

Kindly complete the pre-test from your RPS Lab machine

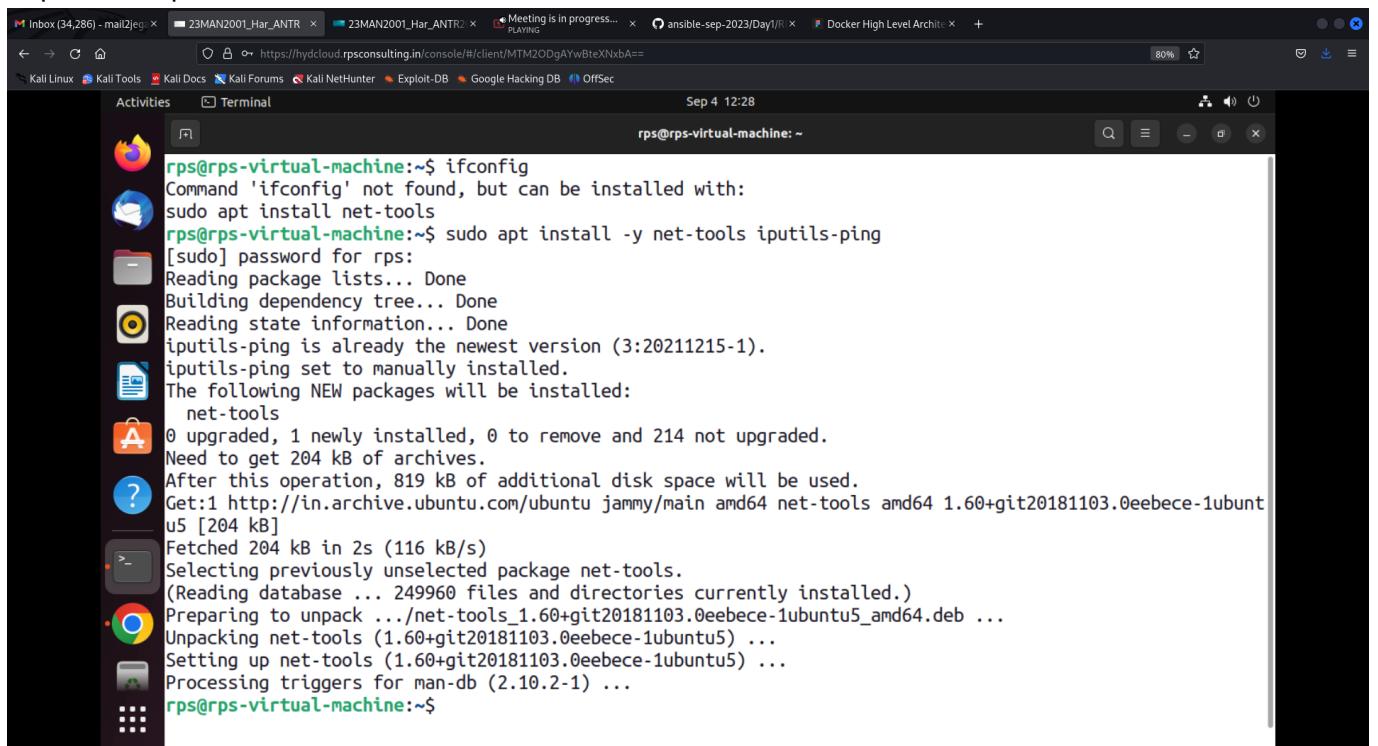
- Once you login to your RPS Lab machine, you will see a web browser with the below URL kept open
- Copy/Paste from your work/personal machine to/from RPS Lab machine is disabled as per your company policy
- Kindly provide your full name and personal email id while registering for the pre-test
- Once you complete the pre-test, kindly confirm me either orally or leave a message via WebEx chat if you have access to WebEx chat
- Once everyone completes the pre-test we can start the training until then I'll be on mute
- For any lab acces issues, please write to the RPS Tech Team user logged in via WebEx chat, you may have to share the RPS cloud user name and describe the technical challenge to

```
https://app.mymapit.in/code4/tiny/VxOr50
```

Installing network tools in Ubuntu lab machine

```
sudo apt install -y net-tools iutils-ping
```

Expected output



```
rps@rps-virtual-machine:~$ ifconfig
Command 'ifconfig' not found, but can be installed with:
sudo apt install net-tools
rps@rps-virtual-machine:~$ sudo apt install -y net-tools iutils-ping
[sudo] password for rps:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iutils-ping is already the newest version (3:20211215-1).
iutils-ping set to manually installed.
The following NEW packages will be installed:
    net-tools
0 upgraded, 1 newly installed, 0 to remove and 214 not upgraded.
Need to get 204 kB of archives.
After this operation, 819 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 net-tools amd64 1.60+git20181103.0eebece-1ubuntu5 [204 kB]
Fetched 204 kB in 2s (116 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 249960 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20181103.0eebece-1ubuntu5_amd64.deb ...
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Setting up net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Processing triggers for man-db (2.10.2-1) ...
rps@rps-virtual-machine:~$
```

Docker Overview

What are Bootloaders?

- are system utilities that helps in setting up dual/multi boot in your laptop/desktop/workstation
- i.e you can install 3~4 Operating System on your laptop/desktop/workstation
- but only one OS can be active at any point of time
- If you switch on your laptop with once the BIOS POST(Power On Self Test) completes, the BIOS will instruct the CPU to execute the system utility at Sector 0, Byte 0 in your Hard Disk. The Sector 0, Byte 0 is 512 bytes and it is referred as Master Boot Record (MBR)
- the Boot Loader System utility is installed on the MBR of your Hard disk
- Boot Loader will scan your Hard disk searching for Operating Systems, if it detects more than one Operating System, then it gives a menu for you to choose to select which OS you wish to boot into
- Let's say you booted machine into Windows, but you need Ubuntu, you first have to shutdown Windows and boot into Ubuntu and vice versa
- Only one OS can be active at any point of time
- Examples
 - LILO (Linux Loader - opensource)
 - GRUB1 - opensource
 - GRUB2 - opensource
 - For Mac users, BootCamp you can boot into windows on a Mac OS-X

What is Hypervisor?

- many OS can be running side-by-side on the same laptop/desktop/workstation/servers
- i.e many OS can be active at the same time
- Virtualization technology
- this is a combination of Hardware + Software technology
- Processors
 - AMD
 - AMD-V - Virtualization Feature set supported in the Processor
 - Intel
 - VT-X - Virtualization Feature set supported in the Processor
- there are two types of Hypervisors
 1. Type1 - used in Servers (Bare-metal Hypervisors - can be installed directly on a HW)
 2. Type2 - used in Laptops/Desktops/Workstations (requires Host OS - Unix/Linux/Windows/Mac)
- Examples VMWare
 - Workstation - Linux & Windows(type2)
 - Fusion - Mac OS-X (type2)
 - vSphere/vCenter - Bare-metal Hypervisor (type 1) Oracle virtualbox (type2 - Free) Parallels (type2 - Mac OS-X) Microsoft Hyper-V
- each Virtual Machine(VM)/Guest OS represents one fully functional Operating System
- each Virtual Machine(VM) gets its own IP address
- each Virtual Machine has its own shell, port range and Network stack

In the absence of Virtualization technology, how many minimal physical servers are required to support 1000 active OS?

1000 Physical servers are required

With Virtualization technology, how many minimal physical servers are required to support 1000 OS?

250 Physical Servers - each supporting 4 Virtual Machines

Processors with multiple CPU Cores

- AMD Processors it comes with 128 CPU Cores

Processors comes in 2 form factors

- SCM (Single Chip Module) - One IC will host one Processor
- MCM (Multi Chip Module) - Once IC will host multiple Processors

Server Motherboards with 4 Sockets

Let's say MCM Processor with 4 Processor per IC Each Processor supporting 128 CPU Cores 1 Socket = 4 Processors with 128 CPU Cores on each Processor = $4 \times 128 \times 4 = 2048$

Hyperthreading - each Physical CPU Core is capable of running 2 threads parallelly Hence, Hypervisor softwares will see each Physical CPU Cores as 2 Virtual Cores

$2048 \times 2 = 4096$ virtual cores

MacBook Pro with Quad Core Processor, 16 GB RAM and 1 TB HDD

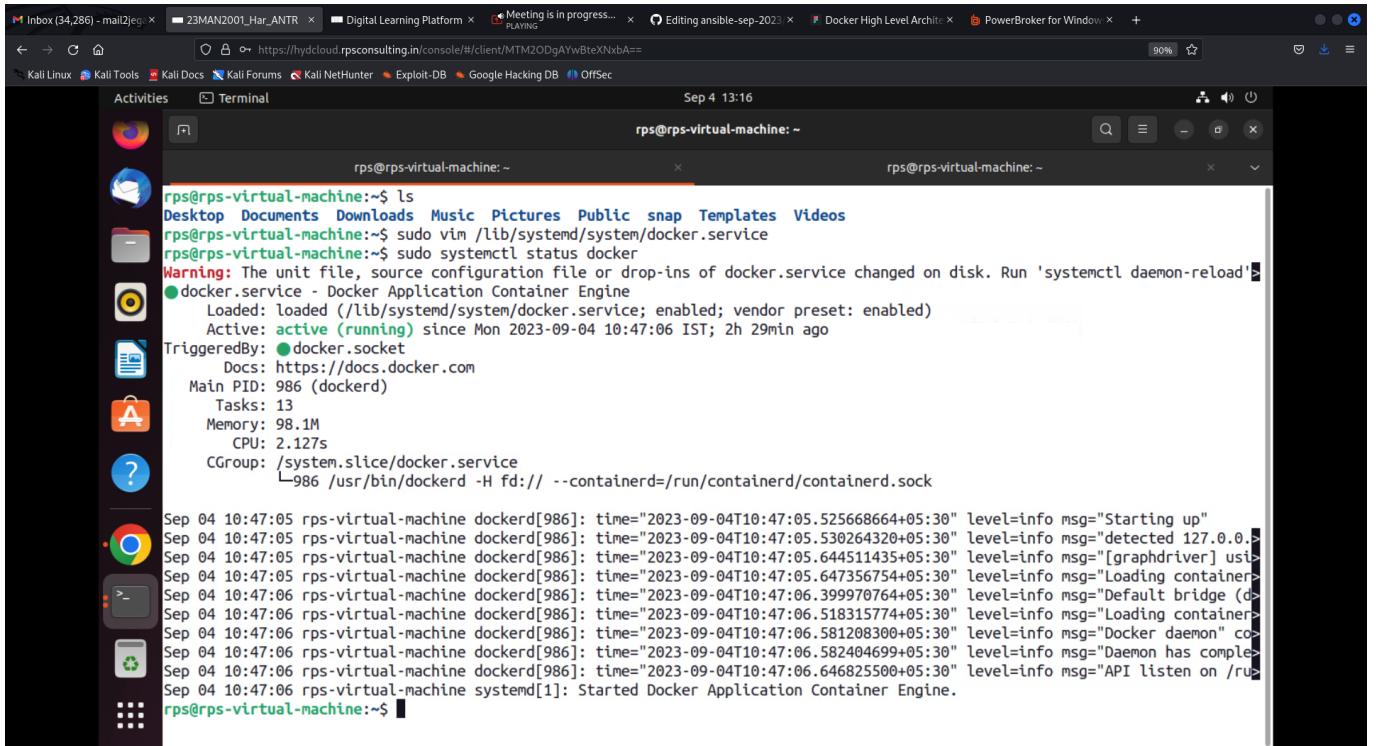
- I used to 2 VMS + 1 Host OS(Mac)

Enable the Docker Server REST API or TCP Socket

Check the status of docker service to find the path of docker service configuration file

```
sudo systemctl status docker
```

Expected output



The screenshot shows a terminal window titled 'rps@rps-virtual-machine: ~'. The user runs several commands to verify Docker's status:

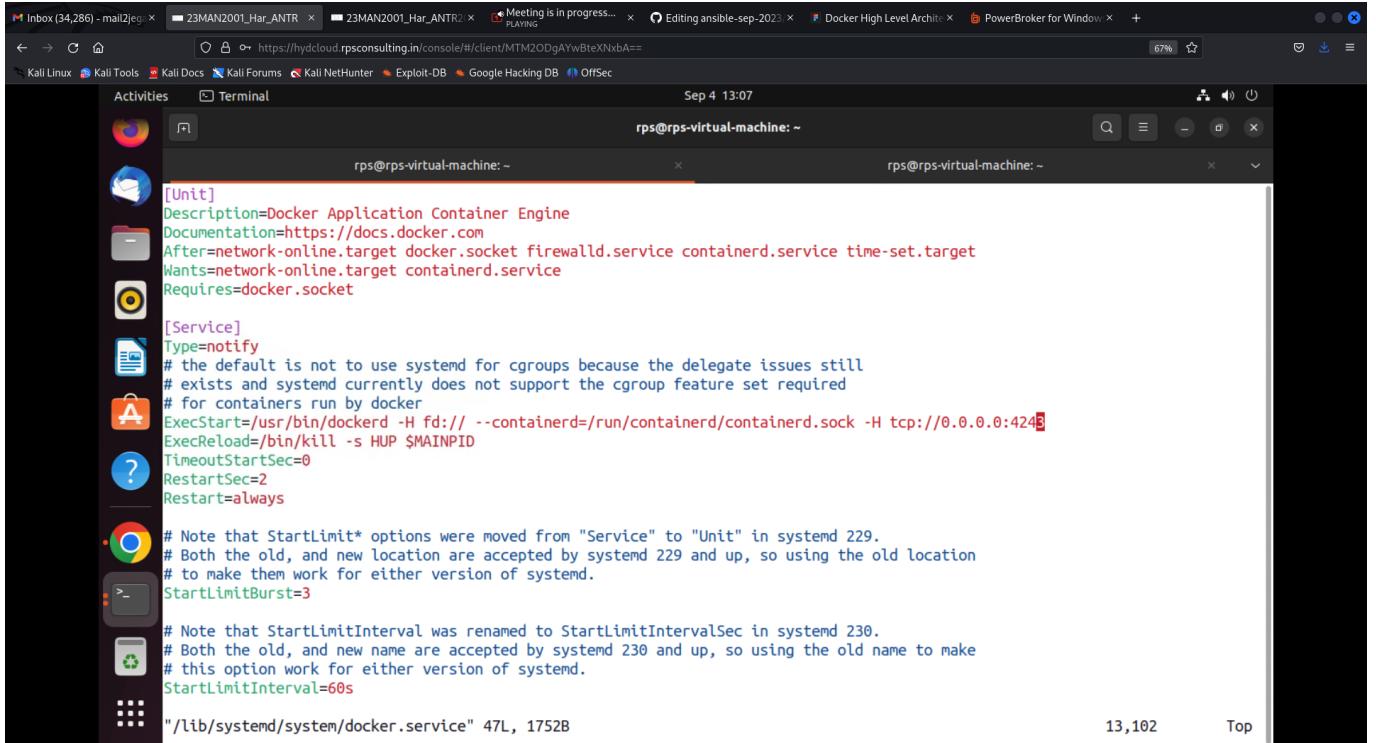
```
rps@rps-virtual-machine:~$ ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
rps@rps-virtual-machine:~$ sudo vim /lib/systemd/system/docker.service
rps@rps-virtual-machine:~$ sudo systemctl status docker
Warning: The unit file, source configuration file or drop-ins of docker.service changed on disk. Run 'systemctl daemon-reload' to reload unit files.
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2023-09-04 10:47:06 IST; 2h 29min ago
    TriggeredBy: ● docker.socket
  Docs: https://docs.docker.com
  Main PID: 986 (dockerd)
    Tasks: 13
      Memory: 98.1M
        CPU: 2.127s
      CGroup: /system.slice/docker.service
              └─986 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Sep 04 10:47:05 rps-virtual-machine dockerd[986]: time="2023-09-04T10:47:05.525668664+05:30" level=info msg="Starting up"
Sep 04 10:47:05 rps-virtual-machine dockerd[986]: time="2023-09-04T10:47:05.530264320+05:30" level=info msg="detected 127.0.0.1/8"
Sep 04 10:47:05 rps-virtual-machine dockerd[986]: time="2023-09-04T10:47:05.644511435+05:30" level=info msg="[graphdriver] using native storage driver"
Sep 04 10:47:05 rps-virtual-machine dockerd[986]: time="2023-09-04T10:47:05.647356754+05:30" level=info msg="Loading containerd"
Sep 04 10:47:06 rps-virtual-machine dockerd[986]: time="2023-09-04T10:47:06.399970764+05:30" level=info msg="Default bridge (docker0) assigned bridge port 58592"
Sep 04 10:47:06 rps-virtual-machine dockerd[986]: time="2023-09-04T10:47:06.518315774+05:30" level=info msg="Loading containerd"
Sep 04 10:47:06 rps-virtual-machine dockerd[986]: time="2023-09-04T10:47:06.581208300+05:30" level=info msg="Docker daemon" commit="0c35749d9d"
Sep 04 10:47:06 rps-virtual-machine dockerd[986]: time="2023-09-04T10:47:06.582404699+05:30" level=info msg="Daemon has completed initialization"
Sep 04 10:47:06 rps-virtual-machine dockerd[986]: time="2023-09-04T10:47:06.646825500+05:30" level=info msg="API listen on /run/docker.sock"
Sep 04 10:47:06 rps-virtual-machine systemd[1]: Started Docker Application Container Engine.
rps@rps-virtual-machine:~$
```

You can edit the `/lib/systemd/system/docker.service` file as Administrator and append `tcp://0.0.0.0:4243` at line 13 as shown below

```
sudo vim /lib/systemd/system/docker.service
```

Expected output



The screenshot shows a terminal window titled 'rps@rps-virtual-machine: ~'. The user has edited the Docker service file to add a new endpoint:

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target docker.socket firewalld.service containerd.service time-set.target
Wants=network-online.target containerd.service
Requires=docker.socket

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock -H tcp://0.0.0.0:4243
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutStartSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to "Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229 and up, so using the old location
# to make them work for either version of systemd.
StartLimitBurst=3

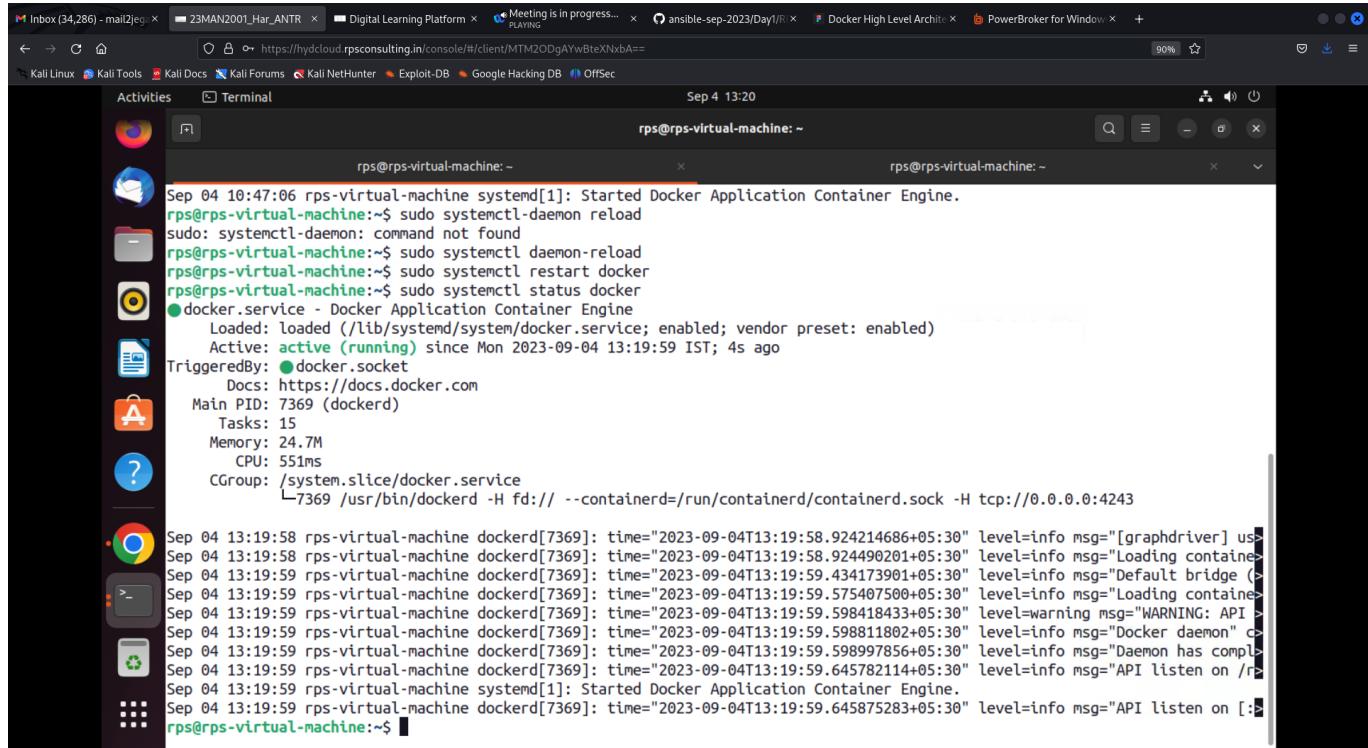
# Note that StartLimitInterval was renamed to StartLimitIntervalSec in systemd 230.
# Both the old, and new name are accepted by systemd 230 and up, so using the old name to make
# this option work for either version of systemd.
StartLimitInterval=60s

"/lib/systemd/system/docker.service" 47L, 1752B
```

You need to restart docker service to apply the config changes

```
sudo systemctl daemon-reload
sudo systemctl restart docker
sudo systemctl status docker
```

Expected output



The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal window title is "rps@rps-virtual-machine: ~". The terminal content displays the command-line session:

```
Sep 04 10:47:06 rps-virtual-machine systemd[1]: Started Docker Application Container Engine.
rps@rps-virtual-machine:~$ sudo systemctl-daemon reload
sudo: systemctl-daemon: command not found
rps@rps-virtual-machine:~$ sudo systemctl daemon-reload
rps@rps-virtual-machine:~$ sudo systemctl restart docker
rps@rps-virtual-machine:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2023-09-04 13:19:59 IST; 4s ago
    TriggeredBy: ● docker.socket
      Docs: https://docs.docker.com
      Main PID: 7369 (dockerd)
        Tasks: 15
       Memory: 24.7M
          CPU: 551ms
        CGroup: /system.slice/docker.service
                └─7369 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock -H tcp://0.0.0.0:4243

Sep 04 13:19:58 rps-virtual-machine dockerd[7369]: time="2023-09-04T13:19:58.924214686+05:30" level=info msg="[graphdriver] us"
Sep 04 13:19:58 rps-virtual-machine dockerd[7369]: time="2023-09-04T13:19:58.924490201+05:30" level=info msg="Loading container"
Sep 04 13:19:59 rps-virtual-machine dockerd[7369]: time="2023-09-04T13:19:59.434173901+05:30" level=info msg="Default bridge (>
Sep 04 13:19:59 rps-virtual-machine dockerd[7369]: time="2023-09-04T13:19:59.575407500+05:30" level=info msg="Loading container"
Sep 04 13:19:59 rps-virtual-machine dockerd[7369]: time="2023-09-04T13:19:59.598418433+05:30" level=warning msg="WARNING: API >
Sep 04 13:19:59 rps-virtual-machine dockerd[7369]: time="2023-09-04T13:19:59.598811802+05:30" level=info msg="Docker daemon" c>
Sep 04 13:19:59 rps-virtual-machine dockerd[7369]: time="2023-09-04T13:19:59.598997856+05:30" level=info msg="Daemon has compl>
Sep 04 13:19:59 rps-virtual-machine dockerd[7369]: time="2023-09-04T13:19:59.645782114+05:30" level=info msg="API listen on /r>
Sep 04 13:19:59 rps-virtual-machine systemd[1]: Started Docker Application Container Engine.
Sep 04 13:19:59 rps-virtual-machine dockerd[7369]: time="2023-09-04T13:19:59.645875283+05:30" level=info msg="API listen on [:>
rps@rps-virtual-machine:~$
```

What is Docker?

- is an application virtualization technology
- each container represents one application
- container doesn't OS Kernel
- container doesn't get its own dedicated Hardware resources
- each container has its own shell, port range and Network stack
- each container gets an IP address
- each container has a file system

Hypervisor vs Docker

- Virtual Machine represents an Operating System
- Container represents an application process
- Virtual machine gets its own hardware resources (CPU, RAM, Disk, etc)
- Containers running on the same system shares the Hardware resources on the underlying OS
- Container technology is an application virtualization technology
- container is light-weight virtualization technology
- Virtualization is heavy-weight - because each VM requires dedicated hardware resources
- Containers need Operating System
- containers are not a replacement for Virtualization (Virtual Machines)
- they both can be used together, they are complementing technology

Container Images

- specification of docker containers
- any software tools that you need on the containers must be installed on the Container image
- any number of containers can be creating using a Docker Image

Containers

- is a running instance of Docker Image
- containers get IP address, file system, network stack

Linux Kernel Features that enables the Container technology

1. Namespace - containers running on same machines are isolated by using namespace
2. Control Groups (CGroups)
 - we can apply resource quota restricts for each containers
 - we can restrict
 - how many CPU cores a container can use at the max
 - how much RAM a container can use at the max
 - how much storage a container can use at the max

How containers are different from normal application process

- containers also are normal application process, the only thing is it runs in a separate network namespace
-

Why Docker?

- high-level user-friendly software that makes things easier to manage images and containers
- internally docker depends on containerd
- containerd internally depends on container runtime called runC

What is a Container Engine?

- a user-friendly high-level software that uses other tools to manage container images/containers
- Examples
 - Docker
 - Podman

What is a Container Runtime?

- is a softwares that manages containers
 - create a container
 - list containers
 - delete containers
 - stop/start/restart/kill/abort containers
- container runtimes are low-level software utility which are not user-friendly, hence they are not normally used by end-users

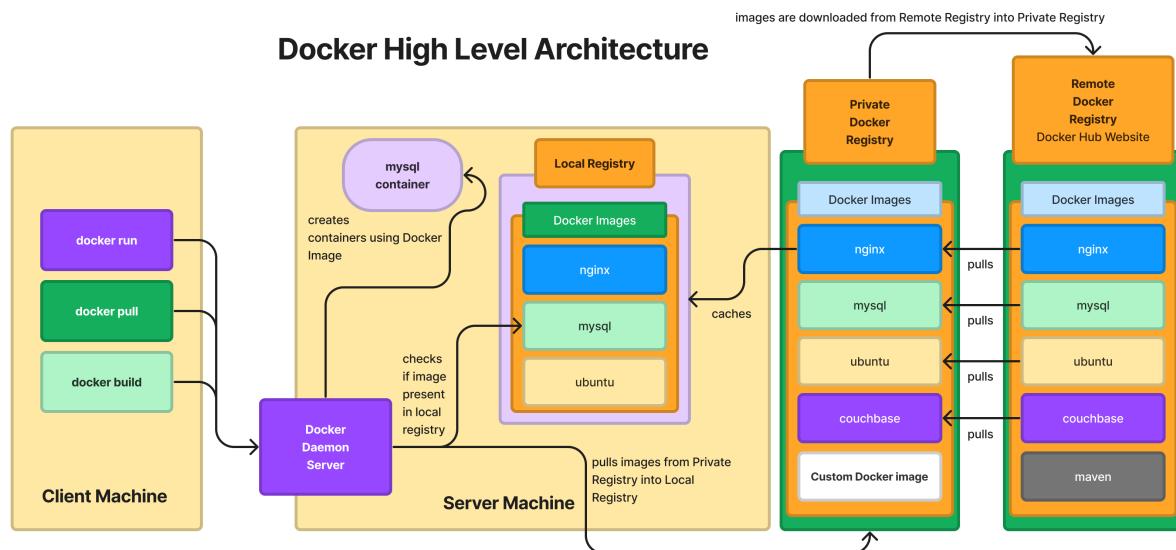
- these software are used by Container engines to manage containers Examples
- runc
- CRI-O

What type of applications can be containerized?

- Web Servers
- Application Servers
- DB Servers
- REST API
- SOAP API
- Microservices

Hypervisor High-Level Architecture

Docker High-Level Architecture



Docker Commands

Demo - Connecting your local Docker client to a remote Docker server

You need to do this on your local machine

```
export DOCKER_HOST=tcp://<ip-address-of-machine-where-docker-server-is-running>:4243
docker images
```

Lab - Finding docker version

```
docker --version
```

Expected output

```
└──(jegan@tektutor.org) - [~/ansible-sep-2023]
└─$ docker --version
Docker version 20.10.25+dfsg1, build b82b9f3
```

Lab - Listing docker images from Local Docker Registry

```
docker images
```

Expected output

```
└──(jegan@tektutor.org) - [~/ansible-sep-2023]
└─$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED
SIZE
tektutor/ansible-centos-node    latest    67e2c2ee6256  2 days ago
428MB
tektutor/ansible-ubuntu-node    latest    a3f39aea7460  2 days ago
220MB
centos                7.9.2009  eeb6ee3f44bd  23 months ago
204MB
ubuntu                16.04     b6f507652425  2 years ago
135MB
```

Lab - Docker Remote Registry

Open the below URL on your Ubuntu Cloud machine chrome web browser, click on Explore

```
https://hub.docker.com
```

Expected output

The screenshot shows the Docker Hub search results for 'hello-world'. The search bar at the top has 'Search Docker Hub' and a magnifying glass icon. Below it, there are tabs for 'Explore', 'Pricing', 'Sign In', and a 'Sign up' button. On the left, there's a sidebar titled 'Filters' with sections for 'Products' (Images, Extensions, Plugins), 'Trusted Content' (Docker Official Image, Verified Publisher, Sponsored OSS), 'Operating Systems' (Linux, Windows), 'Architectures' (ARM, ARM 64, IBM POWER, IBM Z, PowerPC 64 LE), and a 'Suggested' dropdown.

The main area displays 1 - 25 of 10,000 available results. The first four results are:

- alpine**: Docker Official Image · 1B+ · 10K+ · Updated 3 days ago. Description: A minimal Docker image based on Alpine Linux with a complete package index and only 5 ... Tags: Linux, IBM Z, riscv64, x86-64, ARM, ARM 64, 386, PowerPC 64 LE. Pulls: 9,146,575.
- nginx**: Docker Official Image · 1B+ · 10K+ · Updated 6 days ago. Description: Official build of Nginx. Tags: Linux, ARM, ARM 64, 386, mips64le, PowerPC 64 LE, IBM Z, x86-64. Pulls: 9,146,576.
- busybox**: Docker Official Image · 1B+ · 3.1K · Updated 2 months ago. Description: Busybox base image. Tags: Linux, riscv64, IBM Z, x86-64, ARM, ARM 64, 386, mips64le, PowerPC 64 LE. Pulls: 9,146,576.
- ubuntu**: Docker Official Image · 1B+ · 10K+ · Updated 2 days ago. Description: Ubuntu base image. Tags: Linux, riscv64, IBM Z, x86-64, ARM, ARM 64, 386, mips64le, PowerPC 64 LE. Pulls: 9,146,576.

At the bottom, there's a cookie consent banner with 'Accept All Cookies', 'Reject All', and 'Cookies Settings' buttons.

Lab - Downloading the hello-world docker image from Docker Hub Remote Registry to your local docker registry

```
docker pull hello-world:latest
```

Expected output

The screenshot shows a terminal session on a Linux system. The prompt is 'jegan@tektutor:~/ansible-sep-2023\$'. The user runs the command 'docker pull hello-world:latest'. The output shows the image being pulled from the library/hello-world repository. The digest and status are displayed, followed by the URL 'docker.io/library/hello-world:latest'.

```
(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
tektutor/ansible-centos-node    latest   67e2c2ee6256  2 days ago   428MB
tektutor/ansible-ubuntu-node    latest   a3f39aea7460  2 days ago   220MB
centos              7.9.2009  eeb6ee3f44bd  23 months ago  204MB
ubuntu              16.04    b6f507652425  2 years ago   135MB

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker pull hello-world:latest
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
tektutor/ansible-centos-node    latest   67e2c2ee6256  2 days ago   428MB
tektutor/ansible-ubuntu-node    latest   a3f39aea7460  2 days ago   220MB
hello-world          latest   9c7a54a9a43c  4 months ago  13.3kB
centos              7.9.2009  eeb6ee3f44bd  23 months ago  204MB
ubuntu              16.04    b6f507652425  2 years ago   135MB

(jegan@tektutor.org)-[~/ansible-sep-2023]
```

Lab - Deleting a docker image from Local Docker Registry

```
docker rmi hello-world:latest
```

Expected output

```
(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
tektutor/ansible-centos-node    latest   67e2c2ee6256  2 days ago   428MB
tektutor/ansible-ubuntu-node    latest   a3f39aea7460  2 days ago   220MB
hello-world           latest   9c7a54a9a43c  4 months ago  13.3kB
centos               7.9.2009 eeb6ee3f44bd  23 months ago  204MB
ubuntu               16.04    b6f507652425  2 years ago   135MB

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker rmi hello-world:latest
Untagged: hello-world:latest
Untagged: hello-world@sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c
Deleted: sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f9e016fdc3384500c2823d
Deleted: sha256:01bb4fce3eb1b56b05adf99504dafd31907a5aadac736e36b27595c8b92f07f1

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
tektutor/ansible-centos-node    latest   67e2c2ee6256  2 days ago   428MB
tektutor/ansible-ubuntu-node    latest   a3f39aea7460  2 days ago   220MB
centos               7.9.2009 eeb6ee3f44bd  23 months ago  204MB
ubuntu               16.04    b6f507652425  2 years ago   135MB

(jegan@tektutor.org)-[~/ansible-sep-2023]
```

Lab - Creating your first container

```
docker run hello-world:latest
```

Expected output

```
(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker run hello-world:latest
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c
Status: Downloaded newer image for hello-world:latest

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/get-started/
```

Lab - Listing the docker containers

List only running containers

```
docker ps
```

List all containers

```
docker ps -a
```

Expected output

```
jegan@tektutor: ~/ansible-sep-2023
jegan@tektutor: ~/ansible-sep-2023
jegan@tektutor: ~/ansible-sep-2023

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f001814f3174 hello-world:latest "/hello" 8 minutes ago Exited (0) 8 minutes ago
cranky_dewdney

(jegan@tektutor.org)-[~/ansible-sep-2023]
$
```

Lab - Creating container and run them in background

```
docker run -dit --name ubuntu1 --hostname ubuntu1 ubuntu:22.04 /bin/bash
docker run -dit --name ubuntu1 --hostname ubuntu1 ubuntu:22.04 /bin/bash
docker ps
docker ps -a
```

Let's understand the above run command

```
run - creates and start the container
dit - means deattached/backgorund and it stands for interactive terminal
```

Expected output

```
(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker run -dit --name ubuntu1 --hostname ubuntu1 ubuntu:22.04 /bin/bash
49eebeca9a9804579348be87f0d167b064e052b8c11cd18f991a9fae898f4da2

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker run -dit --name ubuntu2 --hostname ubuntu2 ubuntu:22.04 /bin/bash
80146e49a295a273372806478423fe8810e68aa21bf145cea17f143022d84618

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
80146e49a295 ubuntu:22.04 "/bin/bash" 3 seconds ago Up 2 seconds
49eebeca9a98 ubuntu:22.04 "/bin/bash" 44 seconds ago Up 42 seconds

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
80146e49a295 ubuntu:22.04 "/bin/bash" 28 seconds ago Up 27 seconds
49eebeca9a98 ubuntu:22.04 "/bin/bash" About a minute ago Up About a minute
f001814f3174 hello-world:latest "/hello" 21 minutes ago Exited (0) 21 minutes ago

(jegan@tektutor.org)-[~/ansible-sep-2023]
$
```

Lab - Getting inside a container shell

```
docker exec -it ubuntu1 /bin/bash
ls
hostname
hostname -i
exit
```

Expected output

```
(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker run -dit --name ubuntu1 --hostname ubuntu1 ubuntu:22.04 /bin/bash
49eebeca9a9804579348be87f0d167b064e052b8c11cd18f991a9fae898f4da2

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker run -dit --name ubuntu2 --hostname ubuntu2 ubuntu:22.04 /bin/bash
80146e49a295a273372806478423fe8810e68aa21bf145cea17f143022d84618

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
80146e49a295 ubuntu:22.04 "/bin/bash" 3 seconds ago Up 2 seconds
49eebeca9a98 ubuntu:22.04 "/bin/bash" 44 seconds ago Up 42 seconds

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
80146e49a295 ubuntu:22.04 "/bin/bash" 28 seconds ago Up 27 seconds
49eebeca9a98 ubuntu:22.04 "/bin/bash" About a minute ago Up About a minute
f001814f3174 hello-world:latest "/hello" 21 minutes ago Exited (0) 21 minutes ago

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker exec -it ubuntu1 /bin/bash
root@ubuntu1:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
root@ubuntu1:/# hostname
ubuntu1
root@ubuntu1:/# hostname -i
172.17.0.2
root@ubuntu1:/# exit
exit

(jegan@tektutor.org)-[~/ansible-sep-2023]
$
```

Lab - Creating a container in the foreground/interactive mode

```
docker run -it --name ubuntu3 --hostname ubuntu3 ubuntu:22.04 /bin/bash  
ls  
hostname  
hostname -i  
exit
```

Expected output

The screenshot shows a terminal window with two tabs. Both tabs have the title 'jegan@tektutor: ~/ansible-sep-2023'. The left tab contains the command history and output for creating a container:

```
(jegan@tektutor.org)-[~/ansible-sep-2023]  
$ docker run -it --name ubuntu3 --hostname ubuntu3 ubuntu:22.04 /bin/bash  
root@ubuntu3:/# ls  
bin dev home lib32 libx32 mnt proc run srv tmp var  
boot etc lib lib64 media opt root sbin sys usr  
root@ubuntu3:/# hostname  
ubuntu3  
root@ubuntu3:/# hostname -i  
172.17.0.4  
root@ubuntu3:/# exit  
exit
```

The right tab shows the output of the 'docker ps -a' command, listing all containers:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2b7c0642d249	ubuntu:22.04	"/bin/bash"	About a minute ago	Exited (0) 3 seconds ago		ubuntu3
80146e49a295	ubuntu:22.04	"/bin/bash"	11 minutes ago	Up 11 minutes		ubuntu2
49eebeca9a98	ubuntu:22.04	"/bin/bash"	11 minutes ago	Up 11 minutes		ubuntu1
f001814f3174	hello-world:latest	"/hello"	31 minutes ago	Exited (0) 31 minutes ago		cra
nky_dewdney						

At the bottom of the right tab, there is a prompt '\$'.

Lab - Finding more details about a container

```
docker inspect ubuntu1
```

Expected output

```
jegan@tektutor: ~/ansible-sep-2023
[jegan@tektutor: ~/ansible-sep-2023]
$ sudo ps aux | grep 18304
root      18304  0.0  0.0  4492  3880 pts/0    Ss+   15:34   0:00 /bin/bash
jegan     19941  0.0  0.0   6332  2204 pts/0    S+   15:53   0:00 grep --color=auto 18304

[jegan@tektutor: ~/ansible-sep-2023]
$ docker inspect ubuntu1
[
  {
    "Id": "49eebeca9a9804579348be87f0d167b064e052b8c11cd18f991a9fae898f4da2",
    "Created": "2023-09-04T10:04:09.946184655Z",
    "Path": "/bin/bash",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 18304,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-09-04T10:04:10.336425772Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:c6b84b685f35f1a5d63661f5d4aa662ad9b7ee4f4b8c394c022f25023c907b65",
    "ResolvConfPath": "/var/lib/docker/containers/49eebeca9a9804579348be87f0d167b064e052b8c11cd18f991a9fae898f4da2/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/49eebeca9a9804579348be87f0d167b064e052b8c11cd18f991a9fae898f4da2/hostname",
    "HostsPath": "/var/lib/docker/containers/49eebeca9a9804579348be87f0d167b064e052b8c11cd18f991a9fae898f4da2/hosts",
    "LogPath": "/var/lib/docker/containers/49eebeca9a9804579348be87f0d167b064e052b8c11cd18f991a9fae898f4da2/49eebeca9a9804579348be87f0d167b064e052b8c11cd18f991a9fae898f4da2-json.log",
    "Name": "/ubuntu1",
    "Platform": "linux"
  }
]
```

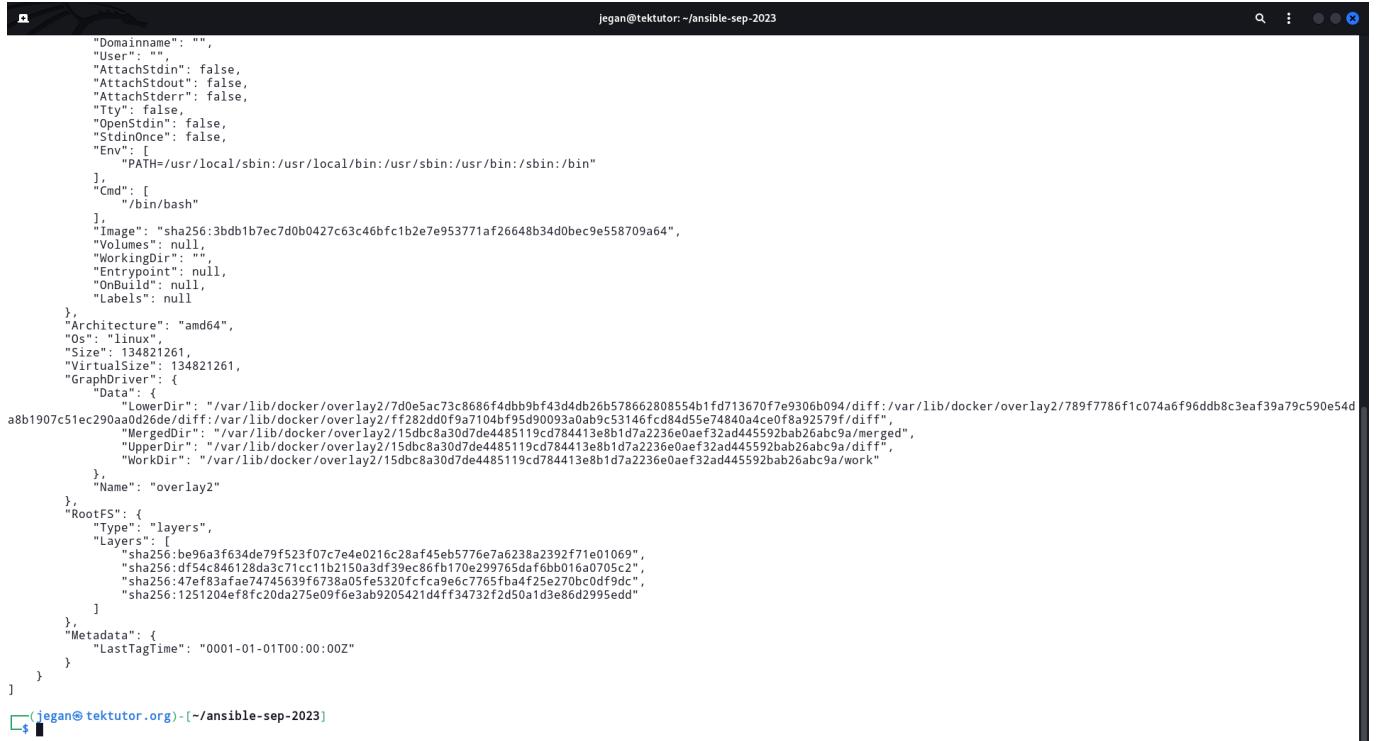
Lab - Finding details of Docker image

```
docker image inspect ubuntu:16.04
```

Expected output

```
jegan@tekttutor: ~/ansible-sep-2023
└─$ docker images
REPOSITORY          TAG      IMAGE ID   CREATED      SIZE
tekttutor/ansible-centos-node  latest   67e2c2ee6256  3 days ago  428MB
tekttutor/ansible-ubuntu-node  latest   a3f39ea7460   3 days ago  220MB
ubuntu              22.04   c6b846b85f35  2 weeks ago  77.8MB
hello-world         latest   9c7af54a9d45c  4 months ago  15.5kB
centos             7.9.2009  eebdee3ff4bd  23 months ago  204MB
ubuntu              16.04   b6f507652425  2 years ago  135MB

[jegan@tekttutor.org] - [~/ansible-sep-2023]
└─$ docker image inspect ubuntu:16.04
[{"Id": "sha256:b6f50765242581c887ff1acc2511fa2d885c52d8fb3ac8c4bba131fd86567f2e",
 "RepoTags": [
     "ubuntu:16.04"
 ],
 "RepoDigests": [
     "ubuntu@sha256:1f1a2d56de1d604801a9671f301190704c25d604a416f59e03c04f5c6ffee0d6"
 ],
 "Parent": "",
 "Comment": "",
 "Created": "2021-08-31T01:21:30.672229355Z",
 "Container": "02b9813c58908e4e449545066e8dfcff693ced2765493be69f64749f8b5ec70",
 "ContainerConfig": {
     "Hostname": "02b9813c5890",
     "Domainname": "",
     "User": "",
     "AttachStdin": false,
     "AttachStdout": false,
     "AttachStderr": false,
     "Tty": false,
     "OpenStdin": false,
     "StdinOnce": false,
     "Env": [
         "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
     ],
     "Cmd": [
         "/bin/sh",
         "-c",
         "#(nop) ",
         "CMD [\"/bin/bash\"]"
     ],
     "Image": "sha256:3bdb1b7ec7d0b0427c63c46bfc1b2e7e953771af26648b34d0bec9e558709a64",
     "Volumes": null,
     "WorkingDir": "",
     "Entrypoint": null,
     "OnBuild": null,
     "Labels": {}
 },
 "DockerVersion": "20.10.7",
 "Author": "",
 "Config": {}
```



```
jegan@tekktutor:~/ansible-sep-2023
{
  "Domainname": "",
  "User": "",
  "AttachStdin": false,
  "AttachStdout": false,
  "AttachStderr": false,
  "Tty": false,
  "OpenStdin": false,
  "StdinOnce": false,
  "Env": [
    "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  ],
  "Cmd": [
    "/bin/bash"
  ],
  "Image": "sha256:3bdb1b7ec7d0b0427c63c46bfc1b2e7e953771af26648b34d0bec9e558709a64",
  "Volumes": null,
  "WorkingDir": "",
  "Entrypoint": null,
  "OnBuild": null,
  "Labels": null
},
"Architecture": "amd64",
"Os": "linux",
"Size": 134821261,
"VirtualSize": 134821261,
"GraphDriver": {
  "Data": {
    "LowerDir": "/var/lib/docker/overlay2/7d0e5ac73c8686f4dbb9bf43d4db26b578662808554b1fd713670f7e9306b094/diff:/var/lib/docker/overlay2/789f7786f1c074a6f96ddb8c3eaf39a79c590e54d8b1907c51ec290aa0d26de/diff:/var/lib/docker/overlay2/ff282dd0f9a7104bf95d90093a0ab9c53146fc84d55e74840a4ce078a92579f/diff",
    "MergedDir": "/var/lib/docker/overlay2/15dbc8a30d7de4485119cd784413e8b1d7a2236e0aef32ad445592bab26abc9a/merged",
    "UpperDir": "/var/lib/docker/overlay2/15dbc8a30d7de4485119cd784413e8b1d7a2236e0aef32ad445592bab26abc9a/diff",
    "WorkDir": "/var/lib/docker/overlay2/15dbc8a30d7de4485119cd784413e8b1d7a2236e0aef32ad445592bab26abc9a/work"
  },
  "Name": "overlay2"
},
"RootFS": {
  "Type": "layers",
  "Layers": [
    "sha256:be96a3f634de79f523f07c7e4e0216c28af45eb5776e7a6238a2392f71e01069",
    "sha256:df54c846128da3c71cc11b2150a3df39ec86fb170e299765da66bb016a0705c2",
    "sha256:47ef83fae74745639f6738a05fe5320fcfc9e6c7765fba4f25e270bc0df9dc",
    "sha256:1251204ef8fc20da275e09f6e3ab9205421d4ff34732f2d50a1d3e86d2995edd"
  ]
},
"Metadata": {
  "LastTagTime": "0001-01-01T00:00:00Z"
}
}
]
(jegan@tekktutor.org) - [~/ansible-sep-2023]
```

Lab - Renaming a container

```
docker rename <current-container-name> <new-name>
docker ps
docker rename ubuntu2 c25a18dbd4922c    nginx:latest    "/docker-entrypoint.
..."    9 minutes ago      Up 1 second      0.0.0.0:8001->80/tcp,  :::8001->80/tcp
lb

docker ps
```

Expected output

The screenshot shows a terminal window with three tabs. The active tab is titled '(jegan@tektutor.org)-[~/ansible-sep-2023]'. It displays the output of the 'docker ps' command, listing two containers:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1a740bc666b9	ubuntu:16.04	"bash"	12 minutes ago	Up 12 minutes		ubuntu2
0837970b39b6	ubuntu:16.04	"bash"	13 minutes ago	Up 13 minutes		ubuntu1

Below this, another command is run: '\$ docker rename ubuntu2 c2'. A third tab is visible in the background, showing the root prompt on an Ubuntu system.

Lab - Deleting containers

```
docker ps
docker stop c2
docker rm c2

docker rm -f ubuntu1
docker ps
```

Expected output

```
jegan@tektutor:~/ansible-sep-2023
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1a740bc666b9 ubuntu:16.04 "bash" 12 minutes ago Up 12 minutes
0837970b39b6 ubuntu:16.04 "bash" 13 minutes ago Up 13 minutes
c2
ubuntu1

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker rm c2 ubuntu2
Error response from daemon: You cannot remove a running container 1a740bc666b9a4a42747525b959b620c531a3efd28db09e2e1476e1ddda0f6d91. Stop the container before attempting removal or force remove
Error: No such container: ubuntu2

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker stop c2
c2

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker rm c2
c2

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker rm -f ubuntu2
Error: No such container: ubuntu2

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker rm -f ubuntu1
ubuntu1

(jegan@tektutor.org)-[~/ansible-sep-2023]
$
```

Lab - Deleting multiple containers without calling out their names individually

```
docker ps
docker ps -q
docker ps -aq
docker rm -f $(docker ps -aq)
```

Expected output

```
jegan@tektutor:~/ansible-sep-2023
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
07cfea5df2ef ubuntu:16.04 "bash" 3 seconds ago Up 1 second
c21e1b384a84 ubuntu:16.04 "bash" 7 seconds ago Up 6 seconds
998fe3ecc6f2 ubuntu:16.04 "bash" 12 seconds ago Up 11 seconds
c3
c2
c1

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker ps -q
07cfea5df2ef
c21e1b384a84
998fe3ecc6f2

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker ps -aq
07cfea5df2ef
c21e1b384a84
998fe3ecc6f2

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker rm -f $(docker ps -aq)
07cfea5df2ef
c21e1b384a84
998fe3ecc6f2

(jegan@tektutor.org)-[~/ansible-sep-2023]
$
```

Lab - Setup a LoadBalancer with 3 nginx web servers - Port Forwarding

First let's create 3 web server containers using nginx:latest docker image

```
docker ps
docker run -d --name web1 --hostname web1 nginx:latest
docker run -d --name web2 --hostname web2 nginx:latest
docker run -d --name web3 --hostname web3 nginx:latest
docker ps
```

Expected output

```
jegan@tektutor.org - [~/ansible-sep-2023]
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

(jegan@tektutor.org) - [~/ansible-sep-2023]
$ docker run -d --name web1 --hostname web1 nginx:latest
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
52d2bf179e3: Pull complete
fd9f026c6310: Pull complete
055fa98b4363: Pull complete
96576293dd29: Pull complete
a7c4092be904: Pull complete
e3b6889c8954: Pull complete
da761d9a302b: Pull complete
Digest: sha256:104c7c5c54f2685f0f46f3be607ce60da7085da3eaa5ad22d3d9f01594295e9c
Status: Downloaded newer image for nginx:latest
ed0c44d9a3207c5a7de43ec6b6fdb30ae3e817305ec2f1bdb7d3fa91673c61c4

(jegan@tektutor.org) - [~/ansible-sep-2023]
$ docker run -d --name web2 --hostname web2 nginx:latest
a84f2edad6040c9cea6aad9953b7c135d8658a82e0384387122f518ddcb84cc6

(jegan@tektutor.org) - [~/ansible-sep-2023]
$ docker run -d --name web3 --hostname web3 nginx:latest
5b16ab857f0790ea74adce79faef88498b9b28b3deb668eeb93c8d04475ff97d

(jegan@tektutor.org) - [~/ansible-sep-2023]
$ 

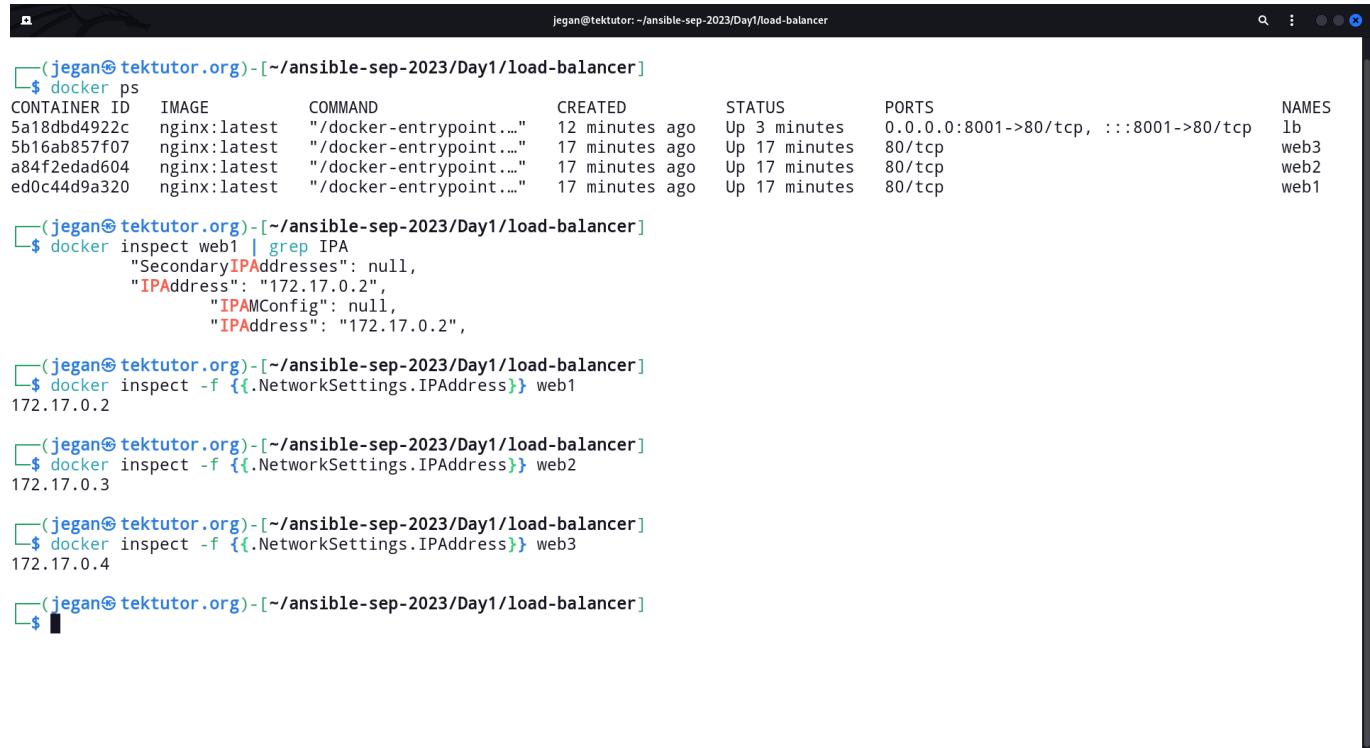
(jegan@tektutor.org) - [~/ansible-sep-2023]
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5b16ab857f07 nginx:latest "/docker-entrypoint...." 2 minutes ago Up 2 minutes 80/tcp web3
a84f2edad604 nginx:latest "/docker-entrypoint...." 2 minutes ago Up 2 minutes 80/tcp web2
ed0c44d9a320 nginx:latest "/docker-entrypoint...." 2 minutes ago Up 2 minutes 80/tcp web1

(jegan@tektutor.org) - [~/ansible-sep-2023]
$ 
```

Let's find the IP addresses of web1, web2 and web3

```
docker inspect web1 | grep IPA
docker inspect -f {{.NetworkSettings.IPAddress}} web1
docker inspect -f {{.NetworkSettings.IPAddress}} web2
docker inspect -f {{.NetworkSettings.IPAddress}} web3
```

Expected output



The screenshot shows a terminal window with the following session:

```
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5a18dbd4922c nginx:latest "/docker-entrypoint...." 12 minutes ago Up 3 minutes 0.0.0.0:8001->80/tcp, :::8001->80/tcp lb
5b16ab857f07 nginx:latest "/docker-entrypoint...." 17 minutes ago Up 17 minutes 80/tcp web3
a84f2edad604 nginx:latest "/docker-entrypoint...." 17 minutes ago Up 17 minutes 80/tcp web2
ed0c44d9a320 nginx:latest "/docker-entrypoint...." 17 minutes ago Up 17 minutes 80/tcp web1

(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker inspect web1 | grep IPA
"SecondaryIPAddresses": null,
"IPAddress": "172.17.0.2",
"IPAMConfig": null,
"IPAddress": "172.17.0.2",

(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker inspect -f {{.NetworkSettings.IPAddress}} web1
172.17.0.2

(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker inspect -f {{.NetworkSettings.IPAddress}} web2
172.17.0.3

(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker inspect -f {{.NetworkSettings.IPAddress}} web3
172.17.0.4

(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$
```

Let's create the load balancer container with port forward to make it accessible from other machines in the same network

```
docker run -d --name lb --hostname lb -p 8001:80 nginx:latest
```

Expected output

```
jegan@tektutor: ~/ansible-sep-2023
a84f2edad604    nginx:latest    "/docker-entrypoint..."    2 minutes ago    Up 2 minutes    80/tcp
web2
ed0c44d9a320    nginx:latest    "/docker-entrypoint..."    2 minutes ago    Up 2 minutes    80/tcp
web1

[jegan@tektutor.org]-[~/ansible-sep-2023]
$ docker run -d --name lb --hostname lb -p 8001:80 nginx:latest
5a18dbd4922c2d675f941b1742c049b7ad24025206778c343344c01bfd00c2fa

[jegan@tektutor.org]-[~/ansible-sep-2023]
$ docker ps
CONTAINER ID   IMAGE      COMMAND
NAMES
5a18dbd4922c   nginx:latest "/docker-entrypoint..."  3 seconds ago  Up 2 seconds  0.0.0.0:8
001->80/tcp,  :::8001->80/tcp
lb
5b16ab857f07   nginx:latest "/docker-entrypoint..."  4 minutes ago  Up 4 minutes  80/tcp
web3
a84f2edad604   nginx:latest "/docker-entrypoint..."  4 minutes ago  Up 4 minutes  80/tcp
web2
ed0c44d9a320   nginx:latest "/docker-entrypoint..."  5 minutes ago  Up 5 minutes  80/tcp
web1

[jegan@tektutor.org]-[~/ansible-sep-2023]
$
```

We need to configure the lb container to work like a load balancer, hence let's copy its config file to local machine to edit it and put it back inside.

```
cd ~/ansible-sep-2023/Day1/load-balancer
docker cp lb:/etc/nginx/nginx.conf .
ls
```

Expected output

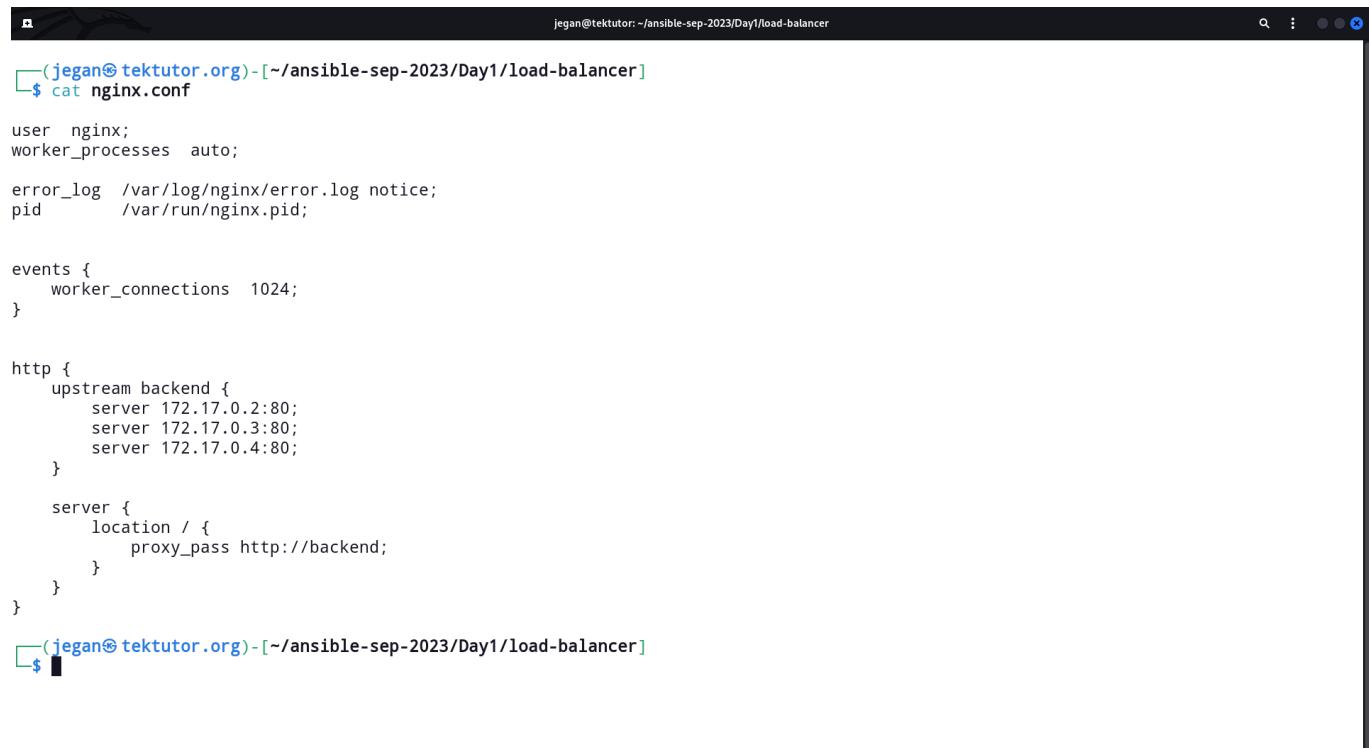
```
jegan@tektutor: ~/ansible-sep-2023/Day1/load-balancer
CONTAINER ID   IMAGE      COMMAND
NAMES
5a18dbd4922c   nginx:latest "/docker-entrypoint..."  2 minutes ago  Up 2 minutes  0.0.0.0:8001->80/tcp,  :::8001->80/tcp
lb
5b16ab857f07   nginx:latest "/docker-entrypoint..."  6 minutes ago  Up 6 minutes  80/tcp
web3
a84f2edad604   nginx:latest "/docker-entrypoint..."  6 minutes ago  Up 6 minutes  80/tcp
web2
ed0c44d9a320   nginx:latest "/docker-entrypoint..."  7 minutes ago  Up 7 minutes  80/tcp
web1

[jegan@tektutor.org]-[~/ansible-sep-2023]
$ docker exec -it lb sh
# ls
bin  dev          docker-entrypoint.sh  home  lib32  libx32  mnt  proc  run  srv  tmp  var
boot docker-entrypoint.d  etc            lib   lib64  media  opt  root  sbin  sys  usr
# cd /etc/nginx
# ls
conf.d  fastcgi_params  mime.types  modules  nginx.conf  scgi_params  uwsgi_params
# vi
sh: 4: vi: not found
# vim
sh: 5: vim: not found
# exit

[jegan@tektutor.org]-[~/ansible-sep-2023]
$ mkdir -p Day1/load-balancer
[jegan@tektutor.org]-[~/ansible-sep-2023]
$ cd Day1/load-balancer
[jegan@tektutor.org]-[~/ansible-sep-2023/Day1/load-balancer]
$ docker cp lb:/etc/nginx/nginx.conf .
[jegan@tektutor.org]-[~/ansible-sep-2023/Day1/load-balancer]
$ ls
nginx.conf

[jegan@tektutor.org]-[~/ansible-sep-2023/Day1/load-balancer]
$
```

We need to edit the nginx.conf as shown below updating your web1, web2 and web3 container IPs as shown below



jegan@tektutor.org: ~/ansible-sep-2023/Day1/load-balancer

```
[jegan@tektutor.org] $ cat nginx.conf

user    nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    upstream backend {
        server 172.17.0.2:80;
        server 172.17.0.3:80;
        server 172.17.0.4:80;
    }

    server {
        location / {
            proxy_pass http://backend;
        }
    }
}
```

[jegan@tektutor.org] \$

We need to copy the nginx.conf file from our lab machine to lb container and restart lb container to apply config changes

```
cd ~/ansible-sep-2023
git pull
cd Day1/load-balancer
docker cp nginx.conf lb:/etc/nginx/nginx.conf
docker restart lb
docker ps
```

Expected output

```
jegan@tektutor: ~/ansible-sep-2023/Day1/load-balancer
worker_connections 1024;
}

http {
    upstream backend {
        server 172.17.0.2:80;
        server 172.17.0.3:80;
        server 172.17.0.4:80;
    }

    server {
        location / {
            proxy_pass http://backend;
        }
    }
}

(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker cp nginx.conf lb:/etc/nginx/nginx.conf

(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker restart lb
lb

(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5a18dbd4922c nginx:latest "/docker-entrypoint..." 17 minutes ago Up 1 second 0.0.0.0:8001->80/tcp, :::8001->80/tcp lb
5b16ab857f07 nginx:latest "/docker-entrypoint..." 21 minutes ago Up 21 minutes 80/tcp web3
a84f2edad604 nginx:latest "/docker-entrypoint..." 21 minutes ago Up 21 minutes 80/tcp web2
ed0c44d9a320 nginx:latest "/docker-entrypoint..." 22 minutes ago Up 22 minutes 80/tcp web1

(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$
```

Let's create a custom html page and copy them to web1, web2 and web3 to differentiate which web server is responding to our request from lb

```
echo "Nginx Web Server1" > index.html
docker cp index.html web1:/usr/share/nginx/html/index.html

echo "Nginx Web Server2" > index.html
docker cp index.html web2:/usr/share/nginx/html/index.html

echo "Nginx Web Server3" > index.html
docker cp index.html web3:/usr/share/nginx/html/index.html
```

Expected output

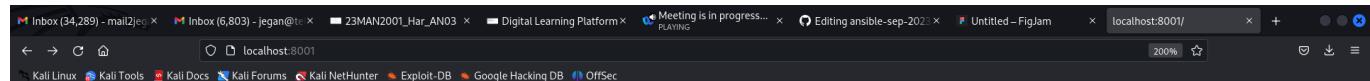
The terminal window shows the following session:

```
jegan@tektutor: ~/ansible-sep-2023/Day1/load-balancer
$ echo "Nginx Web server1" > index.html
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker cp index.html web1:/usr/share/nginx/html/index.html
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ echo "Nginx Web server2" > index.html
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker cp index.html web2:/usr/share/nginx/html/index.html
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ echo "Nginx Web server3" > index.html
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ docker cp index.html web3:/usr/share/nginx/html/index.html
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ curl http://172.17.0.2
Nginx Web server1
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ curl http://172.17.0.3
Nginx Web server2
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ curl http://172.17.0.4
Nginx Web server3
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$ curl http://localhost:8001
Nginx Web server1
(jegan@tektutor.org)-[~/ansible-sep-2023/Day1/load-balancer]
$
```

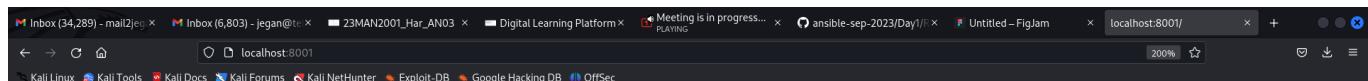
You should able to test the loadbalancer by access the below URL from ubuntu web browser

```
http://localhost:8001
http://localhost:8001
http://localhost:8001
```

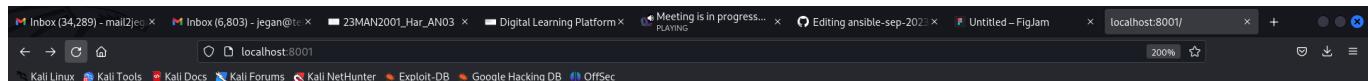
Expected output is each time you browse, it route the calls in round robin fashion



Nginx Web server1



Nginx Web server2



Nginx Web server3

Lab - Understanding Docker networking

Docker supports 3 types of network out of the box

```
docker network ls  
docker network inspect bridge
```

The screenshot shows two terminal windows side-by-side. Both windows have a title bar 'jegan@tektutor: ~/ansible-sep-2023'. The left terminal window displays the command '\$ docker network ls' followed by a table of network information:

NETWORK ID	NAME	DRIVER	SCOPE
dbc48ad61547	bridge	bridge	local
c92d71a403ed	host	host	local
81a86ceaa981	none	null	local

The right terminal window displays the command '\$ docker network inspect bridge' followed by a JSON object representing the bridge network configuration:

```
[{"Name": "bridge", "Id": "dbc48ad6154731145161cdaa0eeeb3c77e437b70ddf5c68eb702e714fee2a7bc", "Created": "2023-09-05T10:04:25.14020554+05:30", "Scope": "local", "Driver": "bridge", "EnableIPv6": false, "IPAM": {"Driver": "default", "Options": null, "Config": [{"Subnet": "172.17.0.0/16", "Gateway": "172.17.0.1"}]}, "Internal": false, "Attachable": false, "Ingress": false, "ConfigFrom": {"Network": ""}, "ConfigOnly": false, "Containers": {}, "Options": {"com.docker.network.bridge.default_bridge": "true", "com.docker.network.bridge.enable_icc": "true", "com.docker.network.bridge.enable_ip_masquerade": "true", "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0", "com.docker.network.bridge.name": "docker0", "com.docker.network.driver.mtu": "1500"}, "Labels": {}}]
```

Creating a custom network in docker

```
docker network ls
docker network create my-network-1 --subnet 200.10.20.0/24
docker network ls
```

Expected output

The screenshot shows a terminal window with three tabs. The current tab displays the output of running Docker commands to create a custom network and inspect it.

```
jegan@tektutor:~/ansible-sep-2023
(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
dbc48ad61547    bridge    bridge      local
c92d71a403ed    host      host       local
81a86ceaa981    none      null       local

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker network create my-network-1 --subnet 200.10.20.0/24
8c1807f922a811c302a44d19f129d1c76fe6944c03e3e4f86e6906af2f097531

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
dbc48ad61547    bridge    bridge      local
c92d71a403ed    host      host       local
8c1807f922a8    my-network-1  bridge      local
81a86ceaa981    none      null       local

(jegan@tektutor.org)-[~/ansible-sep-2023]
$ ifconfig
br-8c1807f922a8: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
      inet 200.10.20.1  netmask 255.255.255.0  broadcast 200.10.20.255
        ether 02:42:dc:f5:4b:6e  txqueuelen 0  (Ethernet)
      RX packets 0 bytes 0 (0.0 B)
      TX packets 0 bytes 0 (0.0 B)
```

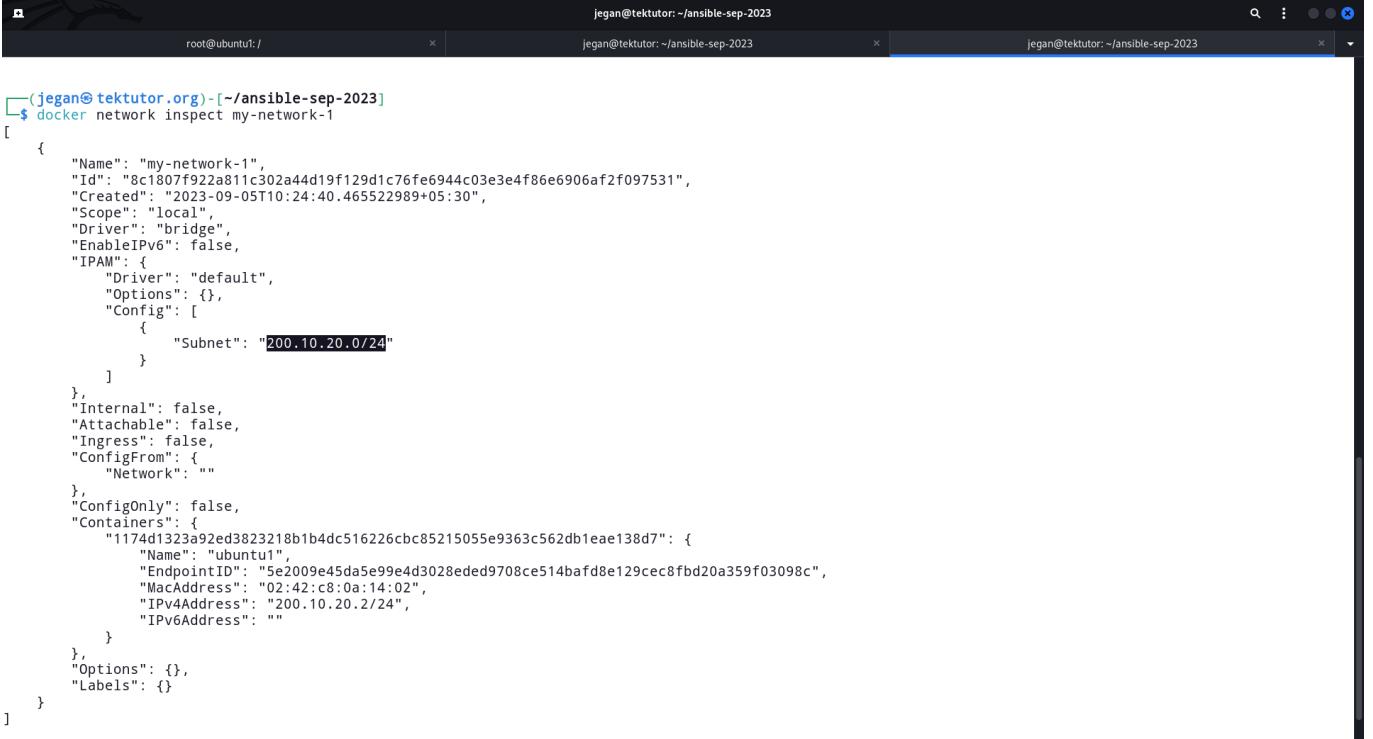
Let's create a new container and attach the new container to our custom network

```
docker run -dit --name ubuntu1 --hostname ubuntu1 --network=my-network-1
ubuntu:16.04 /bin/bash
```

Let's inspect the my-network-1 to see if the ubuntu1 container

```
docker network inspect my-network-1
```

Expected output



```
(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker network inspect my-network-1
[{"Name": "my-network-1", "Id": "8c1807f922a811c302a44d19f129d1c76fe6944c03e3e4f86e6906af2f097531", "Created": "2023-09-05T10:24:40.465522989+05:30", "Scope": "local", "Driver": "bridge", "EnableIPv6": false, "IPAM": {"Driver": "default", "Options": {}, "Config": [{"Subnet": "200.10.20.0/24"}]}, "Internal": false, "Attachable": false, "Ingress": false, "ConfigFrom": {"Network": ""}}, {"ConfigOnly": false, "Containers": {"1174d1323a92ed3823218b1b4dc516226cbc85215055e9363c562db1eae138d7": {"Name": "ubuntu1", "EndpointID": "5e2009e45da5e99e4d3028eded9708ce514baf8e129cec8fb20a359f03098c", "MacAddress": "02:42:c8:0a:14:02", "IPv4Address": "200.10.20.2/24", "IPv6Address": ""}}}, {"Options": {}, "Labels": {}}], }
```

Each time a new container is created, docker creates a pair of veth devices. One of the veth (virtual ethernet) stays in the local machine where docker container is running and the other veth device is used within the docker container. The veth device within container acts as the virtual network card/interface for the container i.e eth0, this helps the container communicate with the outside world, the other veth device in the local machine machine helps the local machine communicate with the container.

The veth supports only uni-directional communication, hence a pair of veth devices are created for every container.

You can check the veth on local machine as shown below

```
ifconfig
```

Expected output

```
jegan@tektutor:~/ansible-sep-2023
[jegan@tektutor.org] -[~/ansible-sep-2023]
$ ifconfig
br-59bbef864258: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255
        ether 02:42:6e:06:e9:f7 txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

br-8c1807f922a8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 200.10.20.1 netmask 255.255.255.0 broadcast 200.10.20.255
        inet6 fe80::42:dcff:fe5:4b6e prefixlen 64 scopeid 0x20<link>
            ether 02:42:dc:f5:4b:6e txqueuelen 0 (Ethernet)
            RX packets 4436 bytes 235030 (229.5 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 6697 bytes 27607456 (26.3 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        ether 02:42:4a:85:c2:9f txqueuelen 0 (Ethernet)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 0 bytes 0 (0.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b0:7b:25:12:de:5b txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device memory 0x90100000-9017ffff

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.108 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::b27b:25ff:fe12:df58 prefixlen 64 scopeid 0x20<link>
            ether b0:7b:25:12:df:58 txqueuelen 1000 (Ethernet)
            RX packets 143519 bytes 73025086 (69.6 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 117699 bytes 67728258 (64.5 MiB)

[jegan@tektutor.org] -[~/ansible-sep-2023]
root@ubuntu1:/
jegan@tektutor:~/ansible-sep-2023
jegan@tektutor:~/ansible-sep-2023
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b0:7b:25:12:de:5b txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device memory 0x90100000-9017ffff

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.108 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::b27b:25ff:fe12:df58 prefixlen 64 scopeid 0x20<link>
            ether b0:7b:25:12:df:58 txqueuelen 1000 (Ethernet)
            RX packets 143519 bytes 73025086 (69.6 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 117699 bytes 67728258 (64.5 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
            device interrupt 16 memory 0x90300000-90320000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 782 bytes 121879 (119.0 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 782 bytes 121879 (119.0 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vethdd2df63: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::1485:85ff:fe36:af3d prefixlen 64 scopeid 0x20<link>
        ether 16:85:85:36:4f:3d txqueuelen 0 (Ethernet)
        RX packets 4436 bytes 297134 (290.1 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 6710 bytes 27608462 (26.3 MiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[jegan@tektutor.org] -[~/ansible-sep-2023]
$
```

In order to check the veth inside the container, we need to get inside the container

```
docker exec -it ubuntu1 bash
```

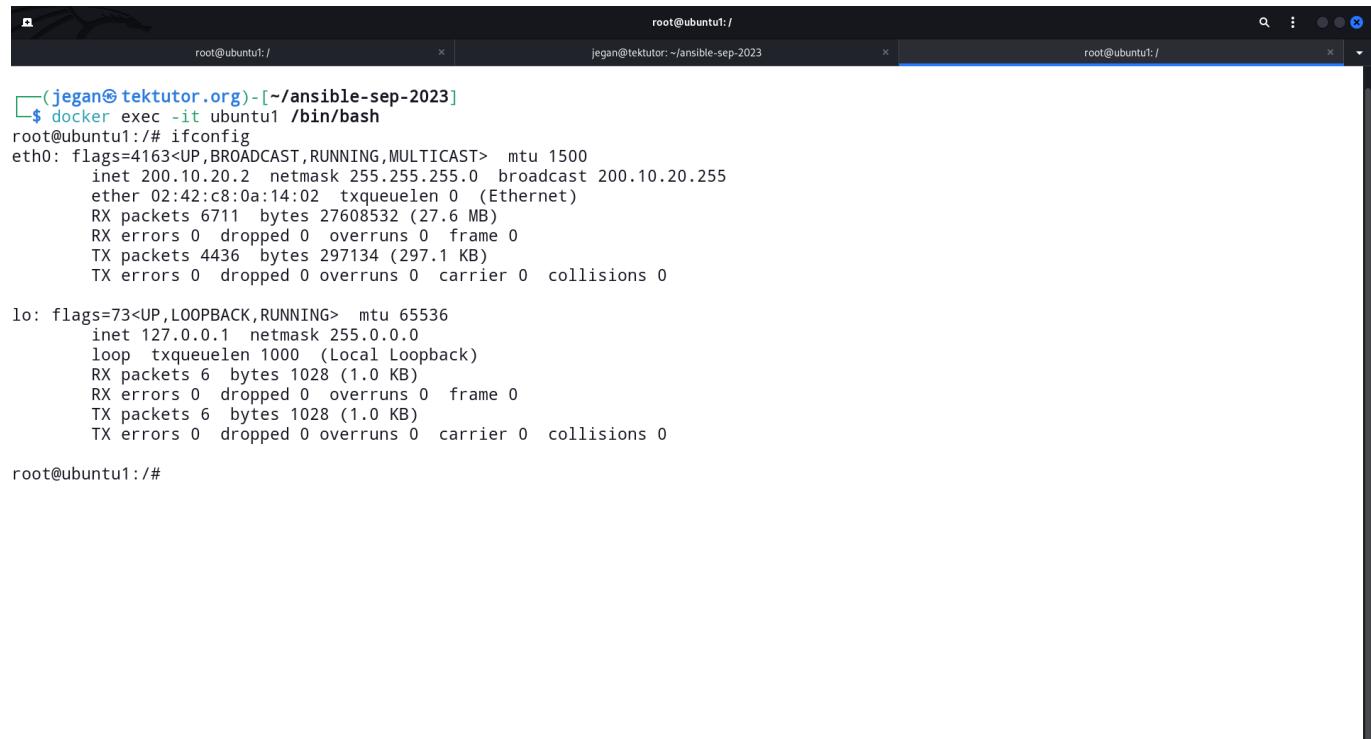
Let's install network tools within the container in order to issue ifconfig command

```
apt update && apt install -y net-tools
```

You may now check ifconfig in the container shell

```
ifconfig
```

Expected output



The screenshot shows a terminal window with three tabs. The active tab displays the output of the 'ifconfig' command in a root shell of a Docker container running Ubuntu 20.04. The output shows two interfaces: 'eth0' and 'lo'. 'eth0' is an Ethernet interface with an IP address of 200.10.20.255 and a broadcast address of 200.10.20.255. 'lo' is a loopback interface with an IP address of 127.0.0.1.

```
(jegan@tektutor.org)-[~/ansible-sep-2023]
$ docker exec -it ubuntu1 /bin/bash
root@ubuntu1:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 200.10.20.2  netmask 255.255.255.0  broadcast 200.10.20.255
        ether 02:42:c8:0a:14:02  txqueuelen 0  (Ethernet)
        RX packets 6711  bytes 27608532 (27.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4436  bytes 297134 (297.1 KB)
        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 6  bytes 1028 (1.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6  bytes 1028 (1.0 KB)
        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

root@ubuntu1:/#
```