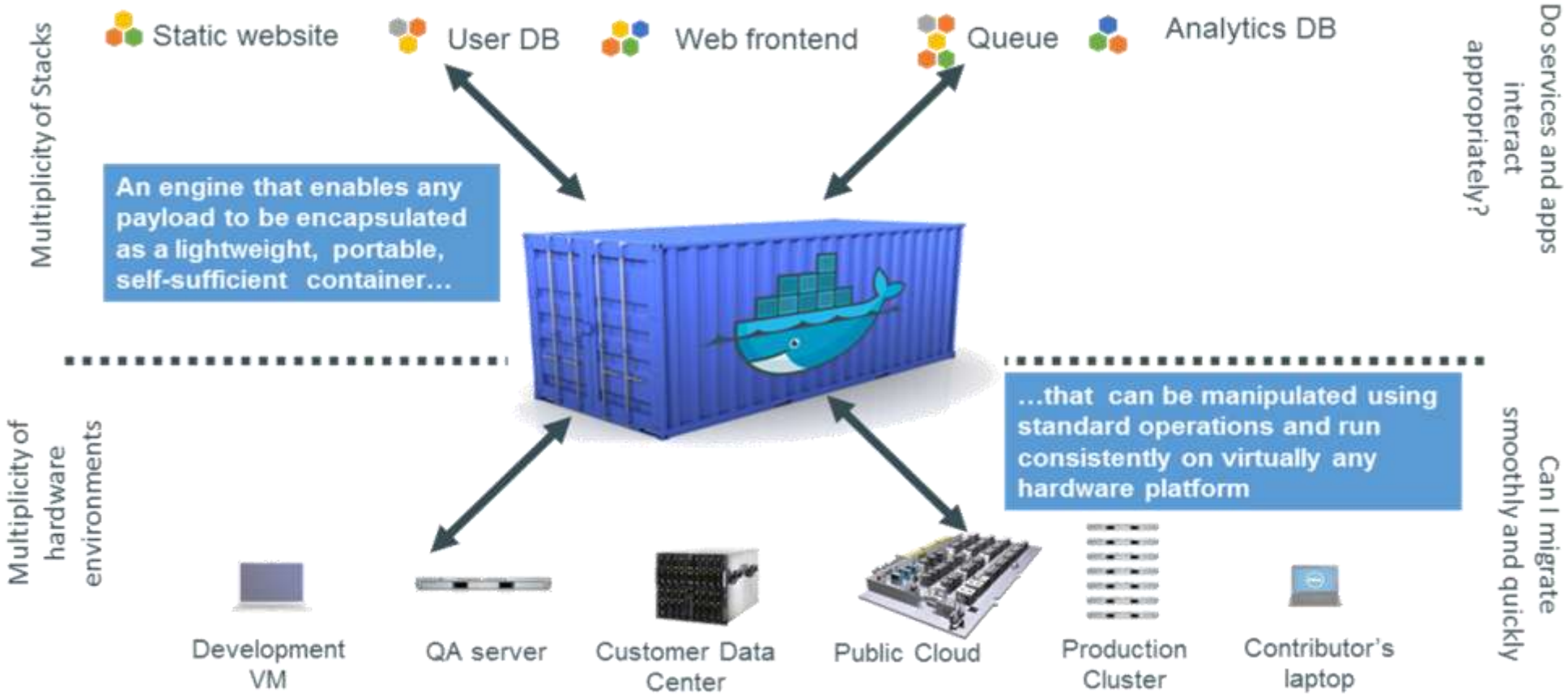
What is Docker?

- Open platform for developers and sysadmins to build, ship and run distributed applications
- Can run on popular 64-bit Linux distributions with kernel 3.8 or later
- Latest development allows docker to be adapted by Windows Platform*
- Supported by several cloud platforms including Amazon EC2, Google Compute Engine, Rackspace, Azure.
- *Windows 10 supports Docker Desktop / Windows Server Supports Docker Enterprise Natively

Features of Docker

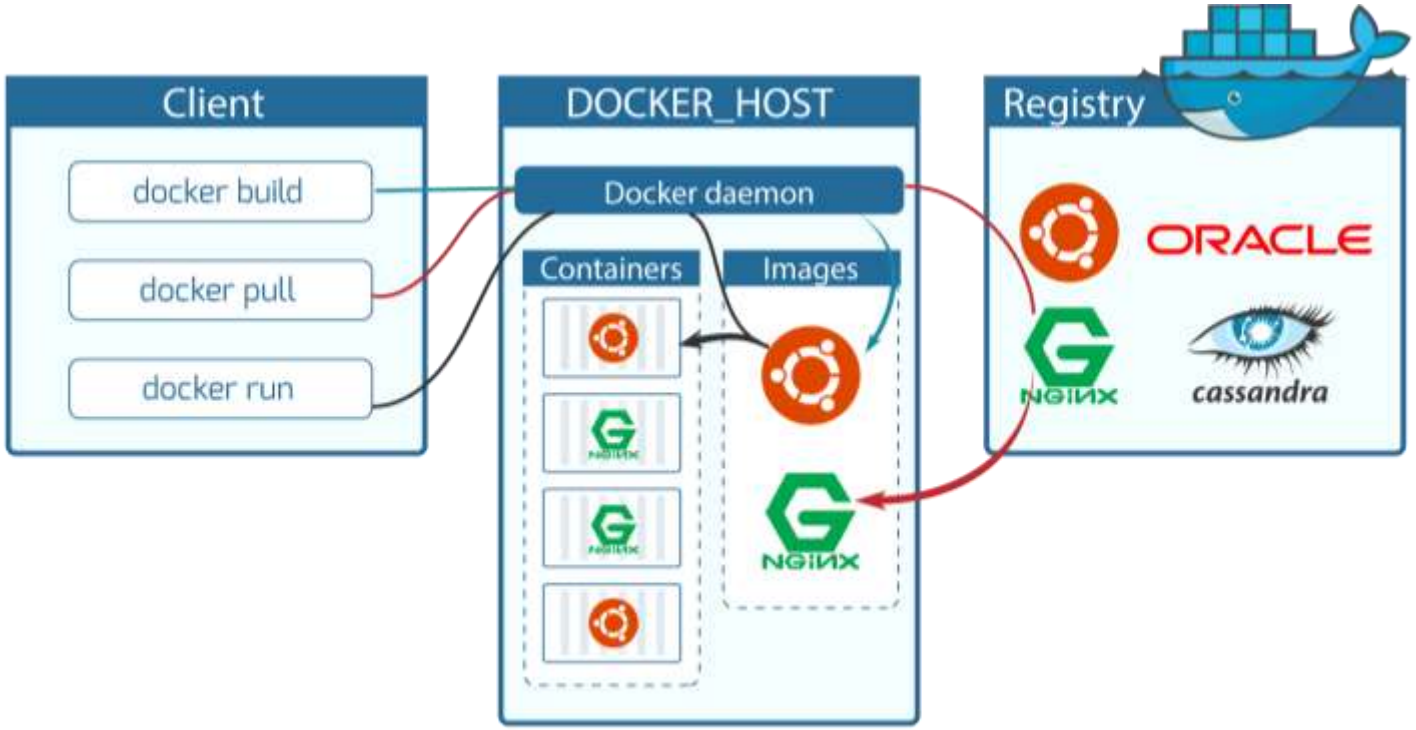
- Light-Weight
  - Minimal overhead
  - Uses layered filesystem to save space
  - Uses a copy-on-write filesystem to track changes
- Portable
  - Can run on any Linux system
  - Support for other operating systems (Solaris, OSX, Windows)
- Self-sufficient
  - A Docker container contains everything it needs to run
  - Minimal Base OS
  - Libraries , frameworks & Application code in one container
  - A docker container should be able to run anywhere that Docker can run

Docker is a Container System for Code



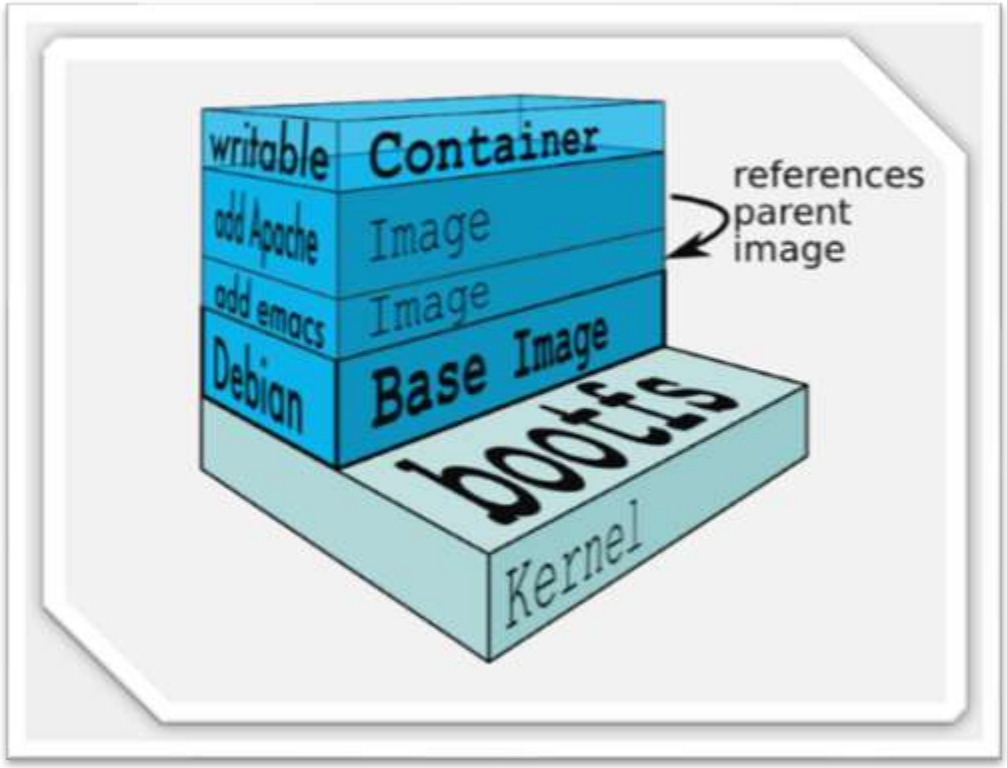
Docker Architecture

- Docker Engine
  - Client
  - Docker Daemon
- Docker Hub
  - Docker Registry
- Docker images



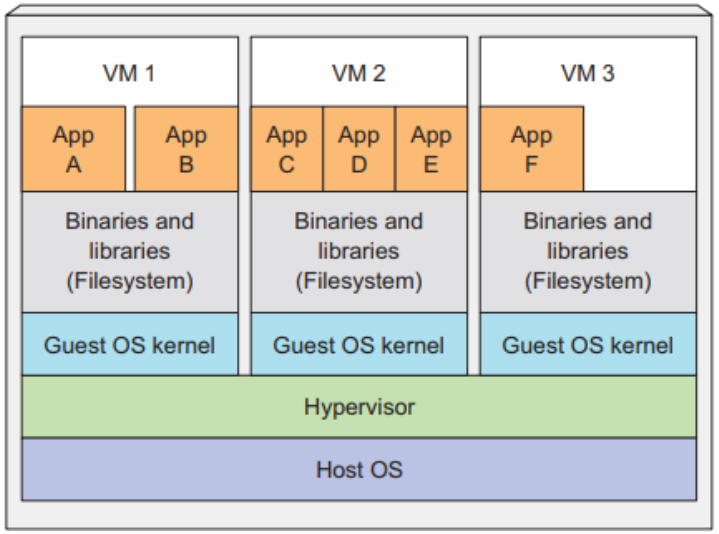
Docker images

- NOT A VHD
- NOT A FILESYSTEM
- uses a Union File System
- do not have state
- Has a hierarchy
- Fits into the Registry

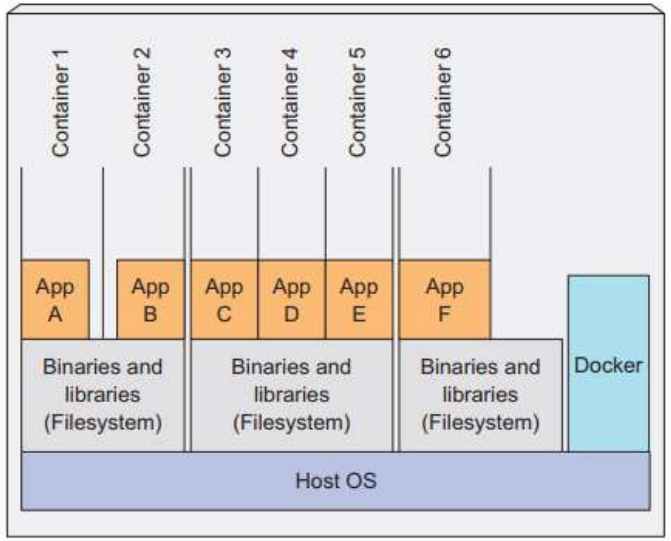


Virtual Machine vs Container

Host running multiple VMs



Host running multiple Docker containers



Lab Setup

- Go to Azure Cloud Shell
- Git Clone https://github.com/stv707/k8_training
- Change directory to k8_training
- Follow gitHub Instruction to run the main.sh script to setup Docker VM
- Write down the Public IP address of vm001
- Download the Private Key to your System
- Convert the PEM to Putty Private Key
- Use Putty to connect to VM001 on Azure
- Remember where you stored your Putty Private Key

Docker Basic

Practice A – Lab Time

Docker basic

```
[root@servera ~]# docker version
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea
 Built:             Wed Nov 13 07:25:41 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.5
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.12
  Git commit:       633a0ea
  Built:            Wed Nov 13 07:24:18 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.10
  GitCommit:        b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:          1.0.0-rc8+dev
  GitCommit:        3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```

```
[root@servera ~]# docker info
Client:
 Debug Mode: false

Server:
 Containers: 2
  Running: 0
  Paused: 0
  Stopped: 2
 Images: 6
 Server Version: 19.03.5
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: b34a5c8af56e510852c35414db4c1f4fa6172339
 runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
 init version: fec3683
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 3.10.0-1062.9.1.el7.x86_64
 Operating System: CentOS Linux 7 (Core)
 OSType: linux
 Architecture: x86_64
 CPUs: 2
 Total Memory: 1.795GiB
```

Docker Basic

- Docker command line structure
- Old (still works) docker <command> (options)
- New : docker <command> <sub-command> (options)

```
[root@servera ~]# docker run hello-world

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

```
[root@servera ~]# docker container run hello-world

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

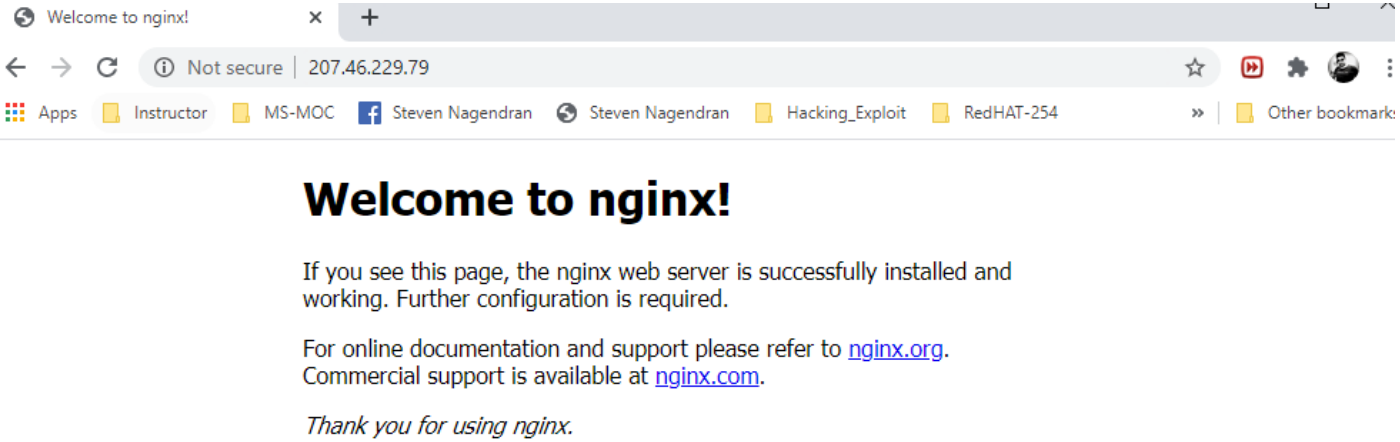
Docker Basic

- Internet is a MUST – high speed
- Docker hub access
- Firstly, verify we have no existing image and any container is running

```
[root@servera ~]# docker image ls -a
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@servera ~]#
[root@servera ~]#
[root@servera ~]# docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@servera ~]#
```

Docker Basic

```
[root@servera ~]# docker container run -p 80:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
8ec398bc0356: Pull complete
dfb2a46f8c2c: Pull complete
b65031b6a2a5: Pull complete
Digest: sha256:8aa7f6a9585d908a63e5e418dc5d14ae7467d2e36e1ab4f0d8f9d059a3d071ce
Status: Downloaded newer image for nginx:latest
```



Docker Basic – detached mode

```
[root@servera ~]# docker container run -p 80:80 -d nginx
7bbfd133c3a247f30b09413c60535ba1e5ca9b65905908b0398bf0c551e2bc79
[root@servera ~]#
```

```
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
7bbfd133c3a2        nginx               "nginx -g 'daemon of…"   2 minutes ago       Up 2 minutes        0.0.0.0:80->80/tcp      elastic_nobel
```

```
[root@servera ~]# ss -tanp | grep 80
LISTEN     0      128       [::]:80                    [::]:*                    users:(("docker-proxy",pid=19123,fd=4))
[root@servera ~]#
```

```
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
7bbfd133c3a2        nginx               "nginx -g 'daemon of…"   6 minutes ago       Up 6 minutes        0.0.0.0:80->80/tcp      elastic_nobel
[root@servera ~]#
[root@servera ~]#
[root@servera ~]# docker container stop 7bb
7bb
[root@servera ~]#
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@servera ~]#
```

Docker Basic -naming

```
[root@servera ~]# docker container run -p 80:80 -d --name  mywebhost nginx
69971d7b9cc935ad22c0792a4b99e4220a3196b421f09b423a715489ce63f2a1
[root@servera ~]#
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
69971d7b9cc9        nginx               "nginx -g 'daemon of…"   6 seconds ago       Up 5 seconds        0.0.0.0:80->80/tcp      mywebhost
```

```
[root@servera ~]# docker container rename mywebhost panpanweb
[root@servera ~]#
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
69971d7b9cc9        nginx               "nginx -g 'daemon of…"   53 seconds ago      Up 53 seconds       0.0.0.0:80->80/tcp      panpanweb
```

Docker Basic - Logs

```
[root@servera ~]# docker container logs panpanweb
172.17.0.1 - - [11/Jan/2020:07:15:15 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/2010010
1 Firefox/68.0" "-"
172.17.0.1 - - [11/Jan/2020:07:15:16 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/2010010
1 Firefox/68.0" "-"
```

Docker Basic - top

```
[root@servera ~]# docker container top panpanweb
UID                 PID                 PPID                C                   STIME               TTY                 TIME
            CMD
root                19890               19873               0                   15:06               ?                   00:00:
00          nginx: master process nginx -g daemon off;
101                 19924               19890               0                   15:06               ?                   00:00:
00          nginx: worker process
```

Docker Basic - Cleanup

```
[root@servera ~]# docker container ls -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS                   NAMES
69971d7b9cc9        nginx               "nginx -g 'daemon of…"   21 minutes ago      Up 21 minutes               0.0.0.0:80->80/tcp      panpanweb
7bbfd133c3a2        nginx               "nginx -g 'daemon of…"   31 minutes ago      Exited (0) 25 minutes ago                           elastic_nobel
fb33db263f88        nginx               "nginx -g 'daemon of…"   35 minutes ago      Exited (0) 31 minutes ago                           goofy_moser
[root@servera ~]#
[root@servera ~]# docker container stop 69971d7b9cc9 7bbfd133c3a2 fb33db263f88
69971d7b9cc9
7bbfd133c3a2
fb33db263f88
[root@servera ~]# docker container ls -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS                   NAMES
69971d7b9cc9        nginx               "nginx -g 'daemon of…"   21 minutes ago      Exited (0) 2 seconds ago                            panpanweb
7bbfd133c3a2        nginx               "nginx -g 'daemon of…"   32 minutes ago      Exited (0) 25 minutes ago                           elastic_nobel
fb33db263f88        nginx               "nginx -g 'daemon of…"   35 minutes ago      Exited (0) 32 minutes ago                           goofy_moser
[root@servera ~]#
[root@servera ~]# docker container rm 69971d7b9cc9 7bbfd133c3a2 fb33db263f88
69971d7b9cc9
7bbfd133c3a2
fb33db263f88
[root@servera ~]#
[root@servera ~]# docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@servera ~]#
```

Docker Basic

Practice A – End of LAB

What happens when we run container?

```
[root@servera ~]# docker container run -p 80:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
8ec398bc0356: Pull complete
dfb2a46f8c2c: Pull complete
b65031b6a2a5: Pull complete
Digest: sha256:8aa7f6a9585d908a63e5e418dc5d14ae7467d2e36e1ab4f0d8f9d059a3d071ce
Status: Downloaded newer image for nginx:latest
```

- 1. Looks for image locally in image cache , if nothing found, then
- 2. docker looks in remote image repo (defaults to Docker Hub)
- 3. Downloads the latest version (nginx:latest by default)
- 4. Creates new container based on that image and prepare to start
- 5) Gives a virtual ip on a private network inside docker engine
- 6) opens up port 80 on host and forwards to port 80 in container
- 7) starts the container fully

Container vs VM : Its just a process

- Containers aren't Mini-VM's
- They are just processes
- Limited to what resources they can access [file paths , network , devices, other running processes]
- Exits when a process stops

```
[root@servera ~]# docker run --name mongo -d mongo
Unable to find image 'mongo:latest' locally
latest: Pulling from library/mongo
2746a4a261c9: Pull complete
4c1d20cdee96: Pull complete
0d3160e1d0de: Pull complete
c8e37668deea: Pull complete
fc3987a82b4c: Pull complete
c75f139e0836: Pull complete
4acc9c8680b4: Pull complete
fb02df30d947: Pull complete
ae725ef3d2ce: Pull complete
e30f54ed6b43: Pull complete
bca9e535ddb8: Pull complete
9c3edad81b2a: Pull complete
6dbcf78fe5ae: Pull complete
Digest: sha256:7a1406bfc05547b33a3b7b112eda6346f42ea93ee06b74d30c4c47dfeca0d5f2
Status: Downloaded newer image for mongo:latest
18438e604a2cf459debbb067f979c64e8242daaa9316b570e267f4394b9e0453
```

Container vs VM : Its just a process

```
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS                   NAMES
18438e604a2c        mongo               "docker-entrypoint.s…"   About a minute ago   Up About a minute   27017/tcp               mongo
4129beedb16e        httpd:2.2.31        "httpd-foreground"       9 minutes ago        Up 8 minutes        0.0.0.0:8080->80/tcp    myapache
[root@servera ~]#
[root@servera ~]# docker container top mongo
UID                 PID                 PPID                C                   STIME               TTY                 TIME
polkitd             21907               21890               0                   15:58               ?                   00:00:00
_all
[root@servera ~]#
[root@servera ~]# ps aux | grep 21907 | grep -v grep
polkitd  21907  0.8  4.4 1576052 84264 ?       Ssl  15:58   0:00 mongod --bind_ip_all
```

```
[root@servera ~]# docker container stop mongo
mongo
[root@servera ~]# docker container top mongo
Error response from daemon: Container 18438e604a2cf459debbb067f979c64e8242daaa9316b570e267f4394b9e0453 is not running
[root@servera ~]#
[root@servera ~]# ps aux | grep mongo  | grep -v grep
```

Common container management

- docker container top – process list in one container
- docker container inspect – details of container configuration , metadata and startup settings
- docker container stats – performance stats for all container [cpu, memory, network and disk]
- docker image - list / manage downloaded container images

Common container management

```
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                   NAMES
6c6fb3f28e4c        nginx               "nginx -g 'daemon of…"   2 hours ago         Up 2 hours          0.0.0.0:80->80/tcp                      proxy
ff2388a9aee8        httpd               "httpd-foreground"       2 hours ago         Up 2 hours          0.0.0.0:7171->80/tcp                    webserver
3d8631b74516        mysql               "docker-entrypoint.s…"   2 hours ago         Up 2 hours          0.0.0.0:3306->3306/tcp, 33060/tcp       db
[root@servera ~]#
[root@servera ~]#
[root@servera ~]# docker container top ff2388a9aee8
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                23544               23527               0                   16:25               ?                   00:00:00            httpd -DFOREGROU
ND
bin                 23587               23544               0                   16:25               ?                   00:00:00            httpd -DFOREGROU
ND
bin                 23588               23544               0                   16:25               ?                   00:00:00            httpd -DFOREGROU
ND
bin                 23589               23544               0                   16:25               ?                   00:00:00            httpd -DFOREGROU
ND
```

Common container management

```
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
6c6fb3f28e4c        nginx               "nginx -g 'daemon of…"   2 hours ago         Up 2 hours          0.0.0.0:80->80/tcp
ff2388a9aee8        httpd               "httpd-foreground"       2 hours ago         Up 2 hours          0.0.0.0:7171->80/tcp
3d8631b74516        mysql               "docker-entrypoint.s…"   3 hours ago         Up 3 hours          0.0.0.0:3306->3306/tcp, 33060/
[root@servera ~]#
[root@servera ~]#
[root@servera ~]#
[root@servera ~]# docker container inspect proxy
[
    {
        "Id": "6c6fb3f28e4c0f46ed17ce0bf38b06b1e5044569628cc526ebea33fd3e5ca6bf",
        "Created": "2020-01-11T08:25:30.059842602Z",
        "Path": "nginx",
        "Args": [
            "-g",
            "daemon off;"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 23723,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2020-01-11T08:25:30.508997798Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:c7460dfcab502275e9c842588df406444069c00a48d9a995619c243079a4c2f7",
        "ResolvConfPath": "/var/lib/docker/containers/6c6fb3f28e4c0f46ed17ce0bf38b06b1e5044569628cc526ebea33fd3e5ca6bf/resolv.conf",
        "HostnamePath": "/var/lib/docker/containers/6c6fb3f28e4c0f46ed17ce0bf38b06b1e5044569628cc526ebea33fd3e5ca6bf/hostname",
        "HostsPath": "/var/lib/docker/containers/6c6fb3f28e4c0f46ed17ce0bf38b06b1e5044569628cc526ebea33fd3e5ca6bf/hosts",
        "LogPath": "/var/lib/docker/containers/6c6fb3f28e4c0f46ed17ce0bf38b06b1e5044569628cc526ebea33fd3e5ca6bf/6c6fb3f28e4c0f46ed17ce0
```

Common container management

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
6c6fb3f28e4c	proxy	0.00%	1.379MiB / 1.795GiB	0.08%	656B / 0B	13.8MB / 0B	2
ff2388a9aee8	webserver	0.00%	2.586MiB / 1.795GiB	0.14%	656B / 0B	8.11MB / 0B	82
3d8631b74516	db	0.55%	371.5MiB / 1.795GiB	20.21%	656B / 0B	341MB / 675MB	38

Common container management

```
[root@servera ~]# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              c7460dfcab50        36 hours ago        126MB
mysql               latest              ed1ffcb5eff3        13 days ago         456MB
httpd               latest              c2aa7e16edd8        13 days ago         165MB
mongo               latest              a0e2e64ac939        3 weeks ago         364MB
httpd               2.2.31              c8a7fb36e3ab        3 years ago         170MB
```

Getting a Shell inside a Container

- docker container run --it – Starts a new container interactively
- docker container exec --it – run additional command in existing container
- No SSH is needed to get into container
- Docker Cli is a great substitute for adding SSH to container

Getting a Shell inside a Container

- docker container run --it will run a container interactively
- You can further pass argument to start interactive shell

```
[root@servera ~]# docker container run -it --name nginx2 nginx bash
root@6614bba70eba:/#
root@6614bba70eba:/#
```

Getting a Shell inside a Container

- docker container exec --it : Run an interactive shell to change/update runtime container

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                   NAMES
6c6fb3f28e4c        nginx               "nginx -g 'daemon of…"   4 hours ago         Up 4 hours          0.0.0.0:80->80/tcp                      proxy
ff2388a9aee8        httpd               "httpd-foreground"       4 hours ago         Up 4 hours          0.0.0.0:7171->80/tcp                    webserver
3d8631b74516        mysql               "docker-entrypoint.s…"   4 hours ago         Up 4 hours          0.0.0.0:3306->3306/tcp, 33060/tcp       db
```

```
[root@servera ~]# docker container exec -it proxy bash
root@6c6fb3f28e4c:/#
root@6c6fb3f28e4c:/#
root@6c6fb3f28e4c:/#
root@6c6fb3f28e4c:/# apt-get update
```

```
root@6c6fb3f28e4c:/# apt-get install vim
Reading package lists... Done
Building dependency tree
```

```
root@6c6fb3f28e4c:/# vim /usr/share/nginx/html/index.html
```

Getting a Shell inside a Container

- Copying files from host to docker container

```
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                   NAMES
6c6fb3f28e4c        nginx               "nginx -g 'daemon of…"   4 hours ago         Up 4 hours          0.0.0.0:80->80/tcp                      proxy
ff2388a9aee8        httpd               "httpd-foreground"       4 hours ago         Up 4 hours          0.0.0.0:7171->80/tcp                    webserver
3d8631b74516        mysql               "docker-entrypoint.s…"   4 hours ago         Up 4 hours          0.0.0.0:3306->3306/tcp, 33060/tcp       db
[root@servera ~]#
[root@servera ~]#
[root@servera ~]# docker cp  data.txt proxy:/tmp/
```

```
root@6c6fb3f28e4c:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt
root@6c6fb3f28e4c:/# cd tmp/
root@6c6fb3f28e4c:/tmp# ls
root@6c6fb3f28e4c:/tmp#
root@6c6fb3f28e4c:/tmp# ls
data.txt
```

Getting a Shell inside a Container

- Getting shell inside running mysql container and perform mysql basic commands on database

```
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                   NAMES
6c6fb3f28e4c        nginx               "nginx -g 'daemon of…"   4 hours ago         Up 4 hours          0.0.0.0:80->80/tcp                      proxy
ff2388a9aee8        httpd               "httpd-foreground"       4 hours ago         Up 4 hours          0.0.0.0:7171->80/tcp                    webserver
3d8631b74516        mysql               "docker-entrypoint.s…"   4 hours ago         Up 4 hours          0.0.0.0:3306->3306/tcp, 33060/tcp       db
```

```
# docker container logs db

Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.
2020-01-11 08:17:09+00:00 [Note] [Entrypoint]: GENERATED ROOT PASSWORD: uXee8phiovoohoom4maethoi9thee0hu
```

Getting a Shell inside a Container

- Getting shell inside running mysql container and perform mysql basic commands on database

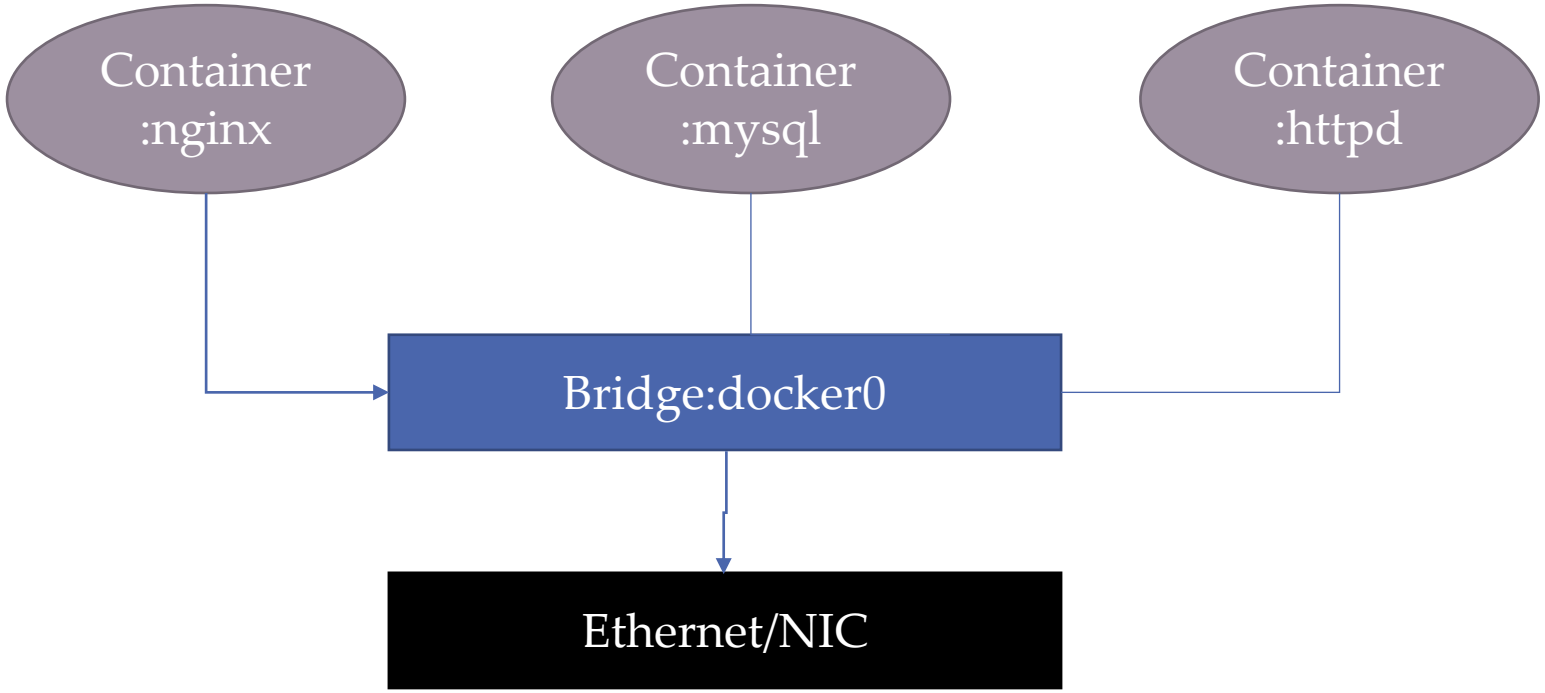
```
[root@servera ~]# docker container exec -it db bash
root@3d8631b74516:/#
root@3d8631b74516:/#
root@3d8631b74516:/# mysql -uroot -puXee8phiovoohoom4maethoi9thee0hu
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.18 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql>
```

Docker networking basic

- Docker networks uses bridge
- All container hooks to default bridge that created during the installation [docker0]
- Any container started with option –p or --publish , will perform NAT / Port Forwarding to the container from the host machine
- This are done automatically
- Should be used as default for development
- New private network should be created for efficient networking between containers

```
[root@servera ~]# ip a show docker0
5: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:b8:20:34:1f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:b8ff:fe20:341f/64 scope link
       valid_lft forever preferred_lft forever
```

Docker networking basic



Docker networking basic

- These are docker network command used to manipulate docker default network
- docker network ls : Show Networks for Docker
- docker network inspect : Show / Inspect which container is connected to a network
- docker network create : Create a new private network
- docker network connect : Attach a container to new network
- docker network disconnect : detach a container from network
- NOTE: new network is recommended if you need to use docker build in DNS resolver based on container naming

Docker networking basic

```
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                   NAMES
6c6fb3f28e4c        nginx               "nginx -g 'daemon of…"   6 hours ago         Up 6 hours          0.0.0.0:80->80/tcp                      proxy
ff2388a9aee8        httpd               "httpd-foreground"       6 hours ago         Up 6 hours          0.0.0.0:7171->80/tcp                    webserver
3d8631b74516        mysql               "docker-entrypoint.s…"   6 hours ago         Up 6 hours          0.0.0.0:3306->3306/tcp, 33060/tcp       db
[root@servera ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ecc864badb01        bridge              bridge              local
5c6ae9b2b226        host                host                local
443f19ca1a9d        none                null                local
```

```
[root@servera ~]# docker network inspect bridge | grep Containers -A 100
        "Containers": {
            "3d8631b745162839714a8690b868aef59b489f938405d27cf5fb6dfed40d4130": {
                "Name": "db",
                "EndpointID": "099e0e7b5899b787de02ce24e39ce698bd5e6e451f9f4be51f9bb3c98906d153",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            },
            "6c6fb3f28e4c0f46ed17ce0bf38b06b1e5044569628cc526ebea33fd3e5ca6bf": {
                "Name": "proxy",
                "EndpointID": "7aacbe846e36102083ec74d6644e9566c374c96747bf1752520106511f1f6eac",
                "MacAddress": "02:42:ac:11:00:04",
                "IPv4Address": "172.17.0.4/16",
                "IPv6Address": ""
            },
            "ff2388a9aee81ef926c42ff87fdb9ce5b8c7814c6727ca518f8c2e49a98290f2": {
                "Name": "webserver",
                "EndpointID": "5b2e067a24fb275709c37e596ea3841f1047c0de12e040d0f8d4b846be8e67c9",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
                "IPv6Address": ""
```

Docker networking basic

```
[root@servera ~]# docker network create myappnet
9a29445d410cf917a234458c800fc0418bf54ddb6cf49d6dfe4806a3cec6504f
[root@servera ~]#
[root@servera ~]# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ecc864badb01        bridge              bridge              local
5c6ae9b2b226        host                host                local
9a29445d410c        myappnet            bridge              local
443f19ca1a9d        none                null                local
[root@servera ~]#
```

```
[root@servera ~]# docker network connect myappnet db
[root@servera ~]# docker network connect myappnet webserver
[root@servera ~]#
[root@servera ~]# docker network disconnect bridge db
[root@servera ~]# docker network disconnect bridge webserver
```

Docker networking basic

- From previous example run
- Container db and webserver are connected to custom private network called myappnet
- Therefore, container db and webserver can resolve each other names using build in docker resolver
- But, since container proxy still connected to default bridge(which has no DNS) , webserver and db container may need to use IP address or rely on external name resolver to reach proxy container

```
[root@servera ~]# docker container exec -it webserver  bash
root@ff2388a9aee8:/usr/local/apache2#
root@ff2388a9aee8:/usr/local/apache2#
root@ff2388a9aee8:/usr/local/apache2# ping db
PING db (172.19.0.2) 56(84) bytes of data.
64 bytes from db.myappnet (172.19.0.2): icmp_seq=1 ttl=64 time=0.050 ms
64 bytes from db.myappnet (172.19.0.2): icmp_seq=2 ttl=64 time=0.057 ms
64 bytes from db.myappnet (172.19.0.2): icmp_seq=3 ttl=64 time=0.046 ms
^C
--- db ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.046/0.051/0.057/0.004 ms
root@ff2388a9aee8:/usr/local/apache2#
root@ff2388a9aee8:/usr/local/apache2# ping proxy
ping: proxy: Name or service not known
root@ff2388a9aee8:/usr/local/apache2#
root@ff2388a9aee8:/usr/local/apache2#
```

Practice B : Container for CLI testing

- Install new package in container
- Use centos:7 and ubuntu:14.04 and install curl in each container
- This practice requires you to pull both images and run interactively to install the package curl
- Ensure curl is installed and latest version for that distro
  - Centos : yum install curl
  - Ubuntu: apt-get update && apt-get install curl
- Verify curl version by [curl --version]

Practice B : Solution

```
[root@servera ~]# docker container run --rm -it centos:7 bash
Unable to find image 'centos:7' locally
7: Pulling from library/centos
ab5ef0e58194: Downloading [==================================>                ]  53.14MB/75.78MB
```

```
[root@servera ~]# docker container run --rm -it ubuntu:14.04 bash
Unable to find image 'ubuntu:14.04' locally
14.04: Pulling from library/ubuntu
2e6e20c8e2e6: Downloading [====================>                              ]  28.51MB/70.69MB
30bb187ac3fc: Download complete
b7a5bcc4a58a: Download complete
```

Practice B : Solution

```
[root@servera ~]# docker container run --rm -it centos:7 bash
Unable to find image 'centos:7' locally
7: Pulling from library/centos
ab5ef0e58194: Pull complete
Digest: sha256:4a701376d03f6b39b8c2a8f4a8e499441b0d567f9ab9d58e4991de4472fb813c
Status: Downloaded newer image for centos:7
[root@54a6c6d9ded5 /]#
[root@54a6c6d9ded5 /]#
[root@54a6c6d9ded5 /]# yum update curl
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
 * base: centos.ipserverone.com
 * extras: centos.ipserverone.com
 * updates: centos.ipserverone.com
```

```
[root@servera ~]# docker container run --rm -it ubuntu:14.04 bash
Unable to find image 'ubuntu:14.04' locally
14.04: Pulling from library/ubuntu
2e6e20c8e2e6: Pull complete
30bb187ac3fc: Pull complete
b7a5bcc4a58a: Pull complete
Digest: sha256:ffc76f71dd8be8c9e222d420dc96901a07b61616689a44c7b3ef6a10b7213de4
Status: Downloaded newer image for ubuntu:14.04
root@be0913d9397f:/#
root@be0913d9397f:/#
root@be0913d9397f:/# apt-get update && apt-get install curl
Get:1 http://security.ubuntu.com trusty-security InRelease [65.9 kB]
Ign http://archive.ubuntu.com trusty InRelease
Get:2 http://security.ubuntu.com trusty-security/main amd64 Packages [1032 kB]
Get:3 http://archive.ubuntu.com trusty-updates InRelease [65.9 kB]
```

Practice B : Solution

```
[root@54a6c6d9ded5 /]# cat  /etc/*rel*
CentOS Linux release 7.7.1908 (Core)
Derived from Red Hat Enterprise Linux 7.7 (Source)
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"

cat: /etc/prelink.conf.d: Is a directory
CentOS Linux release 7.7.1908 (Core)
CentOS Linux release 7.7.1908 (Core)
cpe:/o:centos:centos:7
[root@54a6c6d9ded5 /]# curl --version
curl 7.29.0 (x86_64-redhat-linux-gnu) libcurl/7.29.0 NSS/3.44 zlib/1.2.7 libidn/1.28 libssh2/1.8.0
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtsp scp sftp smtp smtps telnet tftp
Features: AsynchDNS GSS-Negotiate IDN IPv6 Largefile NTLM NTLM_WB SSL libz unix-sockets
[root@54a6c6d9ded5 /]#
```

Practice B : Solution

```
root@be0913d9397f:/# cat /etc/*rel*
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.6 LTS"
NAME="Ubuntu"
VERSION="14.04.6 LTS, Trusty Tahr"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 14.04.6 LTS"
VERSION_ID="14.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
root@be0913d9397f:/# curl --version
curl 7.35.0 (x86_64-pc-linux-gnu) libcurl/7.35.0 OpenSSL/1.0.1f zlib/1.2.8 libidn/1.28 librtmp/2.3
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp smtp smtps telnet tftp
Features: AsynchDNS GSS-Negotiate IDN IPv6 Largefile NTLM NTLM_WB SSL libz TLS-SRP
root@be0913d9397f:/#
```

Practice C : DNS Round Robin

- Since Docker engine 1.XX , you can have multiple containers on a created network respond to same DNS address , this creates a DNS Round Robin
- DNS Round Robin is a mechanism to reach any host with single naming [kind of doing cheap load balancing]
- The practice :
- Create a new virtual network [name it mynet]
- Pull elasticsearch:2 and create 2 container from that image
- Pull centos image and create one container
- All container must be attached to mynet private network
- Use option --network-alias when creating the container to give them additional DNS name to respond
- Use centos container curl to conform the load balancing

Practice C : Solution

```
[root@servera ~]# docker network create mynet
860edc537a9fdec3a0fd67a8ea4a7a0d13121b25194d4ec645fd7901a5a5abaf
[root@servera ~]#
[root@servera ~]# docker pu
pull  push
[root@servera ~]# docker pull elasticsearch:2
2: Pulling from library/elasticsearch
05d1a5232b46: Pull complete
5cee356eda6b: Pull complete
```

```
[root@servera ~]# docker container run -d --net mynet --net-alias search elasticsearch:2
ded89236eafc37e94a18267f510375406a78e7c1d1032edd15b1e1b391a2d84c
[root@servera ~]#
[root@servera ~]# docker container run -d --net mynet --net-alias search elasticsearch:2
c40fbeeb79ea8a438c2576c578a6f597663700d4090c0c37d96f90b98d75f564
[root@servera ~]#
[root@servera ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                   NAMES
c40fbeeb79ea        elasticsearch:2     "/docker-entrypoint.…"   10 seconds ago      Up 8 seconds        9200/tcp, 9300/tcp                      stupefied_mccarthy
ded89236eafc        elasticsearch:2     "/docker-entrypoint.…"   13 seconds ago      Up 11 seconds       9200/tcp, 9300/tcp                      keen_goldwasser
6c6fb3f28e4c        nginx               "nginx -g 'daemon of…"   7 hours ago         Up 7 hours          0.0.0.0:80->80/tcp                      proxy
ff2388a9aee8        httpd               "httpd-foreground"       7 hours ago         Up 7 hours          0.0.0.0:7171->80/tcp                    webserver
3d8631b74516        mysql               "docker-entrypoint.s…"   8 hours ago         Up 8 hours          0.0.0.0:3306->3306/tcp, 33060/tcp       db
[root@servera ~]#
```

Practice C: Solution

```
[root@servera ~]# docker container run --net mynet --rm -it  centos:7 bash
[root@b07d59036bc9 /]#
```

```
[root@b07d59036bc9 /]# curl -s search:9200
{
  "name" : "Locust",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "3ffU3fZrR3a5ETuaTHCkpw",
  "version" : {
    "number" : "2.4.6",
    "build_hash" : "5376dca9f70f3abef96a77f4bb22720ace8240fd",
    "build_timestamp" : "2017-07-18T12:17:44Z",
    "build_snapshot" : false,
    "lucene_version" : "5.5.4"
  },
  "tagline" : "You Know, for Search"
}
[root@b07d59036bc9 /]# curl -s search:9200
{
  "name" : "Titan",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "7pr4RywhQG-DW2zOFAWnUg",
  "version" : {
    "number" : "2.4.6",
    "build_hash" : "5376dca9f70f3abef96a77f4bb22720ace8240fd",
    "build_timestamp" : "2017-07-18T12:17:44Z",
    "build_snapshot" : false,
    "lucene_version" : "5.5.4"
  },
  "tagline" : "You Know, for Search"
}
```

Creating Images through Docker

Docker Hub

What's in this section?

- What's in an image?
- Docker HUB Registry images
- Images and Layers
- Image Tagging and Pushing to Docker HUB
- Building images : dockerfile basic
- Building images : running docker builds

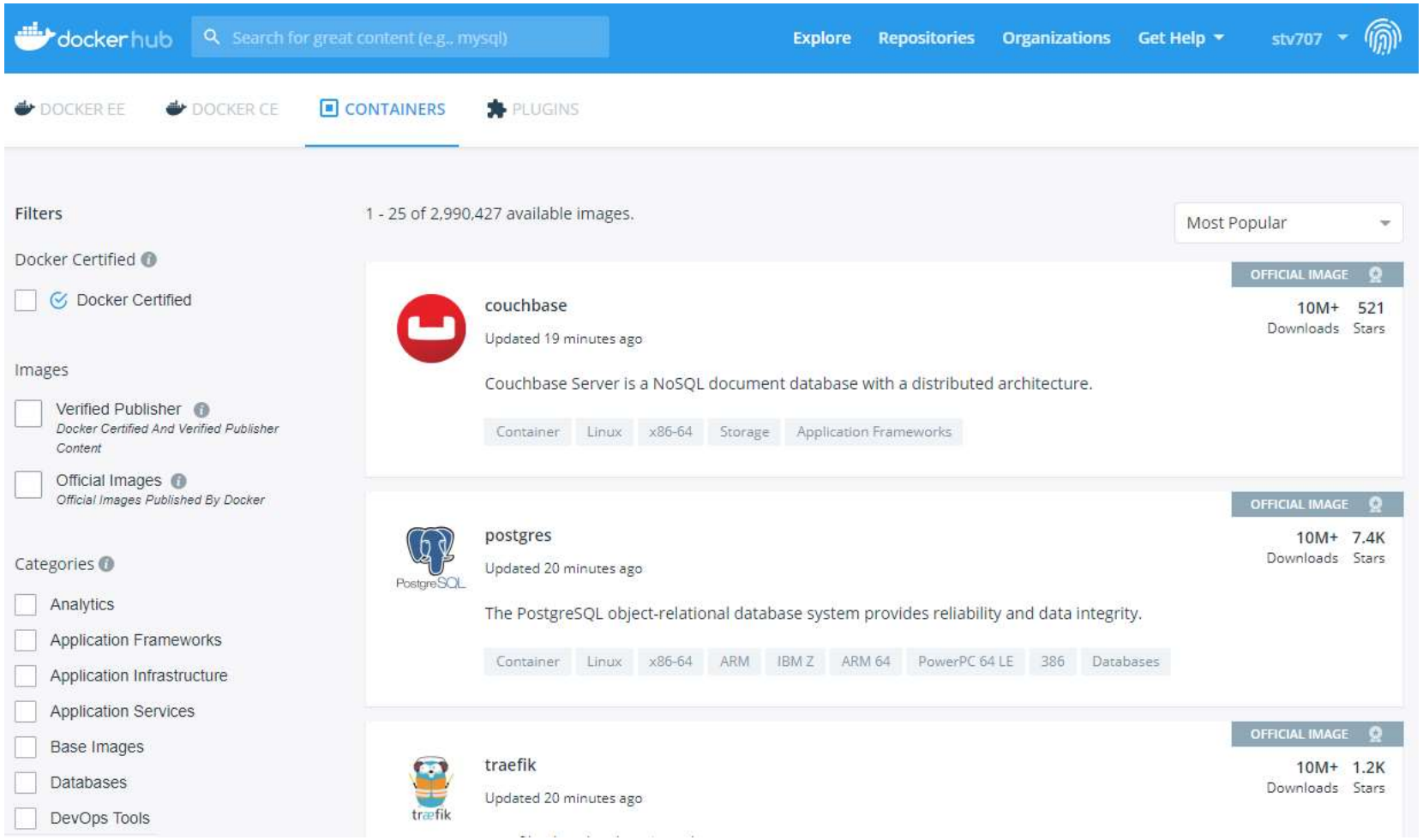
What's in an image?

- App binaries and dependencies
- Metadata about the image data and how to run the image
- Official definition: "An Image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime."
- Not a complete OS. No kernel, kernel modules (e.g. drivers)
- Small as one file (your app binary) like a golang static binary
- Big as a Ubuntu distro with apt, and Apache, PHP, and more installed

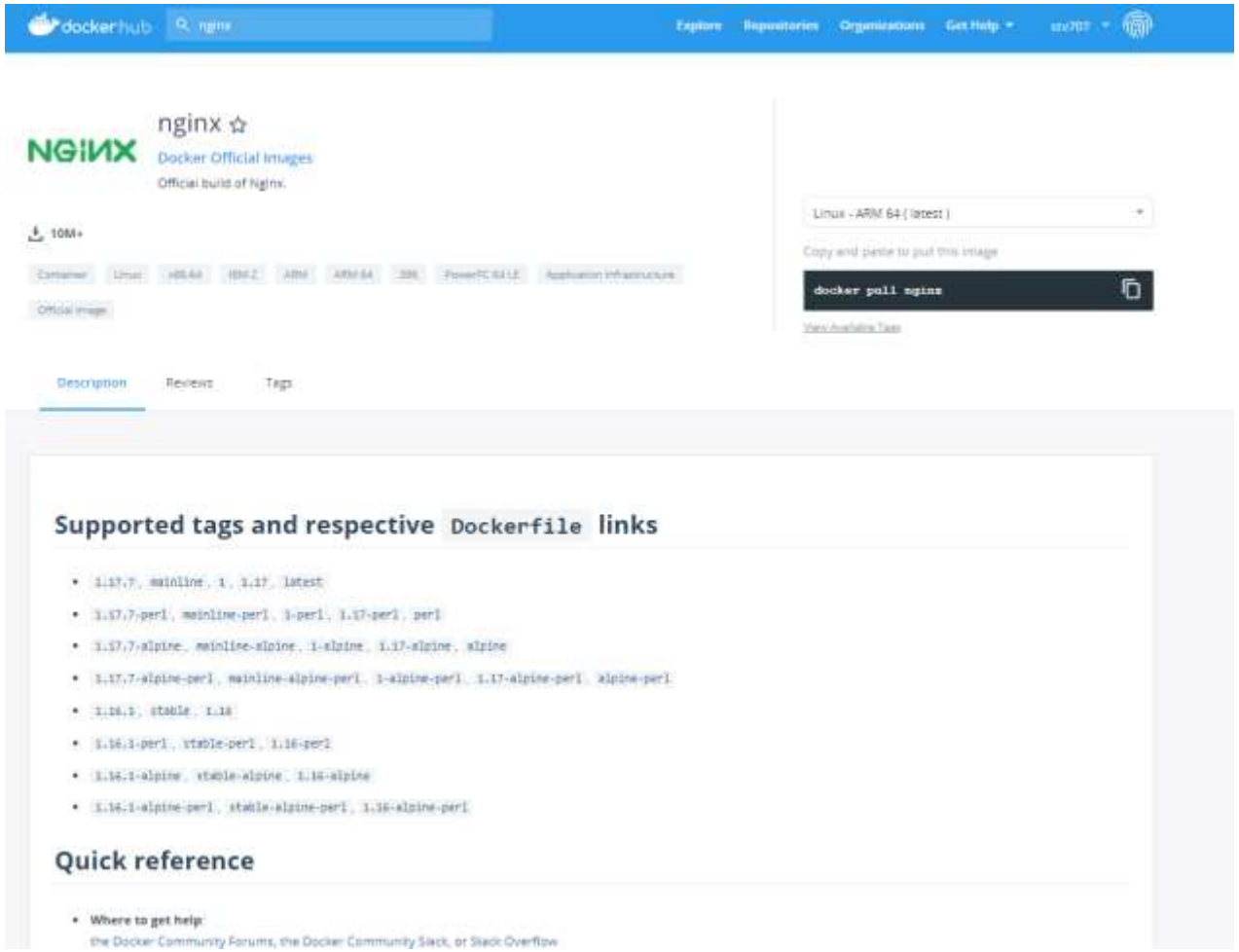
Docker HUB Registry images

- The Registry is a stateless, highly scalable server side application that stores and lets you distribute Docker images. The Registry is open-source, under the permissive Apache license.
- You should use the Registry if you want to:
  - tightly control where your images are being stored
  - fully own your images distribution pipeline
  - integrate image storage and distribution tightly into your in-house development workflow
- Alternatives
- Users looking for a zero maintenance, ready-to-go solution are encouraged to head-over to the Docker Hub, which provides a free-to-use, hosted Registry
- Requirements: The Registry is compatible with Docker engine version 1.6.0 or higher.
- Requirements: Need username and password [free account]

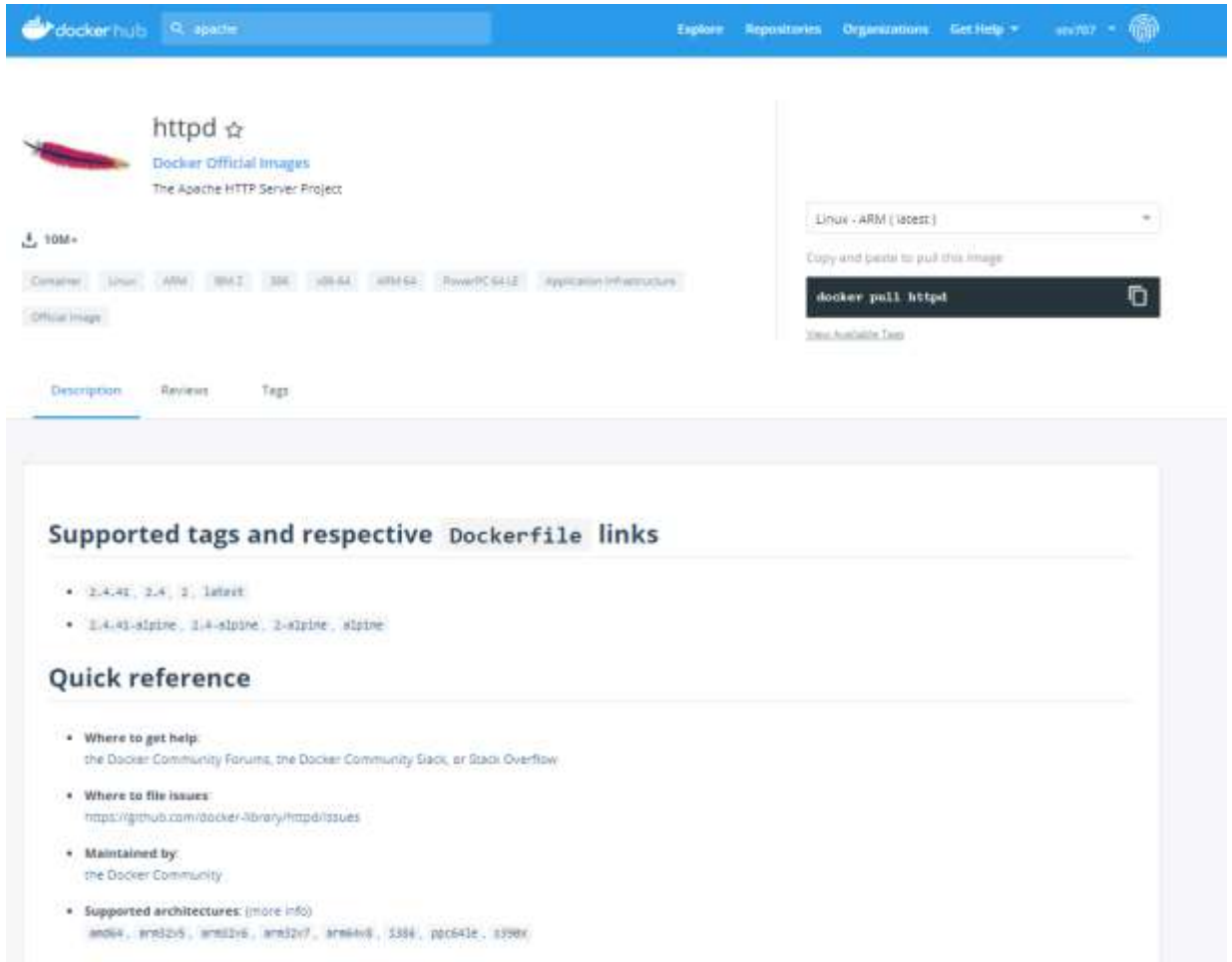
Docker HUB Registry images



Docker HUB Registry images

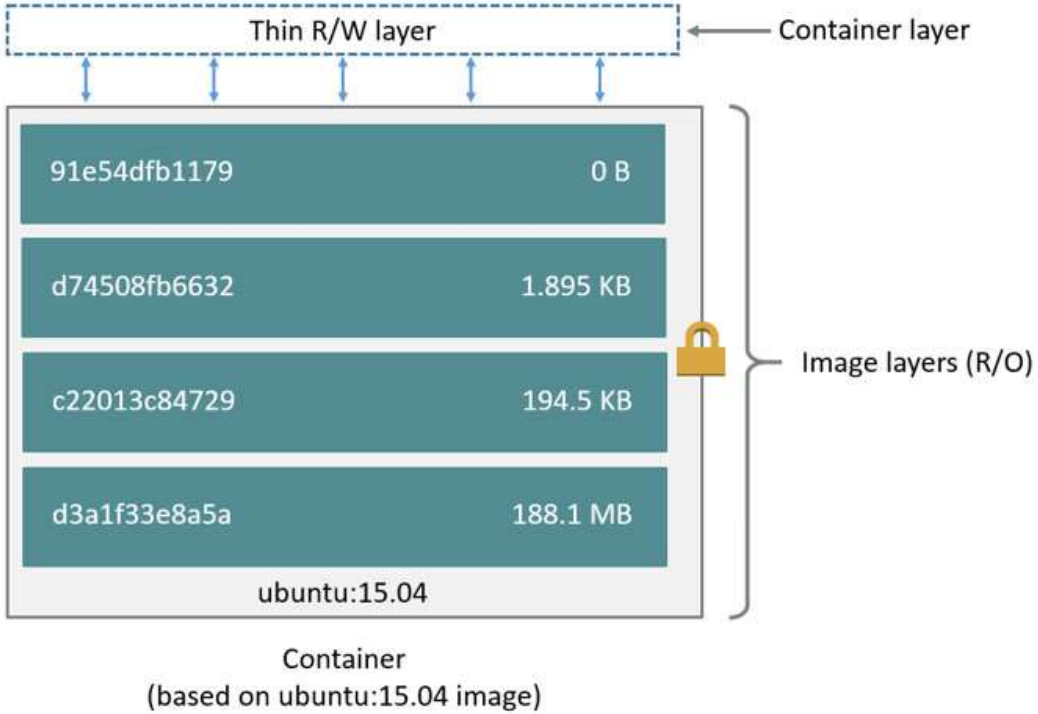


Docker HUB Registry images



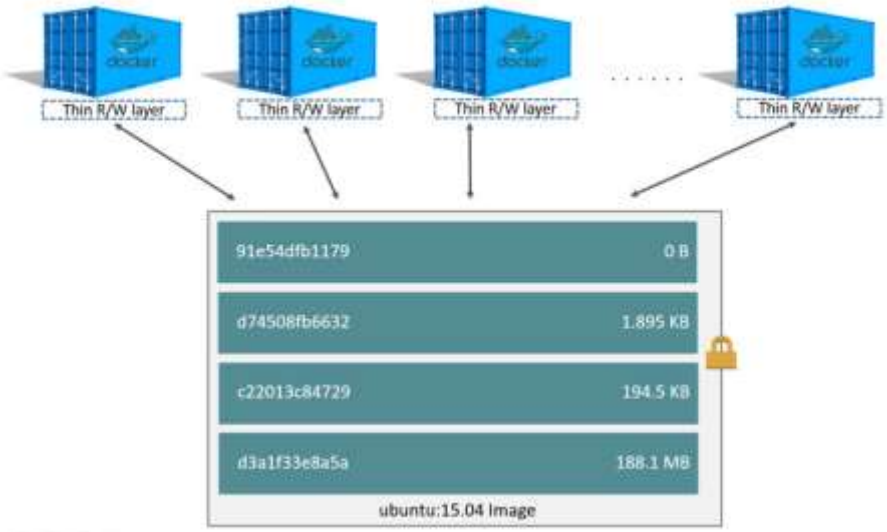
Images and Layers

- A Docker image is built up from a series of layers.
- Each layer represents an instruction in the image's Dockerfile.
- Each layer except the very last one is read-only



Images and Layers

- The major difference between a container and an image is the top writable layer.
- All writes to the container that add new or modify existing data are stored in this writable layer.
- When the container is deleted, the writable layer is also deleted.
- The underlying image remains unchanged



Images and Layers

```
[root@servera ~]# docker image ls
REPOSITORY                      TAG                 IMAGE ID            CREATED             SIZE
stevect707.azurecr.io/mynginx   v1                  6c4967222fb1        16 hours ago        121MB
stv707/nginx                    testing             c7460dfcab50        2 days ago          126MB
mysql                           latest              ed1ffcb5eff3        2 weeks ago         456MB
httpd                           latest              c2aa7e16edd8        2 weeks ago         165MB
```

```
[root@servera ~]# docker history stv707/nginx:testing
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
c7460dfcab50        2 days ago          /bin/sh -c #(nop)  CMD ["nginx" "-g" "daemon…   0B
<missing>           2 days ago          /bin/sh -c #(nop)  STOPSIGNAL SIGTERM           0B
<missing>           2 days ago          /bin/sh -c #(nop)  EXPOSE 80                    0B
<missing>           2 days ago          /bin/sh -c ln -sf /dev/stdout /var/log/nginx…   22B
<missing>           2 days ago          /bin/sh -c set -x     && addgroup --system -…   57.1MB
<missing>           2 days ago          /bin/sh -c #(nop)  ENV PKG_RELEASE=1~buster     0B
<missing>           2 days ago          /bin/sh -c #(nop)  ENV NJS_VERSION=0.3.7        0B
<missing>           2 days ago          /bin/sh -c #(nop)  ENV NGINX_VERSION=1.17.7     0B
<missing>           2 weeks ago         /bin/sh -c #(nop)  LABEL maintainer=NGINX Do…   0B
<missing>           2 weeks ago         /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>           2 weeks ago         /bin/sh -c #(nop) ADD file:04caaf303199c81ff…   69.2MB
```

Image Tagging and Pushing to Docker HUB

- What are Docker tags?
- In simple words, Docker tags convey useful information about a specific image version/variant.
- They are aliases to the ID of your image which often look like this: f1477ec11d12.
- It's just a way of referring to your image

Image Tagging and Pushing to Docker HUB

```
[root@servera ~]# docker image ls
REPOSITORY                      TAG                 IMAGE ID            CREATED             SIZE
stevect707.azurecr.io/mynginx   v1                  6c4967222fb1        16 hours ago        121MB
stv707/nginx                    testing             c7460dfcab50        2 days ago          126MB
mysql                           latest              ed1ffcb5eff3        2 weeks ago         456MB
httpd                           latest              c2aa7e16edd8        2 weeks ago         165MB
[root@servera ~]#
[root@servera ~]#
[root@servera ~]#
[root@servera ~]# docker image tag httpd stv707/myhttpd:testingv1
[root@servera ~]#
[root@servera ~]# docker image ls
REPOSITORY                      TAG                 IMAGE ID            CREATED             SIZE
stevect707.azurecr.io/mynginx   v1                  6c4967222fb1        16 hours ago        121MB
stv707/nginx                    testing             c7460dfcab50        2 days ago          126MB
mysql                           latest              ed1ffcb5eff3        2 weeks ago         456MB
httpd                           latest              c2aa7e16edd8        2 weeks ago         165MB
stv707/myhttpd                  testingv1           c2aa7e16edd8        2 weeks ago         165MB
```

Image Tagging and Pushing to Docker HUB

- Before pushing the image to docker hub
- You are required to login using docker login
- Credentials will be saved in $HOME/.docker/
- Docker hub before push

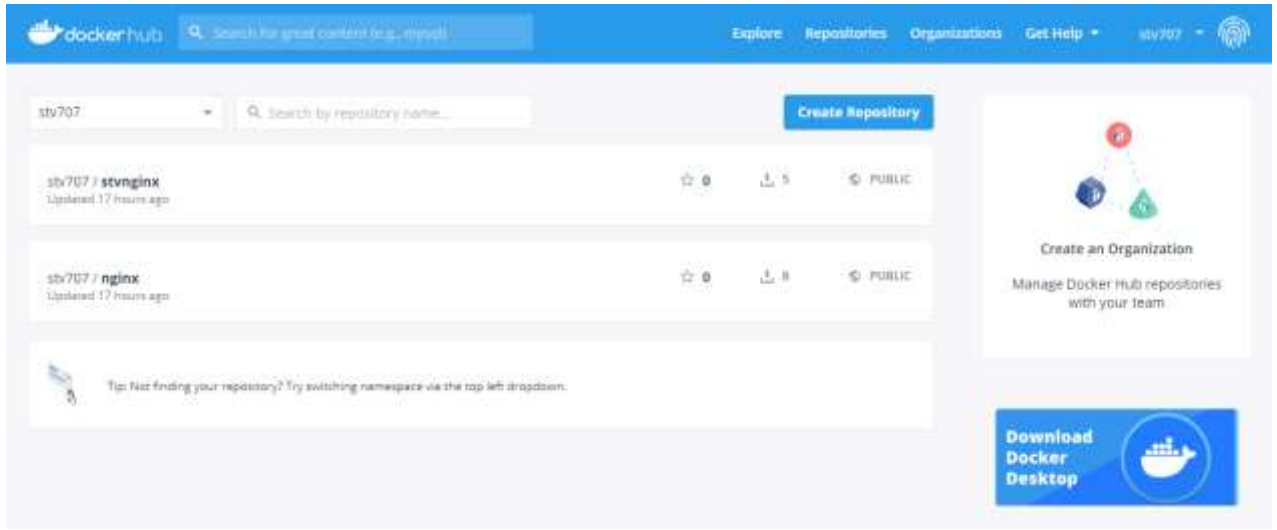


Image Tagging and Pushing to Docker HUB

- Docker Hub after the push

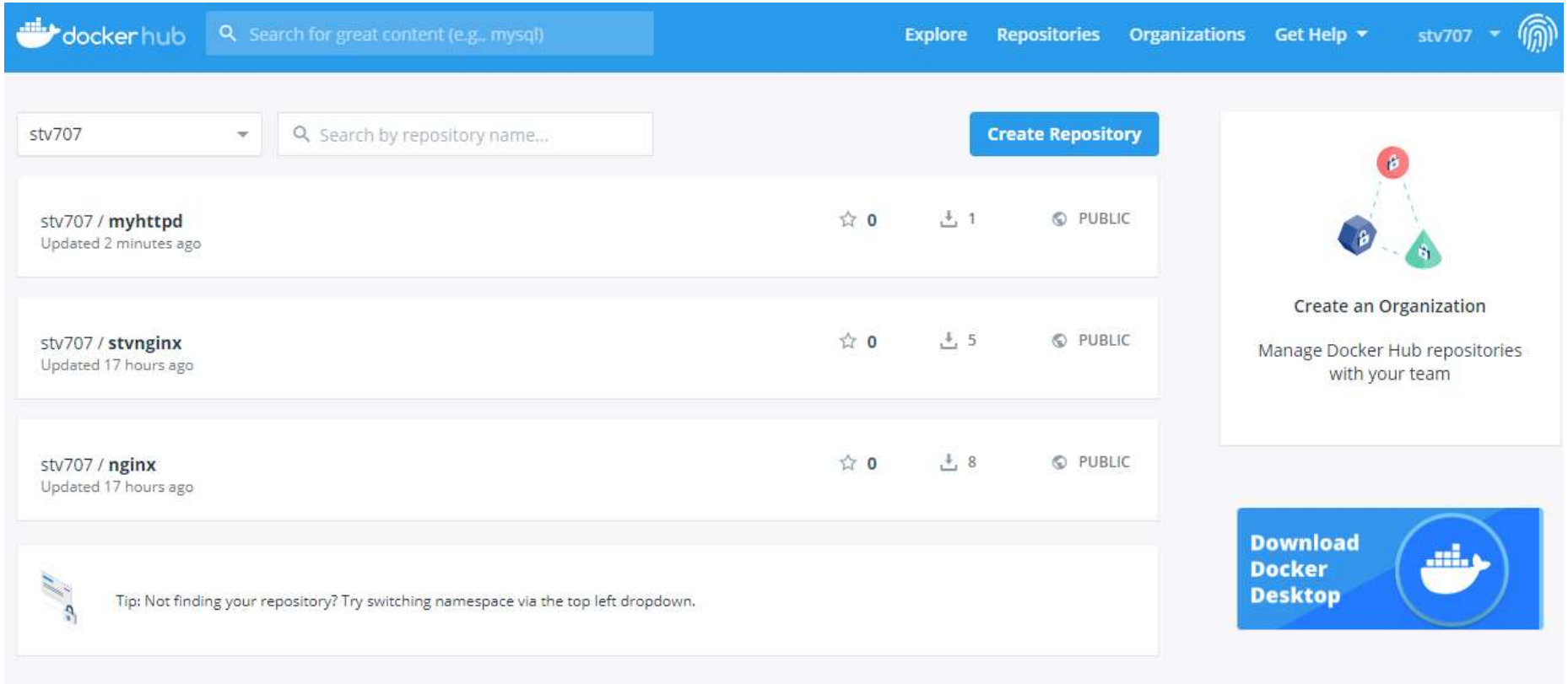


Image Tagging and Pushing to Docker HUB

```
[root@servera ~]# docker image ls
REPOSITORY                      TAG                 IMAGE ID            CREATED             SIZE
stevect707.azurecr.io/mynginx   v1                  6c4967222fb1        17 hours ago        121MB
stv707/nginx                    testing             c7460dfcab50        2 days ago          126MB
mysql                           latest              ed1ffcb5eff3        2 weeks ago         456MB
httpd                           latest              c2aa7e16edd8        2 weeks ago         165MB
stv707/myhttpd                  testingv1           c2aa7e16edd8        2 weeks ago         165MB
[root@servera ~]# docker image tag  mysql stv707/mysql:v1
[root@servera ~]# docker image ls
REPOSITORY                      TAG                 IMAGE ID            CREATED             SIZE
stevect707.azurecr.io/mynginx   v1                  6c4967222fb1        17 hours ago        121MB
stv707/nginx                    testing             c7460dfcab50        2 days ago          126MB
mysql                           latest              ed1ffcb5eff3        2 weeks ago         456MB
stv707/mysql                    v1                  ed1ffcb5eff3        2 weeks ago         456MB
httpd                           latest              c2aa7e16edd8        2 weeks ago         165MB
stv707/myhttpd                  testingv1           c2aa7e16edd8        2 weeks ago         165MB
[root@servera ~]#
[root@servera ~]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, h
Username: stv707
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[root@servera ~]# docker image push stv707/mysql:v1
The push refers to repository [docker.io/stv707/mysql]
162c1d1d22e7: Preparing
ce965e81b9f9: Preparing
```

Building images : dockerfile basic

- Docker can build images automatically by reading the instructions from a Dockerfile.
- A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.
- Using docker build users can create an automated build that executes several command-line instructions in succession.
- Common practice is to build from Official Image and you proceed to add on your own code and required libraries and packages
- Ref : https://docs.docker.com/engine/reference/builder/

Building images : dockerfile basic

- Example Docker File :

```
FROM centos:7

# Install Apache
RUN yum -y update
RUN yum -y install httpd httpd-tools

# Install EPEL Repo
RUN rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm \
 && rpm -Uvh https://mirror.webtatic.com/yum/el7/webtatic-release.rpm

# Install PHP
RUN yum -y install php72w php72w-bcmath php72w-cli php72w-common php72w-gd php72w-intl php72w-ldap php72w-mbstring \
    php72w-mysql php72w-pear php72w-soap php72w-xml php72w-xmlrpc

# Update Apache Configuration
RUN sed -E -i -e '/<Directory "\/var\/www\/html">/,/<\/Directory>/s/AllowOverride None/AllowOverride All/' /etc/httpd/conf/httpd.conf
RUN sed -E -i -e 's/DirectoryIndex (.*)$/DirectoryIndex index.php \1/g' /etc/httpd/conf/httpd.conf

EXPOSE 80

# Start Apache
CMD ["/usr/sbin/httpd","-D","FOREGROUND"]
```

Building images : dockerfile basic

- FROM – to define what base image to build this new image from
- ENV – Variables that can be defined for later use inside the Dockerfile
- RUN – commands or task to run when building an image
- WORKDIR – declare working directory inside image (used with copy , otherwise you define path)
- COPY – copy any files from external location to inside the image [Code, Scripts…]
- EXPOSE – open port in image
- ENTRYPOINT – to run custom script
- CMD – final command to run when the image is starting

Practice D: build your own image

- We will build a simple PHP Calculator webpage
- PHP and HTTPD will be installed in the image
- Base image can any Linux (Centos Recommended)

Building images : Building your own image

- Steps :
- A) clone or download this source : https://github.com/stv707/k8_training.git

Building images : Building your own image

- Modify the Dockerfile to match below :

```
# Build image from official CentOS 7 Image
FROM centos:7

# Install Apache
RUN yum -y update
RUN yum -y install httpd httpd-tools

# Install EPEL Repo
RUN rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm \
 && rpm -Uvh https://mirror.webtatic.com/yum/el7/webtatic-release.rpm

# Install PHP
RUN yum -y install php72w php72w-bcmath php72w-cli php72w-common php72w-gd php72w-intl  php72w-mbstring

# Update Apache Configuration
RUN sed -E -i -e '/<Directory "\/var\/www\/html">/,/<\/Directory>/s/AllowOverride None/AllowOverride All/' /etc/httpd/conf/httpd.conf
RUN sed -E -i -e 's/DirectoryIndex (.*)$/DirectoryIndex index.php \1/g' /etc/httpd/conf/httpd.conf

# Set working dir to copy index.php
WORKDIR  /var/www/html/

# Copy index.php to WORKDIR
COPY index.php index.php

# open Port 80
EXPOSE 80

# Start Apache
CMD ["/usr/sbin/httpd","-D","FOREGROUND"]
```

Building images : Building your own image

- Use/Modify/Add index.php

```
<html>
<p> Welcome to simple PHP page </p>
<p> This page will sum up 2 numbers and give you the answer </p>


<form action="index.php" method="GET">
     Num1: <input type="number" name="num1">
     Num2: <input type="number" name="num2">
     <input type="submit">
</form>

<?php
     $num1 = $_GET["num1"];
     $num2 = $_GET["num2"];
     echo $num1 + $num2;
?>
```

Building images : Building your own image

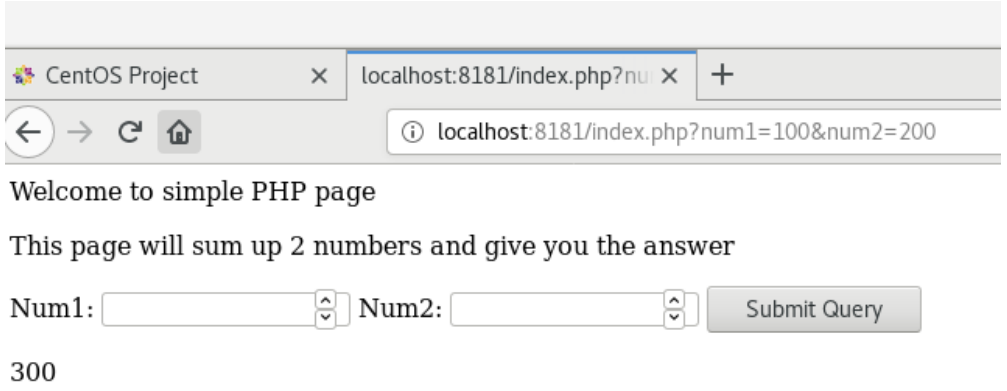
- End Result :

```
[root@servera dockerfile]# pwd
/root/dockerfile
[root@servera dockerfile]#
[root@servera dockerfile]#
[root@servera dockerfile]#
[root@servera dockerfile]# tree .
.
├── Dockerfile
└── index.php

0 directories, 2 files
[root@servera dockerfile]#
```

Building images : Building your own image

- Build your image and run the image and Verify the Image

```
[root@servera dockerfile]# docker build -t myphp2 .
Sending build context to Docker daemon  3.584kB
Step 1/11 : FROM centos:7
 ---> 5e35e350aded
Step 2/11 : RUN yum -y update
 ---> Using cache
 ---> ed99e45ead3e
Step 3/11 : RUN yum -y install httpd httpd-tools
 ---> Using cache
 ---> d72e0acf4497
Step 4/11 : RUN rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-rel
pm
 ---> Using cache
```




```
[root@servera dockerfile]# docker container  run -d -p 8181:80   myphp2
6e29ce677458ad6c2dae4686513034b16500a3573b2fc2c1f2bdb1bbd339c799
[root@servera dockerfile]#
[root@servera dockerfile]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
6e29ce677458        myphp2              "/usr/sbin/httpd -D …"   7 seconds ago       Up 7 seconds
```

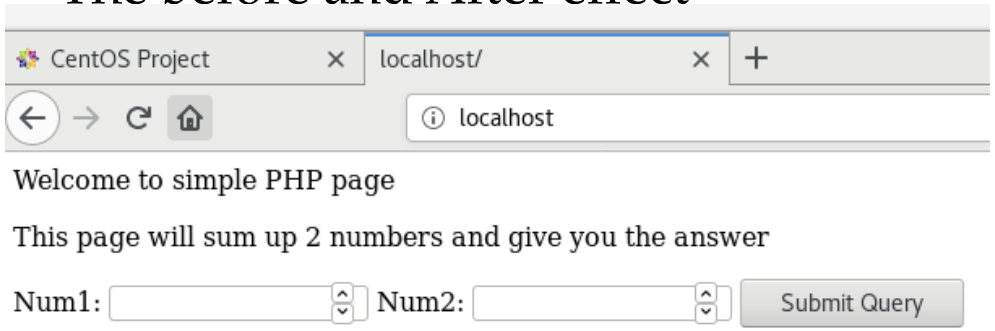
Additional docker image – bind mount

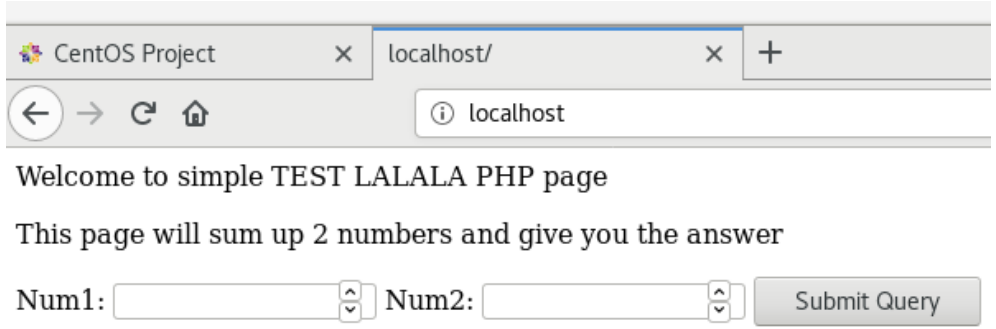
- For Developers, its best you start a container with code directory is bind mounted to your local machine
- With this option, you can develop / update / test your code from host machine and verify in Container

```
[root@servera code]# pwd
/root/code
[root@servera code]# ls
index.php
[root@servera code]# docker container run -d -p 80:80 -v /root/code/:/var/www/html  myphp
3ad4cdcd6ccb2e5f55d51513f4de32d2ecd2650a403de46e218c74a0792033cd
[root@servera code]#
[root@servera code]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
3ad4cdcd6ccb        myphp               "/usr/sbin/httpd -D …"   22 seconds ago      Up 22 seconds
[root@servera code]#
```

Additional docker image – bind mount

- Changes on local /root/code will be reflected to running container whenever there is changes
- The before and After effect






```
<html>
<p> Welcome to simple TEST LALALA PHP page </p>
<p> This page will sum up 2 numbers and give you the answer </p>


<form action="index.php" method="GET">
     Num1: <input type="number" name="num1">
     Num2: <input type="number" name="num2">
     <input type="submit">
```

Persistent Data: Bind Mounting/Volume

- Maps a host file or directory to container
- Using your local disk rather then containers UFS based disk
- Containers are immutable infrastructure
- For persistent data, aka DATABASE cannot be immutable
- Persistent Data is solution :
  - Volume
  - Run time Solution: bind volume

Persistent Data: Volume Mounting

```
[root@serverc ~]# docker volume ls
DRIVER              VOLUME NAME
[root@serverc ~]#
[root@serverc ~]#
[root@serverc ~]# docker run -d --name mysql mysql:latest
3a43b2e4311925e0f15f1d379bf7baa9f00f898c71d42fbc6e2ab85a4213142a
[root@serverc ~]#
[root@serverc ~]#
[root@serverc ~]# docker volume ls
DRIVER              VOLUME NAME
local               c4e92647ac3e0a91529e91977d397e76bb8be192ae849d8e63379b217c3f6c84
[root@serverc ~]#
[root@serverc ~]# df -h
Filesystem               Size  Used Avail Use% Mounted on
devtmpfs                 903M     0  903M   0% /dev
tmpfs                    919M   14M  905M   2% /dev/shm
tmpfs                    919M  9.6M  910M   2% /run
tmpfs                    919M     0  919M   0% /sys/fs/cgroup
/dev/mapper/centos-root   28G  5.0G   24G  18% /
/dev/vda1               1014M  237M  778M  24% /boot
tmpfs                    184M  4.0K  184M   1% /run/user/42
tmpfs                    184M   20K  184M   1% /run/user/0
overlay                   28G  5.0G   24G  18% /var/lib/docker/overlay2/0932bcc843297ebf0814a84569145fe0b4ed2c2680af3d004bb3cc080e992f1d/merged
```

Persistent Data: Volume Mounting

```
[root@serverc ~]# docker run -d --name mysql -e MYSQL_ROOT_PASSWORD=mypassword -v mysqlvol:/var/lib/mysql -d mysql:latest
d99c975fd26f2892c7038d4d912d2681abc814d0b85186e31594eb0bd84e922f
[root@serverc ~]#
[root@serverc ~]#
[root@serverc ~]# docker volume ls
DRIVER              VOLUME NAME
local               mysqlvol
```

Persistent Data: Volume Mounting

```
[root@serverc ~]# docker volume ls
DRIVER              VOLUME NAME
local               mysqlvol
[root@serverc ~]# docker volume inspect mysqlvol
[
    {
        "CreatedAt": "2020-01-15T04:19:20+08:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/mysqlvol/_data",
        "Name": "mysqlvol",
        "Options": null,
        "Scope": "local"
    }
]
[root@serverc ~]# cd /var/lib/docker/volumes/mysqlvol/_data
[root@serverc _data]# ls
auto.cnf       binlog.index  client-cert.pem  ibdata1      ibtmp1        mysql.ibd           public_key.pem   sys
binlog.000001  ca-key.pem    client-key.pem   ib_logfile0  #innodb_temp  performance_schema  server-cert.pem  undo_001
binlog.000002  ca.pem        ib_buffer_pool   ib_logfile1  mysql         private_key.pem     server-key.pem   undo_002
[root@serverc _data]#
```

Persistent Data: Volume Mounting

```
[root@serverc ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
d99c975fd26f        mysql:latest        "docker-entrypoint.s…"   4 minutes ago       Up 4 minutes
a79a3eaba97b        nginx               "nginx -g 'daemon of…"   15 minutes ago      Up 15 minutes
[root@serverc ~]# docker container  exec -it  mysql  bash
root@d99c975fd26f:/#
root@d99c975fd26f:/#
root@d99c975fd26f:/# mysql -uroot -pmypassword
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.18 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database sometest
    -> ;
Query OK, 1 row affected (0.01 sec)

mysql> show databases ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sometest           |
| sys                |
```

Persistent Data: Volume Mounting

```
[root@serverc ~]# docker run -d --name mysql2 -e MYSQL_ROOT_PASSWORD=mypassword -v mysqlvol:/var/lib/mysql -d mysql:latest
5bca35563fcb824cf4dd03ee10946fadfd26d77f2e32a495d4eadfac7134e86c
[root@serverc ~]#
[root@serverc ~]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
5bca35563fcb        mysql:latest        "docker-entrypoint.s…"   5 seconds ago       Up 3 seconds        3306/tcp, 33060/tcp     mysql2
d99c975fd26f        mysql:latest        "docker-entrypoint.s…"   7 minutes ago       Up 7 minutes        3306/tcp, 33060/tcp     mysql
a79a3eaba97b        nginx               "nginx -g 'daemon of…"   18 minutes ago      Up 18 minutes       0.0.0.0:80->80/tcp      nginx
[root@serverc ~]#
```

Persistent Data: Volume Mounting

```
[root@serverc ~]# docker container  exec -it  mysql2  bash
root@5bca35563fcb:/# mysql -uroot -pmypassword
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.18 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sometest           |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

mysql>
```

Introduction to Docker Compose

docker-compose

What's in this section?

- Introduction to Docker Compose
- YAML file
- docker-compose sample usage
- Using compose to Build Image and bringing up the containers
- Practice E: enable docker-compose command
- Practice F : Using docker-compose to create multi-container environment – nginx-httpd-mysql
- Practice G : Using docker-compose to create multi-container environment – Drupal - Postgres

Introduction to Docker Compose

- Compose is a tool for defining and running multi-container Docker applications.
- With Compose, you use a YAML file to configure your application's services.
- Then, with a single command, you create and start all the services from your configuration.
- Compose works in all environments: production, staging, development, testing, as well as CI workflows.
- Using Compose is basically a three-step process:
  - A) Define your app's environment with a Dockerfile so it can be reproduced anywhere
  - B) Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment
  - C) Run docker-compose up and Compose starts and runs your entire app

Practice E: enable docker-compose command

- Install docker-compose on vm001
- Refer practice-E

YAML file concept for build setup

```yaml
version: '2'

services:
  proxy:
    image: nginx
    ports:
      - "8080:80"

  db:
    image: mysql
    environment:
     - MYSQL_ROOT_PASSWORD=mypassword

  webserver:
    image: httpd
    ports:
     - "80:80"
```

Version to match Docker Version
Version 3 is the current

Services – the containers we want to run

Container Specific settings and Naming

docker-compose sample usage

- Create the yaml file – you might not get it right the first time, try again

```yaml
version: '2'

services:
   proxy:
     image: nginx
     ports:
       - "8080:80"

   db:
     image: mysql
     environment:
      - MYSQL_ROOT_PASSWORD=mypassword

   webserver:
     image: httpd
     ports:
      - "80:80"
```

docker-compose sample usage

- Run docker-compose command with up

```
[root@servera compose1]# vim docker-compose.yaml
[root@servera compose1]# docker-compose up
Creating network "compose1_default" with the default driver
Pulling proxy (nginx:)...
latest: Pulling from library/nginx
8ec398bc0356: Pull complete
dfb2a46f8c2c: Pull complete
b65031b6a2a5: Pull complete
Digest: sha256:8aa7f6a9585d908a63e5e418dc5d14ae7467d2e36e1ab4f0d8f9d059a3d071ce
Status: Downloaded newer image for nginx:latest
Pulling db (mysql:)...
latest: Pulling from library/mysql
804555ee0376: Already exists
c53bab458734: Pull complete
ca9d72777f90: Pull complete
2d7aad6cb96e: Pull complete
8d6ca35c7908: Pull complete
6ddae009e760: Pull complete
327ae67bbe7b: Pull complete
0e26af624120: Pull complete
5e70feb9365d: Pull complete
f5595dde544e: Pull complete
87399808d2ba: Pull complete
7312ab6d79b5: Pull complete
Digest: sha256:e1b0fd480a11e5c37425a2591b6fbd32af886bfc6d6f404bd362be5e50a2e632
Status: Downloaded newer image for mysql:latest
```

docker-compose sample usage

- If the docker-compose ran successfully, it will pull all the image needed and will start the containers with all the settings supplied in docker-compose.yml file

```
[root@servera compose1]# docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
64120d81ce68        httpd               "httpd-foreground"       10 minutes ago      Up 10 minutes       0.0.0.0:80->80/tcp
776b39de8525        nginx               "nginx -g 'daemon of…"   10 minutes ago      Up 10 minutes       0.0.0.0:8080->80/tcp
41e840fdc5d0        mysql               "docker-entrypoint.s…"   10 minutes ago      Up 10 minutes       3306/tcp, 33060/tcp
```

- Naming of the containers are automated , if no settings supplied, images will use default

Using compose to Build Image and bringing up the containers

- Standard build image operation:

```
[root@servera compose2]#
[root@servera compose2]# tree .
.
├── Dockerfile
├── index.js
└── package.json

0 directories, 3 files
[root@servera compose2]# cat Dockerfile
FROM node:alpine

WORKDIR '/app'

COPY package.json .

RUN npm install

COPY . .

CMD ["npm", "start"]
[root@servera compose2]#
[root@servera compose2]# docker build -t mynode:v1 .
Sending build context to Docker daemon  4.096kB
Step 1/6 : FROM node:alpine
 ---> 364fb8e7f28a
Step 2/6 : WORKDIR '/app'
```

Using compose to Build Image and bringing up the containers

- Instead of using standard build operation, you can use docker-compose to build the image before bringing up the container
- In the docker-compose.yml, you are required to use build option to specify the build operation before bringing the container up

Using compose to Build Image and bringing up the containers

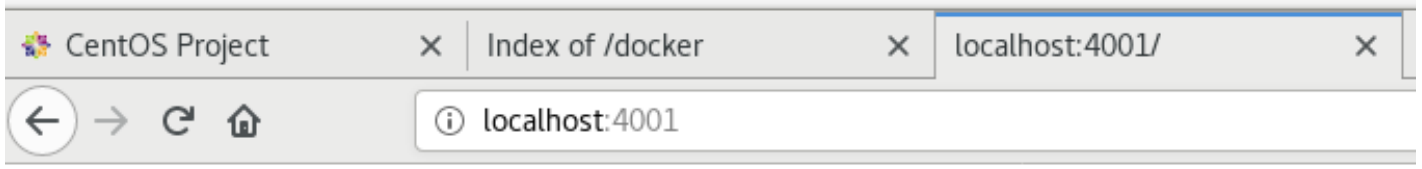
```
[root@servera compose2]# tree .
.
├── docker-compose.yml
├── Dockerfile
├── index.js
└── package.json

0 directories, 4 files
[root@servera compose2]# cat docker-compose.yml
version: '3'
services:
     redis-server:
        image: 'redis'
     node-app:
      build: .
      ports:
         - "4001:8081"
[root@servera compose2]#
```

```
[root@servera compose2]# docker-compose up
Starting compose2_redis-server_1 ... done
Starting compose2_node-app_1     ... done
Attaching to compose2_redis-server_1, compose2_node-app_1
redis-server_1   | 1:C 13 Jan 2020 10:46:34.426 # oO0OoO0OoO0Oo Redis is
redis-server_1   | 1:C 13 Jan 2020 10:46:34.426 # Redis version=5.0.7, bi
redis-server_1   | 1:C 13 Jan 2020 10:46:34.426 # Warning: no config file
path/to/redis.conf
redis-server_1   | 1:M 13 Jan 2020 10:46:34.428 * Running mode=standalone
redis-server_1   | 1:M 13 Jan 2020 10:46:34.428 # WARNING: The TCP backlo
er value of 128.
redis-server_1   | 1:M 13 Jan 2020 10:46:34.428 # Server initialized
redis-server_1   | 1:M 13 Jan 2020 10:46:34.428 # WARNING overcommit_memo
 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run t
redis-server_1   | 1:M 13 Jan 2020 10:46:34.428 # WARNING you have Transp
 usage issues with Redis. To fix this issue run the command 'echo never
n order to retain the setting after a reboot. Redis must be restarted af
redis-server_1   | 1:M 13 Jan 2020 10:46:34.430 * DB loaded from disk: 0.
redis-server_1   | 1:M 13 Jan 2020 10:46:34.430 * Ready to accept connect
node-app_1       |
node-app_1       | > @ start /app
node-app_1       | > node index.js
node-app_1       |
node-app_1       | Listening on port 8081
```

Using compose to Build Image and bringing up the containers

```
[root@servera compose2]# docker-compose up -d
Building node-app
Step 1/6 : FROM node:alpine
 ---> 364fb8e7f28a
Step 2/6 : WORKDIR '/app'
 ---> Running in fb24021ec51f
Removing intermediate container fb24021ec51f
 ---> 1fb87eed076d
Step 3/6 : COPY package.json .
 ---> 2c057d56af3b
Step 4/6 : RUN npm install
 ---> Running in c1a1bac45152
```

```
[root@servera compose2]# docker container ls
CONTAINER ID        IMAGE                 COMMAND
db6f8c18fa75        compose2_node-app     "docker-entrypoint.s…"
23b92ea8bfc7        redis                 "docker-entrypoint.s…"
[root@servera compose2]#
```



Number of visits is 11

Practice F : Using docker-compose to create multi-container environment – nginx-httpd-mysql

- Create a docker-compose.yml file that will bring up 3 containers
- 3 containers are:
  - Nginx that listens to port 8080 named as proxy
  - Mysql that uses 'mypass' as the password and named as mydb
  - Httpd service that uses port 80 and named as webserver
- All 3 containers are not connected to each other, this practice is about docker-compose.yml and understanding of how to bring up multiple container up in single command

Practice F: Solution

```
version: '2'

services:
   proxy:
     image: nginx
     ports:
       - "8080:80"

   mydb:
     image: mysql
     environment:
      - MYSQL_ROOT_PASSWORD=mypass

   webserver:
     image: httpd
     ports:
      - "80:80"
```

```
[root@servera ~]# docker-compose up -d
Creating network "root_default" with the default driver
Pulling proxy (nginx:)...
latest: Pulling from library/nginx
8ec398bc0356: Already exists
dfb2a46f8c2c: Pull complete
b65031b6a2a5: Pull complete
Digest: sha256:8aa7f6a9585d908a63e5e418dc5d14ae7467d2e36e1ab4f0d8f9d059a3d071ce
Status: Downloaded newer image for nginx:latest
Pulling mydb (mysql:)...
latest: Pulling from library/mysql
804555ee0376: Pull complete
c53bab458734: Pull complete
ca9d72777f90: Pull complete
2d7aad6cb96e: Pull complete
8d6ca35c7908: Pull complete
6ddae009e760: Pull complete
```

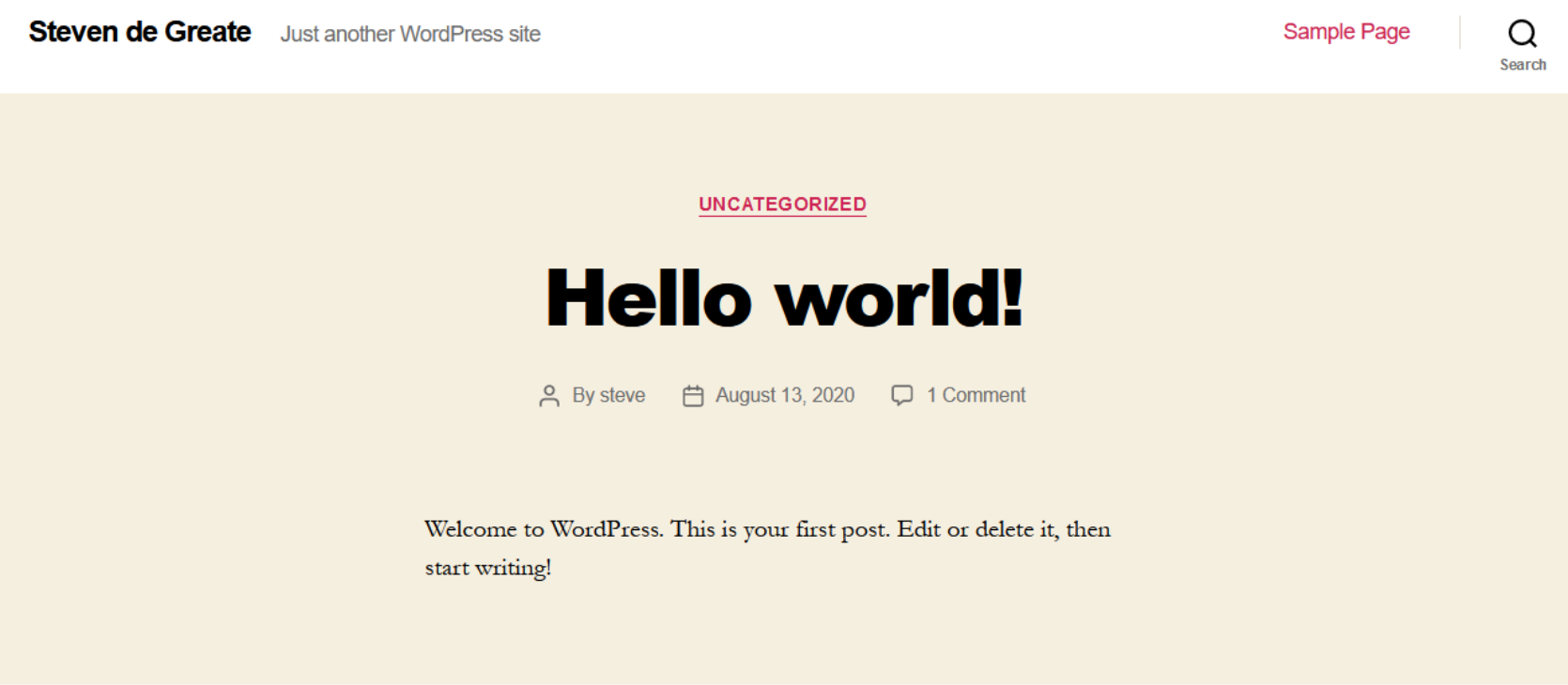
```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
c73bde809090        httpd               "httpd-foreground"       34 seconds ago      Up 32 seconds       0.0.0.0:80->80/tcp       root_webserver_1
5eaa6718e6f0        nginx               "nginx -g 'daemon of…"   34 seconds ago      Up 32 seconds       0.0.0.0:8080->80/tcp     root_proxy_1
d696c9b3e5c3        mysql               "docker-entrypoint.s…"   34 seconds ago      Up 32 seconds       3306/tcp, 33060/tcp      root_mydb_1
```

Practice G : Using docker-compose to create multi-container environment – WordPress & MySQL

- Setup 2 containers using WordPress CMS and MySQL Database as backend using Docker-compose
- The WordPress image need customization
  - Runs on port 8282
  - Volume name : wp_data mapped to ./wp_data
- The MySQL image need customization
  - Volume name: db_data mapped to ./db_data
- You also need to use volume to create separate volume to preserve the data on both WordPress and MySQL
  - This is required to keep data when you bring the down the container, when you start again, it will use the volumes and your data are preserved

Practice G: Solution

- Refer to Practice-G Solution Directory



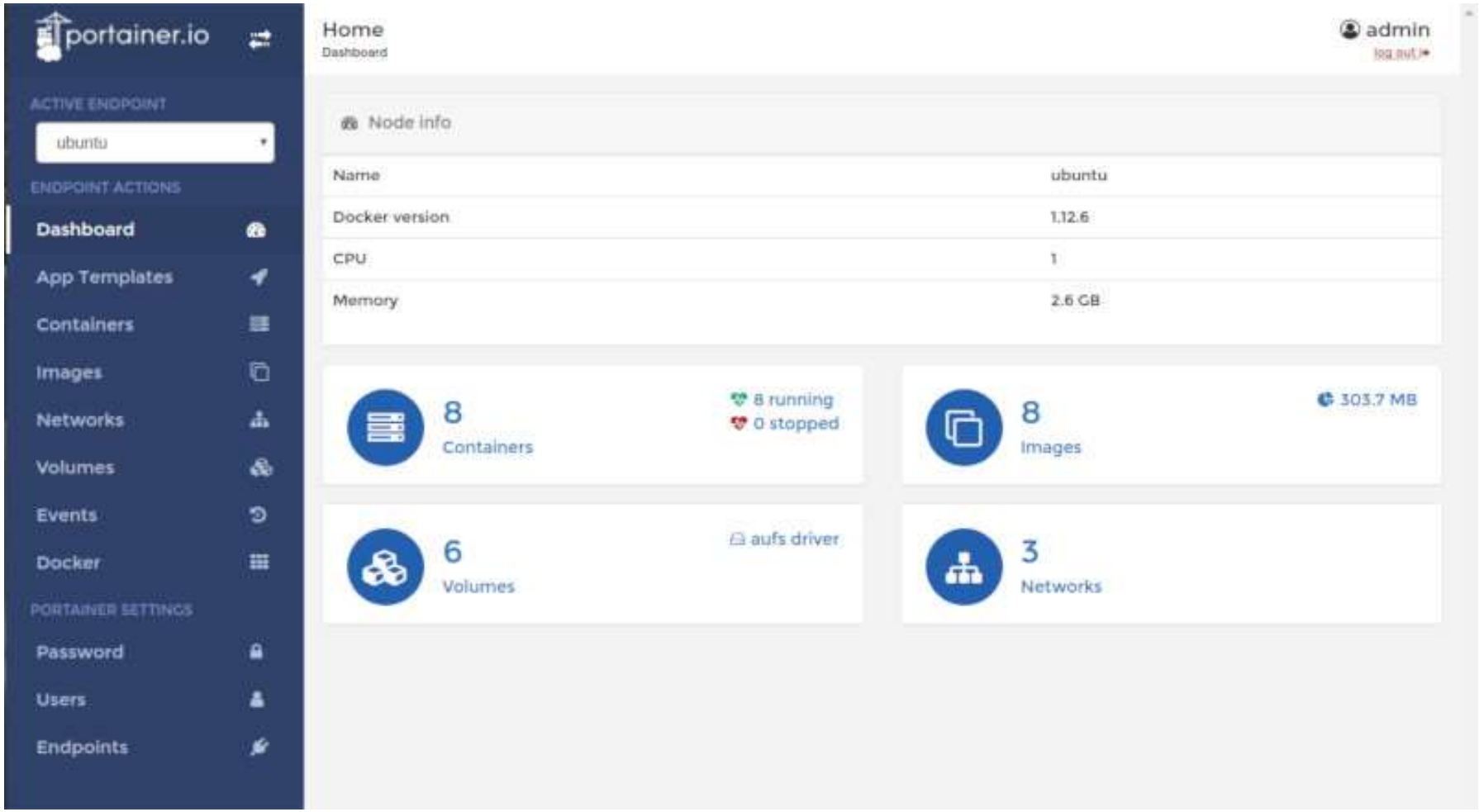
Portainer

Opensource Docker webUI

What is Portainer

- Portainer is a lightweight management UI which allows you to easily manage your Docker host.
- Portainer is meant to be as simple to deploy as it is to use.
- It consists of a single container that can run on any Docker engine (Docker for Linux and Docker for Windows are supported).
- Portainer allows you to manage your Docker stacks, containers, images, volumes, networks and more !
- It is compatible with the standalone Docker engine

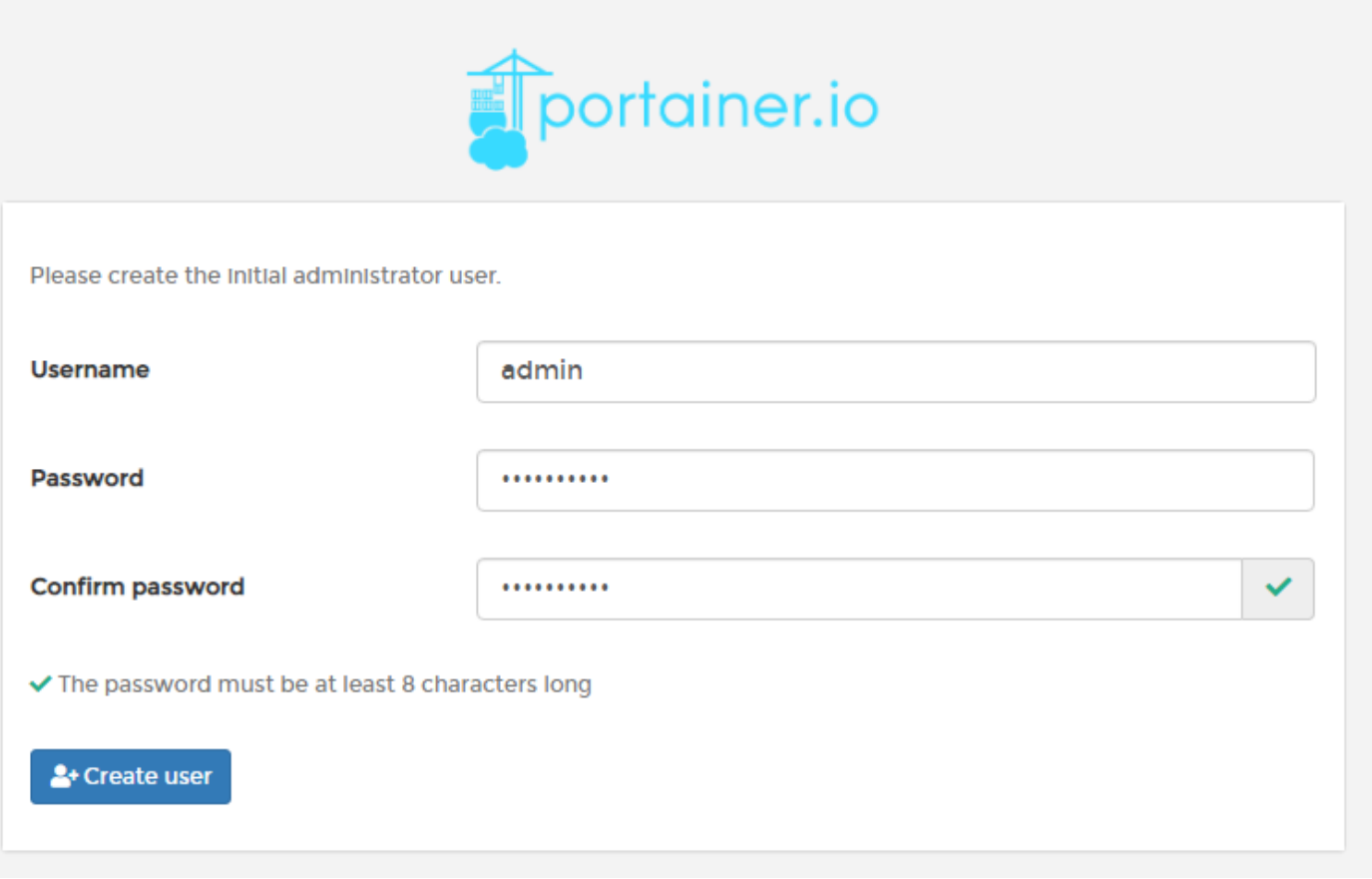
Portainer


portainer.io
Home
Dashboard
admin
log out
ACTIVE ENDPOINT
ubuntu
ENDPOINT ACTIONS
Dashboard
App Templates
Containers
Images
Networks
Volumes
Events
Docker
PORTAINER SETTINGS
Password
Users
Endpoints
Node info
Name
ubuntu
Docker version
1.12.6
CPU
1
Memory
2.6 GB
8
Containers
8 running
0 stopped
8
Images
303.7 MB
6
Volumes
aufs driver
3
Networks

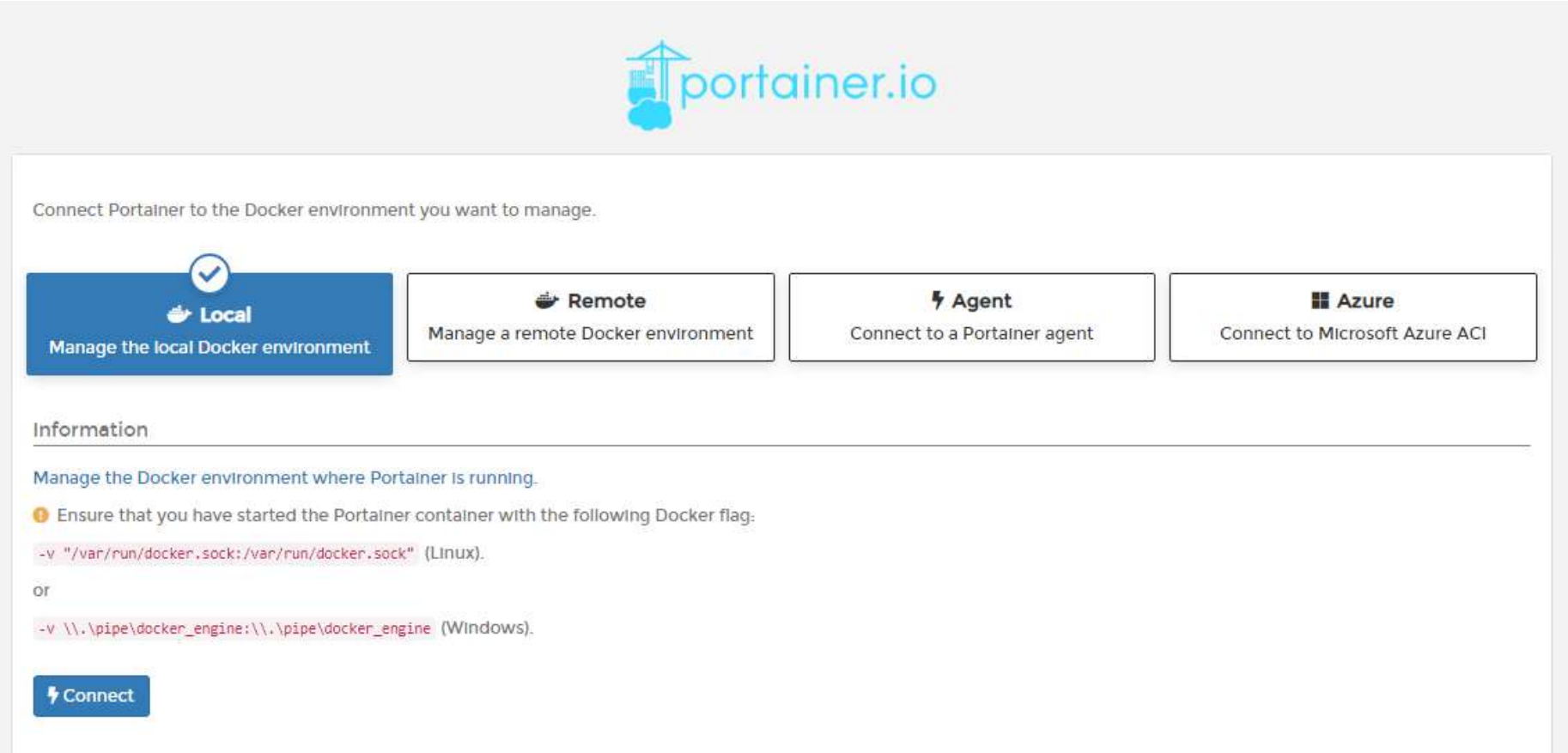
Practice H: Install and configure Portainer

- Please navigate to practice-H
- Follow the steps

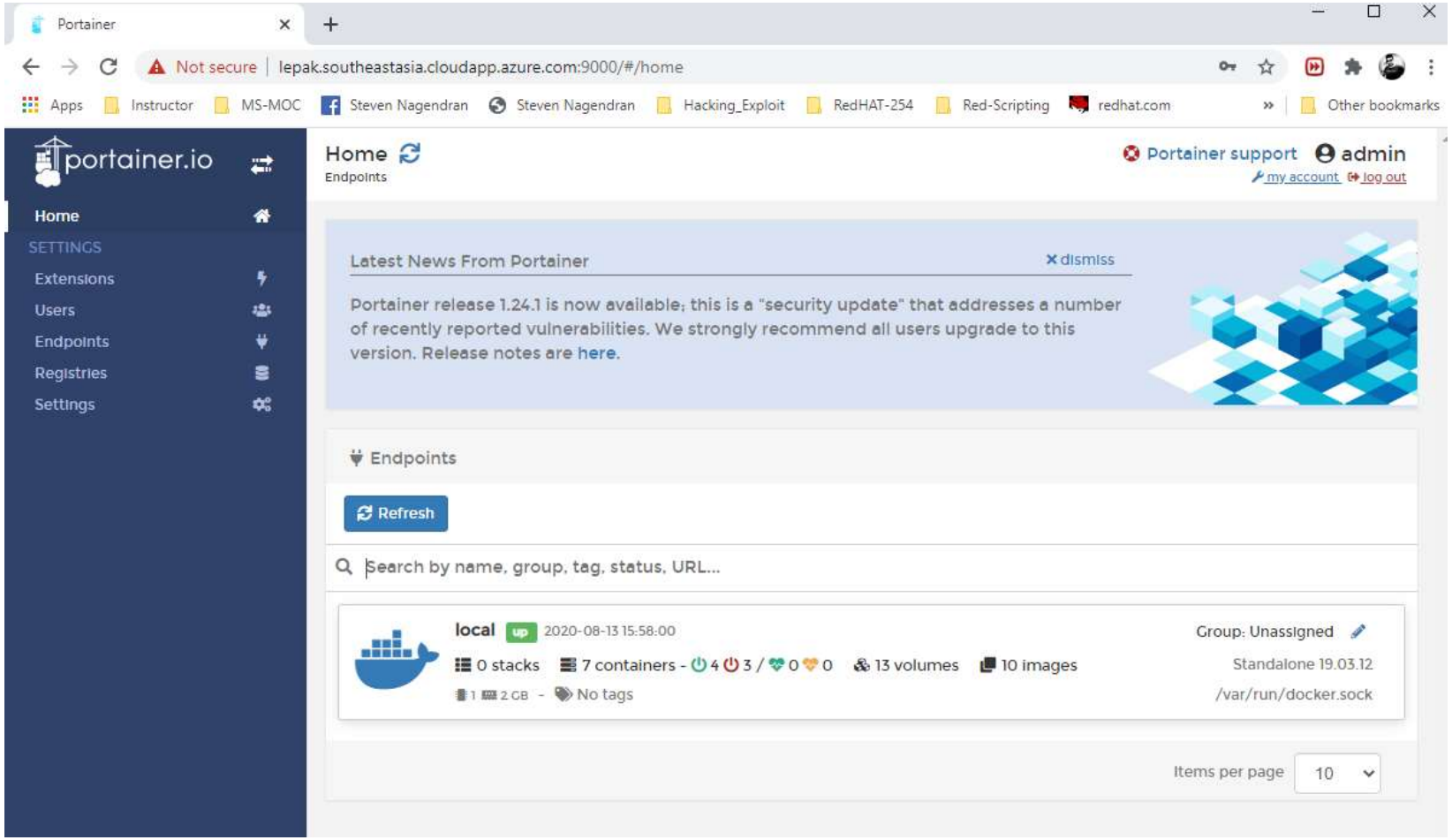
Practice H: Install and configure Portainer



Practice H: Install and configure Portainer



Practice H: Install and configure Portainer



The End

End of Day 1