🡪First create your **own** **network** and launch container.

🡪Docker is one tool of **Devops**.

🡪start of os known as boot.

🡪Install and boot os is known as **provisioning**.

🡪Here os = container.

🡪docker public registry- hub.docker.com

🡪It uses around **10-20 mb**.

🡪If docker command failed with OCI or permission denied error then stop SElinux security for some time.

🡪docker have **management** commands.

🡪docker containers are **isolated.**

**🡪**If you don’t have internet connectivity inside docker, choose your network (bridge), inside of default bridge.

**Docker use case**

1.sometime load come and one server may go down. we need to create new server as soon as possible. Docker launch (includes that server related files) & configure very fast.

2.when we have to create & destroy os many times in a day.

Install OS ----> boot OS ----> run program ----> shut down/terminate OS

**image**

1.Sometimes extra load come to site and if you have image, then you immediately create one more server.

**share image**

1.Export 🡪docker save webserver:v1 -o web.tar

Transfer🡪scp web.tar root@192.168.0.124:/root

Here, :/root -->path

Import 🡪docker load web.tar

2.put in docker repository

->first change image name according to docker rules.

->docker tag webserver:v1 akshit07/webserver:v1

->docker push akshit07/webserver:v1

🡪docker most off time come with **yum repo** files.

**start Docker commands.**

1.systemctl status docker

2.systemctl start docker

->temporary start docker

3.systemctl enable docker

4.Ctrl +p +q 🡪 to come outside container without exiting.

**Docker commands**

1.docker ps

->shows currently active container

(ps, ls list are alias).

2.docker ps -a

->shows all container (stopped also).

3.docker run -it ubuntu:14.04

-> -i for interact

-> -t for terminal

-> -d for run docker in bg.

-> run 🡪 install boot and login to new os.

🡪name akshit (Give name to container).

4.docker run -it ubuntu:14.04 date

->They run date command inside new os and give output and exit from it.

5.docker run -it --name z1 --rm ubuntu:14.04 date

->After executing terminate os.

6.docker start (name of os or id of os)

->start container which is in background.

7.docker attach (name of os or id of os)

->attach docker and go inside container.

8.docker stop (name of os or id of os)

->stop docker.

8.docker rm -f (name of os or id of os).

9.docker **commit** os1 myos:v1

->copy or clone and create new image.

10.docker **container** rm -f $(docker container ls -a -q)

11.exit / docker stop (name of os or id of os).

12.docker container inspect web

->docker container inspect --format "{{ .NetworkSettings.IPAddress }}" web

13.docker info

14.docker version

15.docker pull ubuntu:14.04(tag or version).

16.docker logs myos.

->Show last output of container

🡪 -f ->fetch output in redhat continues.

17.docker exec myos date

🡪Go inside container, do task, give output, come out of container.

**Script** hang-up or failed.

**Script** command shouldn’t be interactive.

🡪 Don’t go inside docker

->docker attach

->passwd

18. docker exec -it myos bash

🡪Go inside container (bash shell)

🡪it thinks he performing bash program, does not go inside os.

🡪if u do exit here, it exits bash shell. You returned to host os, but container running in background.

**Why systemctl command don’t work inside docker?**

->Every process has one **parent** id.  
->In redhat when u **boot**, first process run in known as **systemd**.  
->Contain doesn’t boot, so they don’t have systemd command.

->so, in container systemctl command don’t work.

**Networking**

->route -n

🡪**dhcp** provides ip to system.

->it connected with switch, when system connect to switch, threw dhcp it gets ipaddress.

->it gives dns also.

->IPAM

->One type of database stored info about network.

->**private** ip are not routable.

**DNS** server

->Convert name (URL) into ip address.

->Cat /etc/resolv.conf  
 ->gives ip of dns server.

**LAN:** local area network

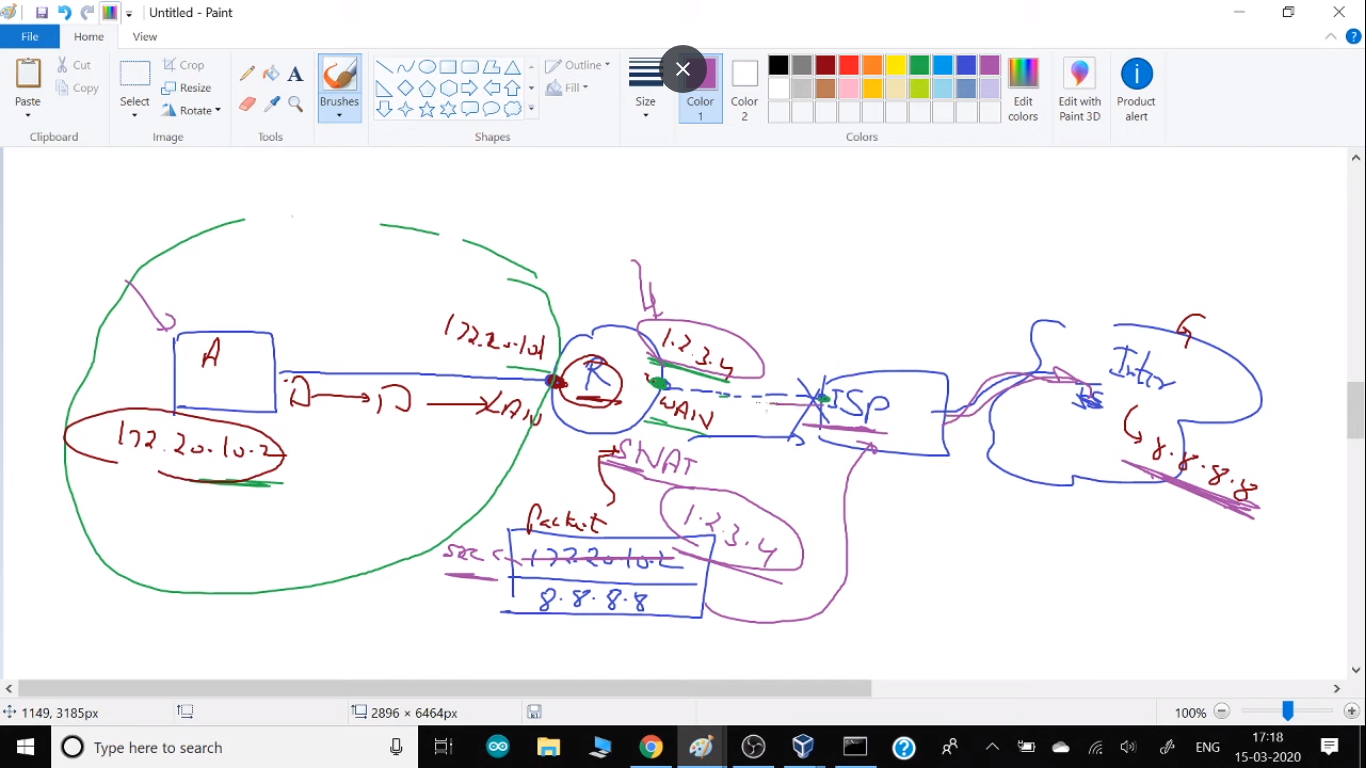
**WAN:** wide area network

**IP**1.public

2.private

->public can connect to public ---chargeable

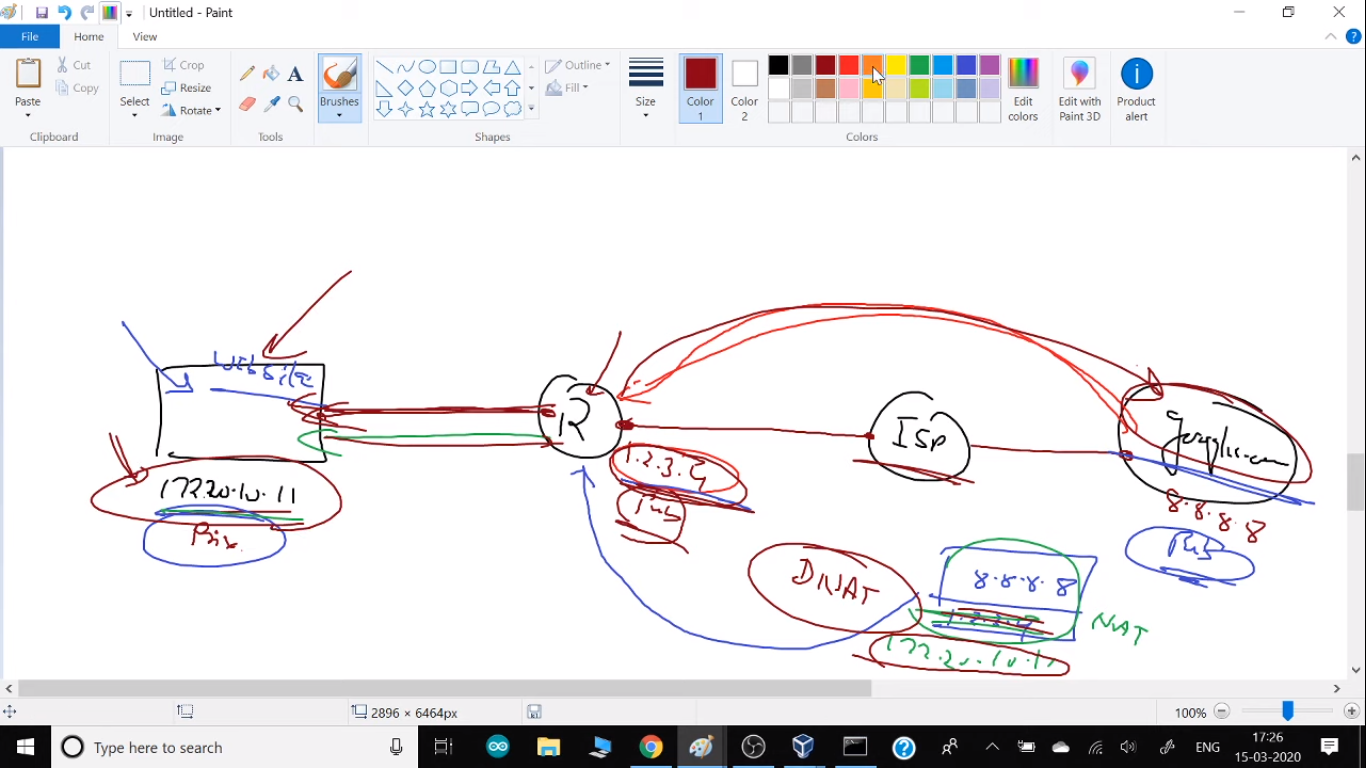
->private can connect to private ---not chargeable



🡪**when you want to connect with google.** ->First **packet** from your pc transmitted. It contains source ip and destination IP.

->Your IP private, but when it goes to through router, router change source IP to router IP (Gateway). This replacement is known as **SNAT**. Source Network Address Transmission

->After this it goes to ISP.  
 ->ISP feels, it is public so he passes it and it goes to google.



->**DNAT** – destination NAT

->It supports only one container to be connected. So, we use PAT.

🡪**PAT**: Port Address Translation

->It requires firewall enabled.

->NATing done by firewall…. inside iptables.

-> docker run -it -p 1234:80 -- name web2 -- network webnet webserver:v1 ->-p for PATing.

->a rule creates in Iptables (Iptables -nvL)  
 ->iptables -p FORWARD ACCEPT

🡪container **exposed.**

🡪Router public IP = redhat main IP (enp0s3)  
 🡪Router private IP = docker 0 IP  
🡪 IP forwarding: When your packet from one network card go to another.  
 ->To check ip forwarding active or not.  
 ->sysctl -a | grep ip\_forward

🡪Gateway=router address

🡪yum **whatprovies** nslookup

->nameserver

🡪nslookup [www.google.com](http://www.google.com)

->gives ip of url.

->host google.com

🡪172.17.0.0/16

->16 = 255.255.0.0 ->network name.

🡪docker network **inspect** bridge

->In this network they created dhcp server.

->switch use to connect system with same network.

->physically known as **switch**.

->software known as **bridge**.

->to connect **switch-router** & **enable** SNAT bridge driver used.

->For connecting two different networks, we require **router**.

->docker network **create** - -network **bridge** mynet

->Create your own network.

->**none** network doesn’t have net-card.

-> u don’t have internet connectivity.

->**Host** network also share networking things.

->In Host network cannot able to ping.

->docker network **disconnect** mynet myos

->docker network connect mynet2 myos

🡪docker run -it - - name web1 - - **network-alias** site - - network webnet phpweb:v1

🡪docker run -it - - name web2 - - **network-alias** site - - network webnet phpweb:v1

->now you can connect with nickname-site

->link in is part of your personal network (LAN).

**Network Isolation- Multi-tenant**

->Containers from same network are able to ping with one-another.  
-> Containers from different network are not able to ping with one-another.  
->It provides isolation.

->Provides security.

->Container of same network can ping each other with name.

->for this u need to connect with your own network.

🡪docker run -it - -link myos - -name myos2 ubuntu:14.04

->by this you can connect with name with default bridge.

**Alias**

**->**to provide nickname of command… alias ddel =” main command”  
 ->aliad ddel = ’ docker rm -f $(docker ps -a -q)’

**Storage:**

->temporary storage  
->permanent storage  
 1.persistent 2.Empheral

->docker **volume** create mystg  
->**Mount /** **Bind:** attach external storage (Pen drive) to docker

->docker run -it –name web1 -v mystg :/var/www/html webserver:v1

**Local** **YUM**  
->docker run -it –name web1 -v /run/media/root/RHEL-8-0-0-BaseOS-\*86\_64:/dvd webserver:v1  
 ->configure yum

**Kernel**

🡪uname -r

->first program when boot is kernel.

->When os boot, it scans complete hardware.

->drivers are part of kernel.

->kernel will take complete control on hardware

**Virtualization**

->It gives only some part of ram and kernel feels this much resource is available.

->Our windows approx. 5% resources use. 95% wastage.  
 ->And it can not be share with another os (bare-metal).

->Here, we can use many vm parallel but it also wastes resources.

->we also have to wait.

->we are sharing our hardware.

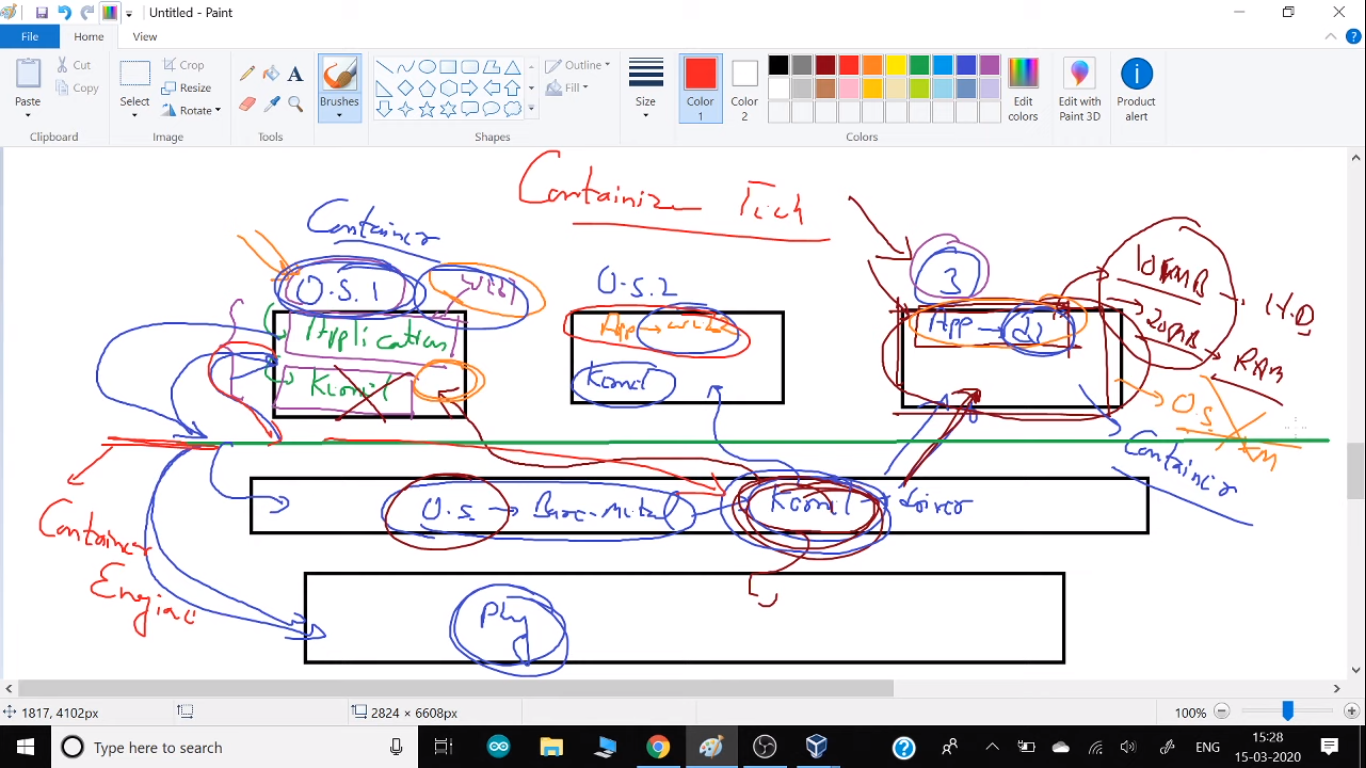
🡪10-20mb

**Containerization**

🡪host os shares his kernel, drivers and some other stuff.

🡪we are sharing our hardware and kernel and some other stuff too.

🡪light-weight-virtualization (many people say)



**configure webserver in docker:** -

🡪create you web pages inside

--cd /var/www/html/

🡪**XXX**----systemctl **don’t** work inside docker---**XXX**

🡪so, we hv to go with another option

🡪systemctl start httpd....

->behind the seen it use /usr/sbin/httpd

->but systemctl do some another thing.

🡪systemctl issue

->when you type systemctl stop it & kill process id. also remove Pid file.

->but if you use killall httpd it does not remove pid.so bash think process is already runing.so first remove file, then start services.

->we have to write this command in bash shell.

->rm -rf /var/run/httpd/\*

->/usr/sbin/httpd  
  
  
**Project:**  
🡪To pass environment variable use -e at runtime.

-> docker run -it -e z=$x centos  
 -> docker run -it -e name=akshit centos

🡪docker run -it -e MYSQL\_ROOT\_PASSWORD=redhat -e MYSQL\_USER=vimal

-e MYSQL\_PASSWORD=redhat -e MYSQL\_DATABASE=mydb mysql:5.7

->mysql server must require password at launch time.

->This is Entry point concept.

->They run script at launching container.

->yum install mysql

This use as a client.

🡪mysql -h 172.17.0.2 -u root -predhat

-> -h for host

-> -u for user

-> -p for password (There is no space).

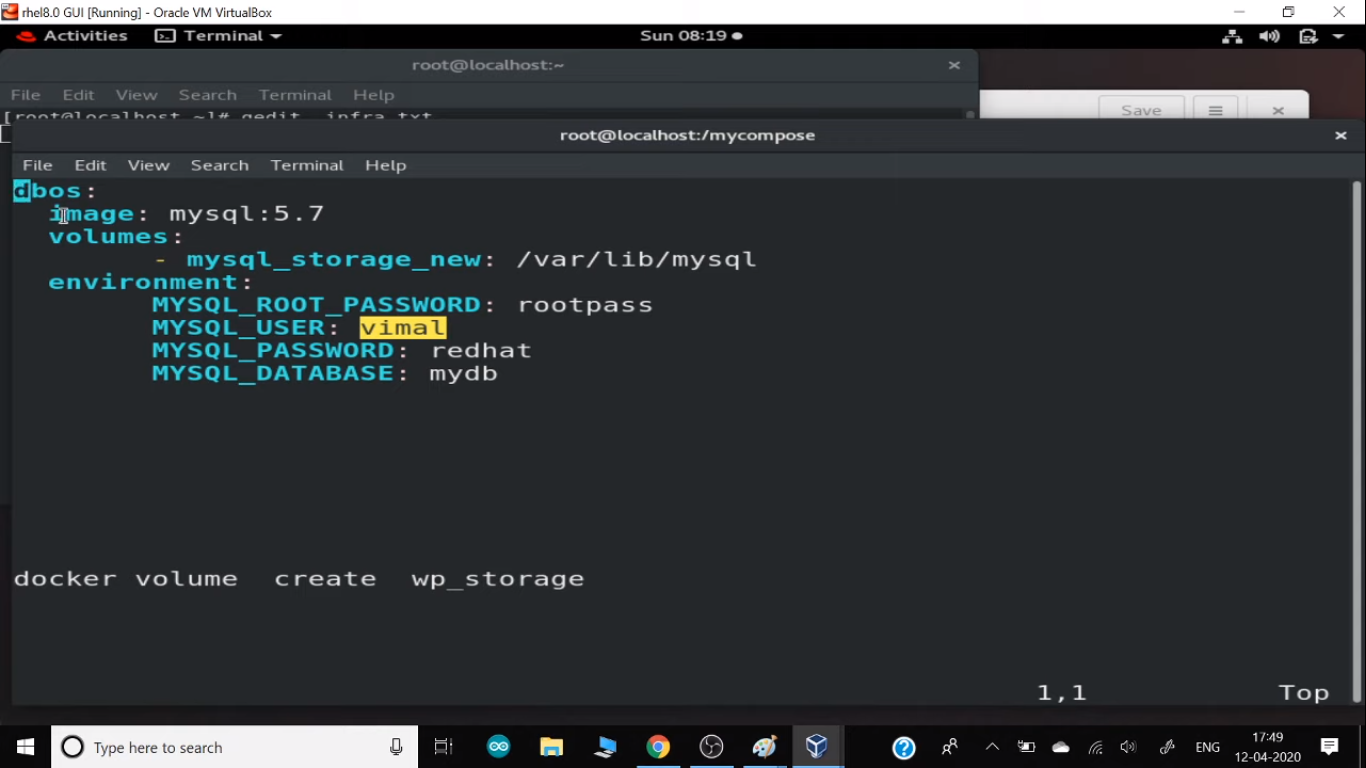
->show database; 🡪for show database.

->exit;

🡪docker run -it -e WORDPRESS\_DB\_HOST=db\_os -e WORDPRESSS\_DB\_USER=vimal -e WORDPRESSS\_DB \_PASSWORD=redhat -e WORDPRESSS\_DB \_NAME=mydb -v wp\_storage:/var/www/html --name dbos --link db\_os mysql:5.7

🡪Link will create a dns for us.

🡪db will connect to database.



**YML** **language**:

->name: “Akshit”

Key-value.

->phone: 1245

->variable-value.

🡪to write multiple value.  
->email:

- “hi”

- “hello”

🡪for block of code

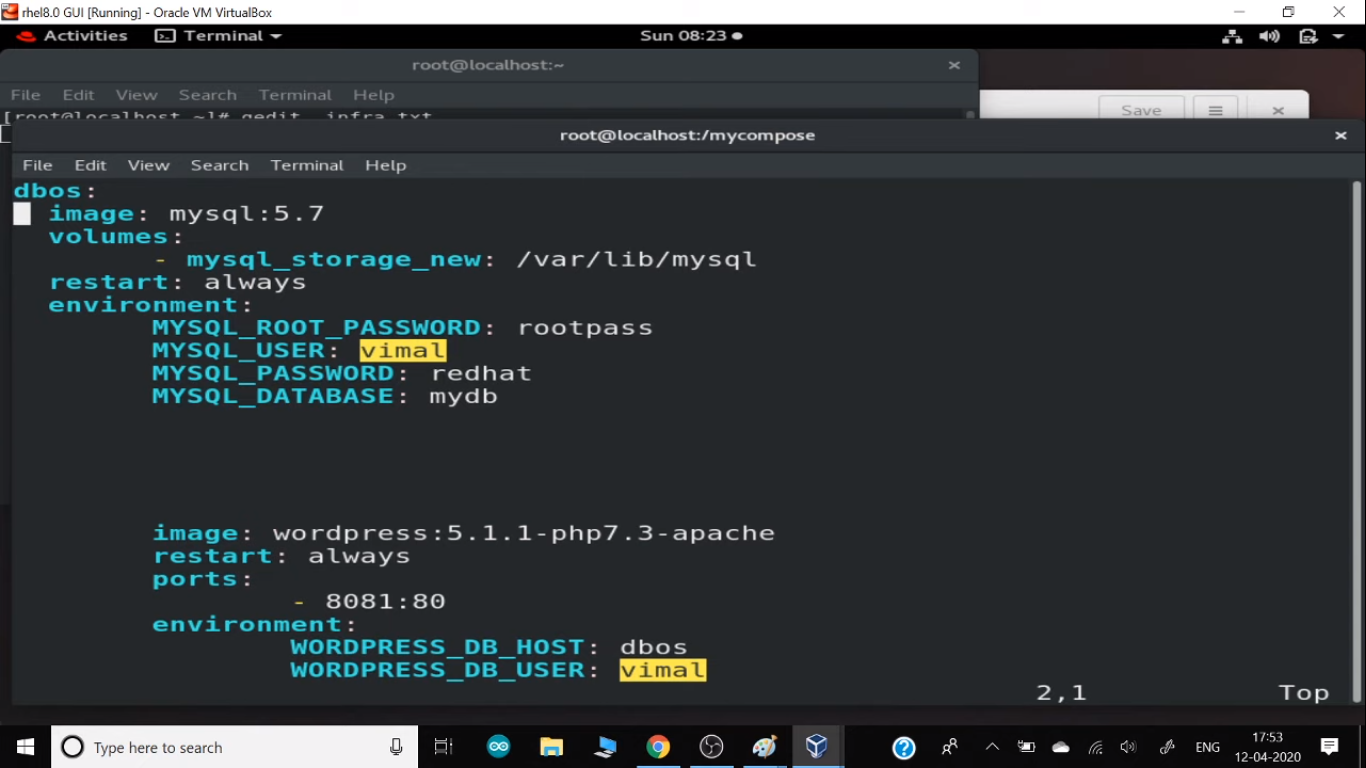
->name:

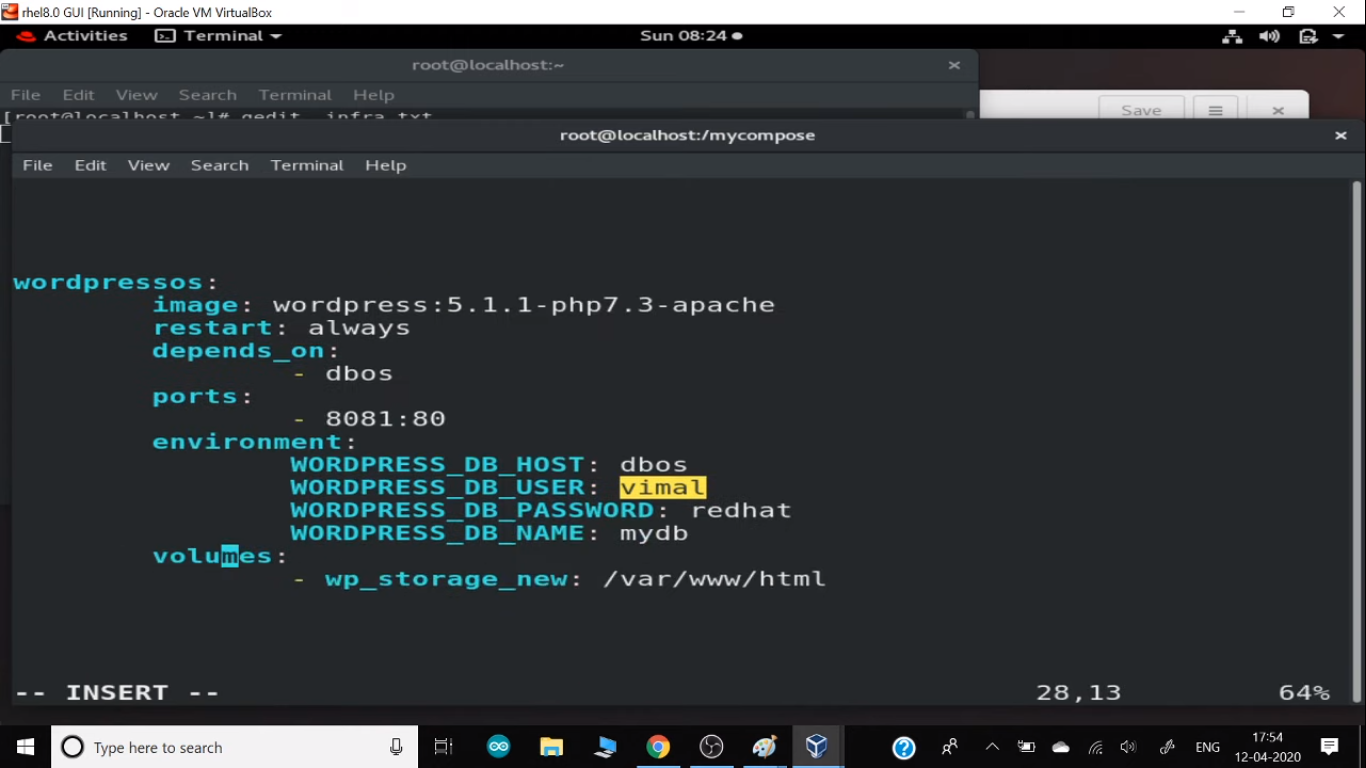
Email: --(two space)

- “hi”

- “hello”

🡪First create your own network.  
🡪use your own image.  
🡪vim /root/.bashrc



0

**Docker Compose**  
<https://docs.docker.com/compose/install/>

# curl -L "https://github.com/docker/compose/releases/download/1.25.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# chmod +x /usr/local/bin/docker-compose

* docker compose up