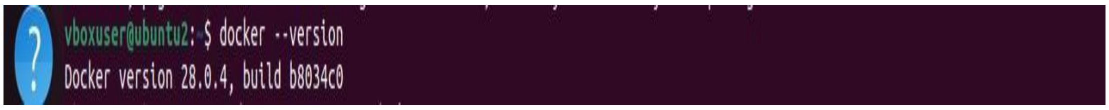
EXPERIMENT-8

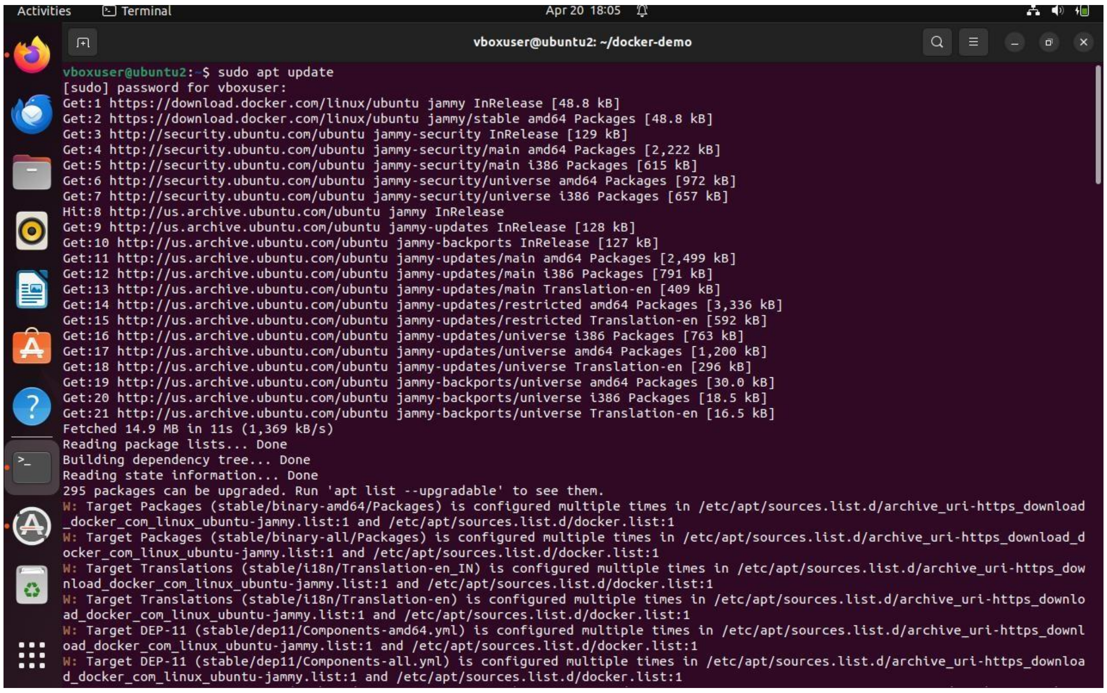
DOCKER COMPOSE

1) Verify Docker



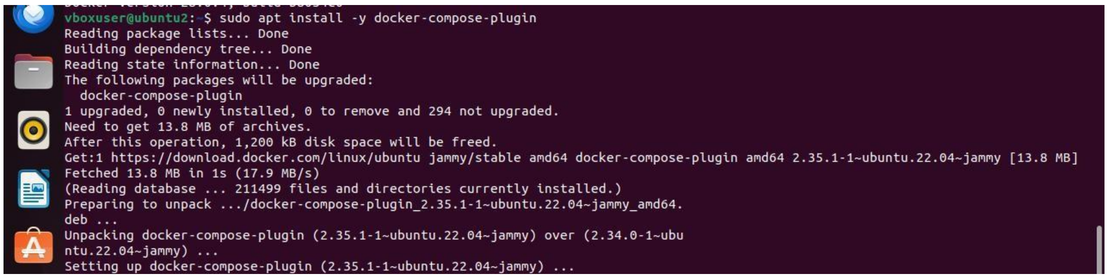
```
vboxuser@ubuntu2:~$ docker --version
Docker version 28.0.4, build b8034c0
```

2) Update Docker



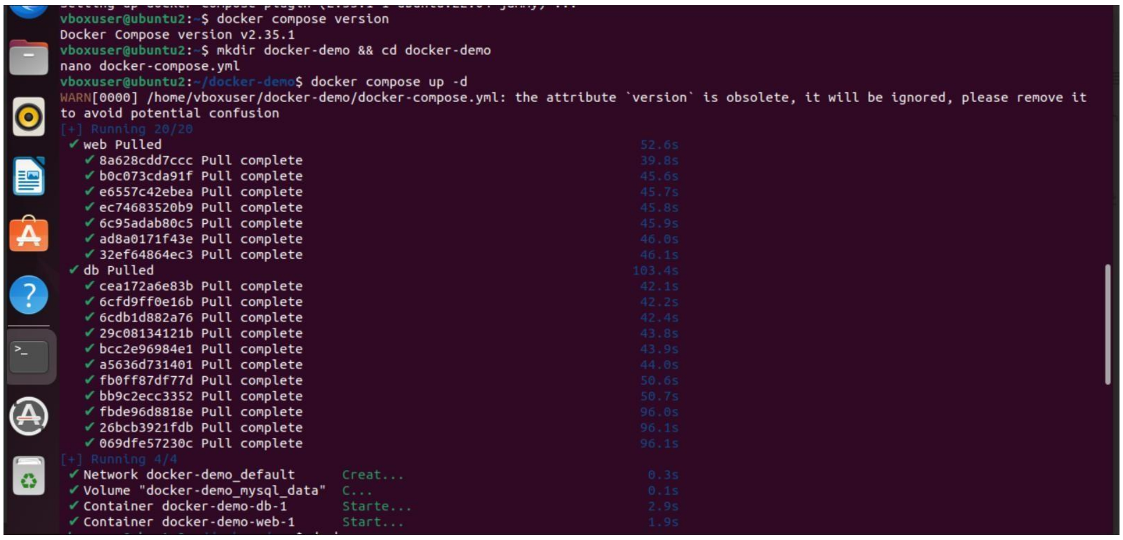
```
vboxuser@ubuntu2:~$ sudo apt update
[sudo] password for vboxuser:
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [48.8 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2,222 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [615 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [972 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/universe i386 Packages [657 kB]
Hit:8 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:9 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,499 kB]
Get:12 http://us.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [791 kB]
Get:13 http://us.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [409 kB]
Get:14 http://us.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [3,336 kB]
Get:15 http://us.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [592 kB]
Get:16 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [763 kB]
Get:17 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,200 kB]
Get:18 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [296 kB]
Get:19 http://us.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [30.0 kB]
Get:20 http://us.archive.ubuntu.com/ubuntu jammy-backports/universe i386 Packages [18.5 kB]
Get:21 http://us.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.5 kB]
Fetched 14.9 MB in 11s (1,369 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
295 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: Target Packages (stable/binary-amd64/Packages) is configured multiple times in /etc/apt/sources.list.d/archive_uri-https_download
_docker_com_linux_ubuntu-jammy.list:1 and /etc/apt/sources.list.d/docker.list:1
W: Target Packages (stable/binary-all/Packages) is configured multiple times in /etc/apt/sources.list.d/archive_uri-https_download_d
ocker_com_linux_ubuntu-jammy.list:1 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en_IN) is configured multiple times in /etc/apt/sources.list.d/archive_uri-https_dow
nload_docker_com_linux_ubuntu-jammy.list:1 and /etc/apt/sources.list.d/docker.list:1
W: Target Translations (stable/i18n/Translation-en) is configured multiple times in /etc/apt/sources.list.d/archive_uri-https_downlo
ad_docker_com_linux_ubuntu-jammy.list:1 and /etc/apt/sources.list.d/docker.list:1
W: Target DEP-11 (stable/dep11/Components-amd64.yml) is configured multiple times in /etc/apt/sources.list.d/archive_uri-https_downl
oad_docker_com_linux_ubuntu-jammy.list:1 and /etc/apt/sources.list.d/docker.list:1
W: Target DEP-11 (stable/dep11/Components-all.yml) is configured multiple times in /etc/apt/sources.list.d/archive_uri-https_downloa
d_docker_com_linux_ubuntu-jammy.list:1 and /etc/apt/sources.list.d/docker.list:1
```

3) Install Plugin



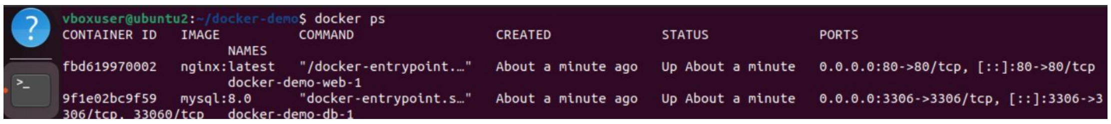
```
vboxuser@ubuntu2:~$ sudo apt install -y docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  docker-compose-plugin
1 upgraded, 0 newly installed, 0 to remove and 294 not upgraded.
Need to get 13.8 MB of archives.
After this operation, 1,200 kB disk space will be freed.
Get:1 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-compose-plugin amd64 2.35.1-1~ubuntu.22.04~jammy [13.8 MB]
Fetched 13.8 MB in 1s (17.9 MB/s)
(Reading database ... 211499 files and directories currently installed.)
Preparing to unpack .../docker-compose-plugin_2.35.1-1~ubuntu.22.04~jammy_amd64.
deb ...
Unpacking docker-compose-plugin (2.35.1-1~ubuntu.22.04~jammy) over (2.34.0-1~ubu
ntu.22.04~jammy) ...
Setting up docker-compose-plugin (2.35.1-1~ubuntu.22.04~jammy) ...
```

4) Create Docker Compose File and Checking Compose Version



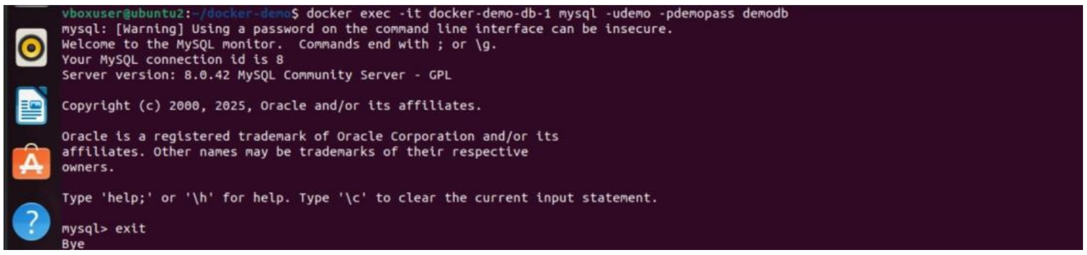
```
vboxuser@ubuntu2:~$ docker compose version
Docker Compose version v2.35.1
vboxuser@ubuntu2:~$ mkdir docker-demo && cd docker-demo
nano docker-compose.yml
vboxuser@ubuntu2:~/docker-demo$ docker compose up -d
WARN[0000] /home/vboxuser/docker-demo/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it
to avoid potential confusion
[+] Running 20/20
 ✔ web Pulled                                                                  52.6s
   ✔ 8a628cdd7ccc Pull complete                                                39.8s
   ✔ b0c073cda91f Pull complete                                                45.6s
   ✔ e6557c42ebea Pull complete                                                45.7s
   ✔ ec74683520b9 Pull complete                                                45.8s
   ✔ 6c95adab80c5 Pull complete                                                45.9s
   ✔ ad8a0171f43e Pull complete                                                46.0s
   ✔ 32ef64864ec3 Pull complete                                                46.1s
 ✔ db Pulled                                                                  103.4s
   ✔ cea172a6e83b Pull complete                                                42.1s
   ✔ 6cfd9ff0e16b Pull complete                                                42.2s
   ✔ 6cdb1d882a76 Pull complete                                                42.4s
   ✔ 29c08134121b Pull complete                                                43.8s
   ✔ bcc2e96984e1 Pull complete                                                43.9s
   ✔ a5636d731401 Pull complete                                                44.0s
   ✔ fb0ff87df77d Pull complete                                                50.6s
   ✔ bb9c2ecc3352 Pull complete                                                50.7s
   ✔ fbde96d8818e Pull complete                                                96.0s
   ✔ 26bcb3921fdb Pull complete                                                96.1s
   ✔ 069dfe57230c Pull complete                                                96.1s
[+] Running 4/4
 ✔ Network docker-demo_default      Creat...                                    0.3s
 ✔ Volume "docker-demo_mysql_data"  C...                                        0.1s
 ✔ Container docker-demo-db-1       Starte...                                   2.9s
 ✔ Container docker-demo-web-1      Start...                                    1.9s
```

5)Check running containers



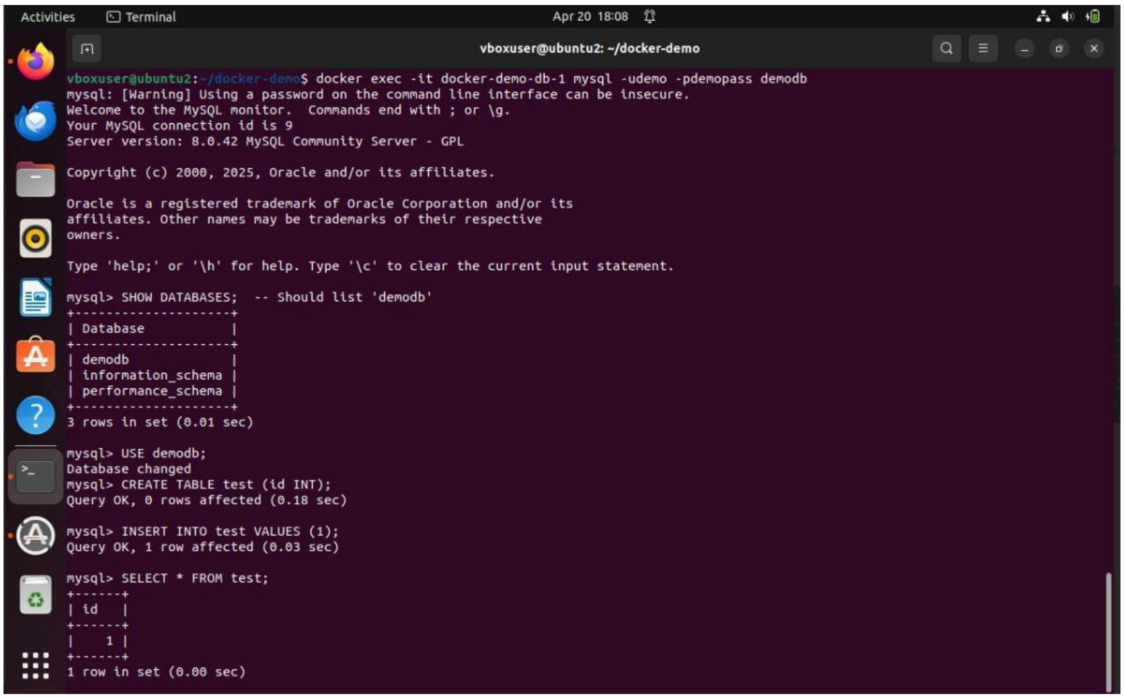
```
vboxuser@ubuntu2:~/docker-demo$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS
                     NAMES
fbd619970002   nginx:latest   "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:80->80/tcp, [::]:80->80/tcp
                     docker-demo-web-1
9f1e02bc9f59   mysql:8.0      "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3306->3306/tcp, [::]:3306->3
306/tcp, 33060/tcp   docker-demo-db-1
```

5) Verify Service are Running



```
vboxuser@ubuntu2:~/docker-demo$ docker exec -it docker-demo-db-1 mysql -udemo -pdemopass demodb
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: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

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> exit
Bye
```

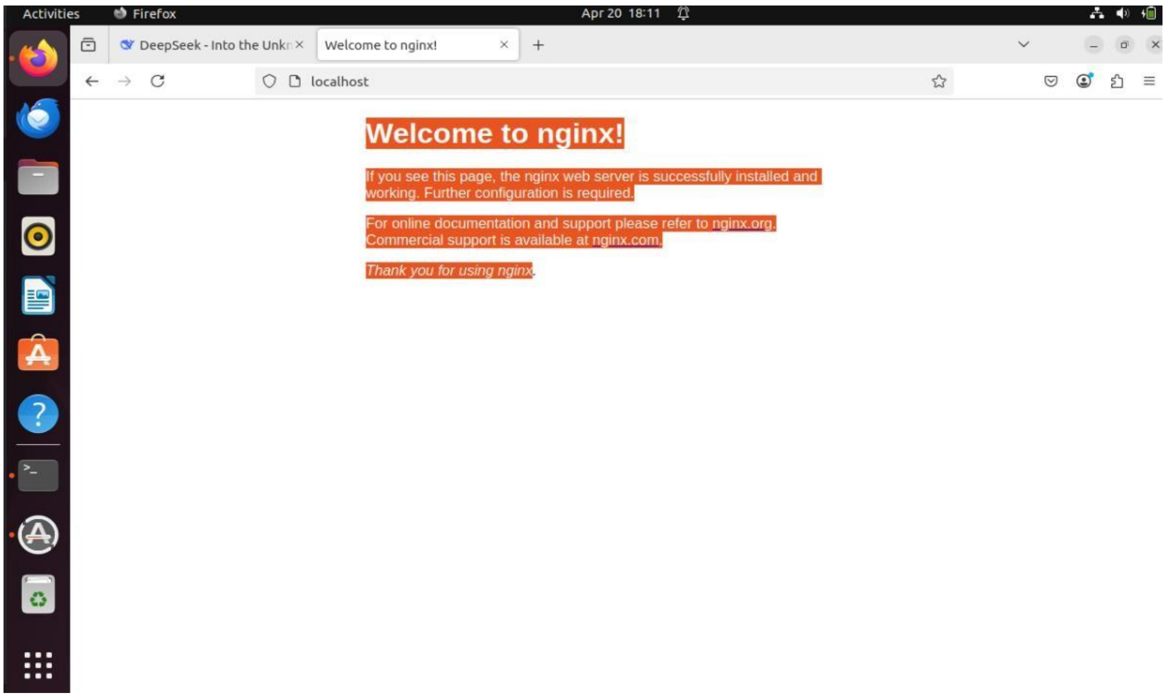
6) MySql Testing



```
vboxuser@ubuntu2:~/docker-demo$ docker exec -it docker-demo-db-1 mysql -udemo -pdemopass demodb
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: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

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;  -- Should list 'demodb'
+--------------------+
| Database           |
+--------------------+
| demodb             |
| information_schema |
| performance_schema |
+--------------------+
3 rows in set (0.01 sec)

mysql> USE demodb;
Database changed
mysql> CREATE TABLE test (id INT);
Query OK, 0 rows affected (0.18 sec)

mysql> INSERT INTO test VALUES (1);
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM test;
+------+
| id   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
```

7) Verify Nginx Service

URL:http://localhost



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

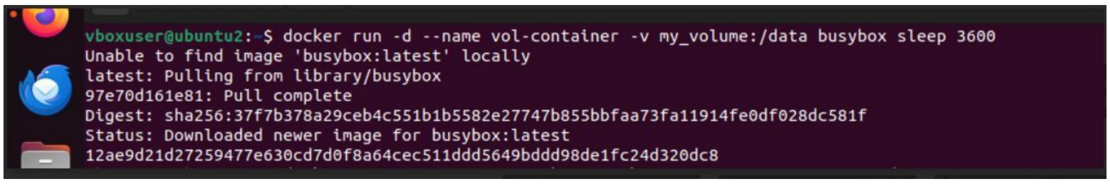
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

EXPERIMENT-9

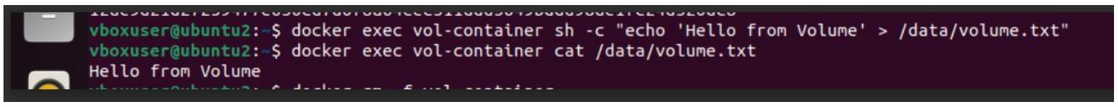
Docker Storage Experiment: Using Volume, Bind Mount, and tmpfs

1) create a container witth volume



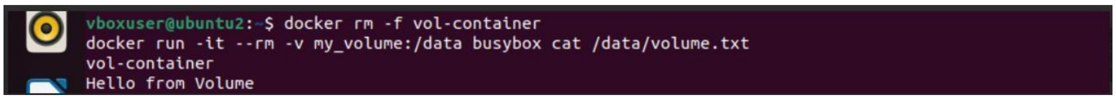
```
vboxuser@ubuntu2:~$ docker run -d --name vol-container -v my_volume:/data busybox sleep 3600
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
97e70d161e81: Pull complete
Digest: sha256:37f7b378a29ceb4c551b1b5582e27747b855bbfaa73fa11914fe0df028dc581f
Status: Downloaded newer image for busybox:latest
12ae9d21d27259477e630cd7d0f8a64cec511ddd5649bddd98de1fc24d320dc8
```

2) adding file and checking wheather it exist or not



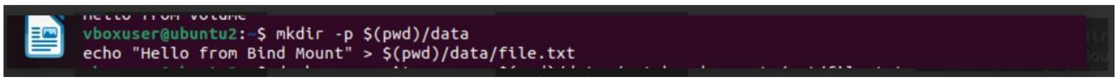
```
vboxuser@ubuntu2:~$ docker exec vol-container sh -c "echo 'Hello from Volume' > /data/volume.txt"
vboxuser@ubuntu2:~$ docker exec vol-container cat /data/volume.txt
Hello from Volume
```

3) removing container and reusing volume



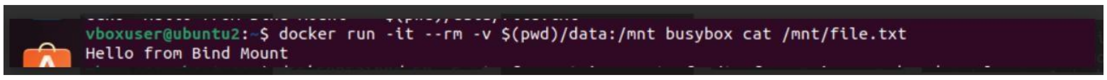
```
vboxuser@ubuntu2:~$ docker rm -f vol-container
docker run -it --rm -v my_volume:/data busybox cat /data/volume.txt
vol-container
Hello from Volume
```

4) creating a local directory and file



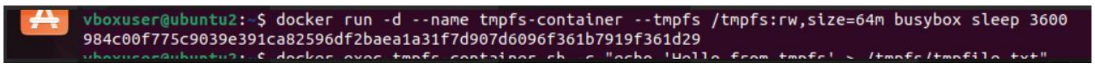
```
vboxuser@ubuntu2:~$ mkdir -p $(pwd)/data
echo "Hello from Bind Mount" > $(pwd)/data/file.txt
```

5) running contgainer with bind mount



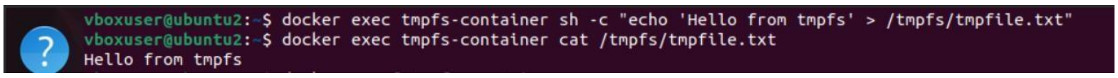
```
vboxuser@ubuntu2:~$ docker run -it --rm -v $(pwd)/data:/mnt busybox cat /mnt/file.txt
Hello from Bind Mount
```

6) running container with tmpfs mount



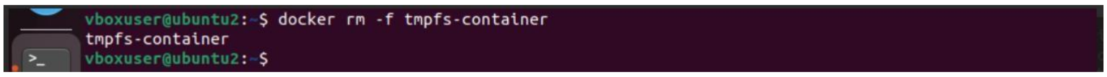
```
vboxuser@ubuntu2:~$ docker run -d --name tmpfs-container --tmpfs /tmpfs:rw,size=64m busybox sleep 3600
984c00f775c9039e391ca82596df2baea1a31f7d907d6096f361b7919f361d29
```

7) adding file and checking content



```
vboxuser@ubuntu2:~$ docker exec tmpfs-container sh -c "echo 'Hello from tmpfs' > /tmpfs/tmpfile.txt"
vboxuser@ubuntu2:~$ docker exec tmpfs-container cat /tmpfs/tmpfile.txt
Hello from tmpfs
```

8) removing container and verify data is lost



```
vboxuser@ubuntu2:~$ docker rm -f tmpfs-container
tmpfs-container
vboxuser@ubuntu2:~$
```

EXPERIMENT-10

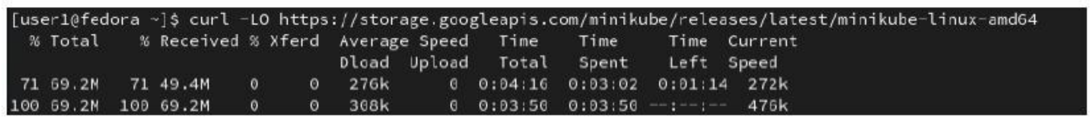
MiniKube Tutorial Installing & Starting in this MiniKube Tutorial

Before jumping into Kubernetes, you'll first have to install MiniKube on your machine. Later, you'll use MiniKube to transform your machine into a single node Kubernetes cluster.

Unlike other packages, you'll have to download the MiniKube binary with the curl command before installing MiniKube.

1. Run the curl command below to download a copy of the MiniKube binary on your machine (minikube-linux-amd64).

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linuxamd64



```
[user1@fedora ~]$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
 71 69.2M   71 49.4M    0     0   276k      0  0:04:16  0:03:02  0:01:14  272k
100 69.2M  100 69.2M    0     0   308k      0  0:03:50  0:03:50 --:--:--  476k
```

Downloading MiniKube

2. In the same directory where the binary was downloaded, run the following command to install the MiniKube binary (minikube-linux-amd64) to the appropriate location (/usr/local/bin/minikube).

Installing MiniKube doesn't provide output, but you'll be asked for your sudo password.

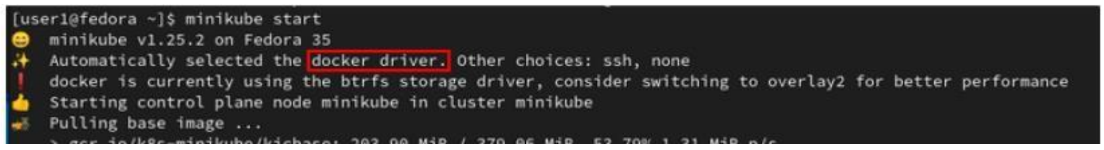
sudo install minikube-linux-amd64 /usr/local/bin/minikube

3. Lastly, run the below command to start MiniKube using the installed binary.

minikube start

If all goes well, you'll get an output similar to the one below. If Docker is your container manager, Docker has been automatically selected as the driver.

At this point, Kubernetes is now running perfectly.



```
[user1@fedora ~]$ minikube start
😄  minikube v1.25.2 on Fedora 35
✨  Automatically selected the docker driver. Other choices: ssh, none
❗  docker is currently using the btrfs storage driver, consider switching to overlay2 for better performance
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
```

Starting the MiniKube Tutorial

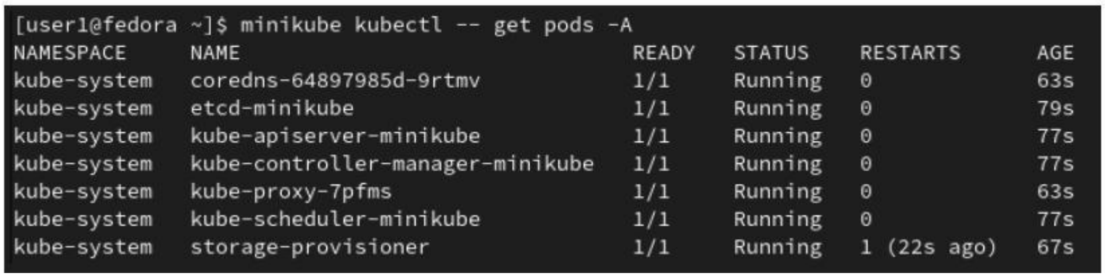
Setting up kubectl to Interact with MiniKube

When you start MiniKube for the first time, the Kubernetes client (kubectl) is automatically installed. But how do you let kubectl interact with MiniKube? By running the alias command to shorten the minikube kubectl command.

Perhaps you're more into clicking on a GUI. If so, you can set up the Kubernetes dashboard instead.

1. Run the following command to get all pods running in all namespaces (-A) in the cluster.

Below, you can see the minikube kubectl works fine, but it's a bit lengthy, and creating an alias would be a big help (step two).



```
[user1@fedora ~]$ minikube kubectl -- get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-64897985d-9rtmv            1/1     Running   0             63s
kube-system   etcd-minikube                      1/1     Running   0             79s
kube-system   kube-apiserver-minikube            1/1     Running   0             77s
kube-system   kube-controller-manager-minikube   1/1     Running   0             77s
kube-system   kube-proxy-7pfms                   1/1     Running   0             63s
kube-system   kube-scheduler-minikube            1/1     Running   0             77s
kube-system   storage-provisioner                1/1     Running   1 (22s ago)   67s
```

Listing All Pods Running in the Cluster

2. Next, run the following command to create an alias for the minikube kubectl command to save you a few keystrokes.

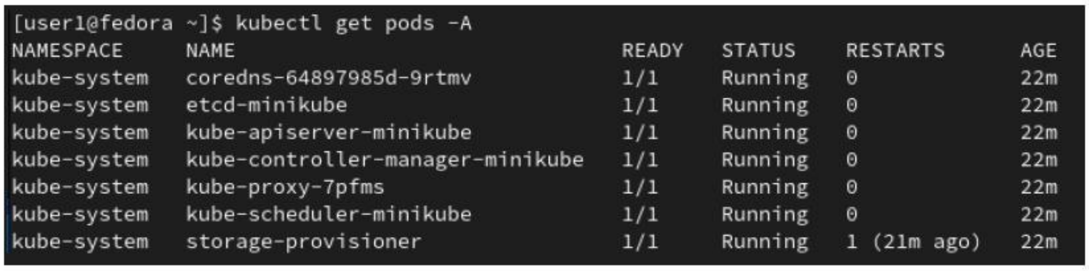
At this point, and throughout this tutorial, you'll only need to run the kubectl command as you interact with the cluster (step three).

alias kubectl="minikube kubectl --"

3. Finally, run the kubectl command below to get all pods running in all namespaces (-A) in the cluster, as you did in step one. This time, you start with just the kubectl command instead of minikube kubectl.

kubectl get pods -A

Notice below that you get the same output as step one using only the kubectl command.



```
[user1@fedora ~]$ kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-64897985d-9rtmv            1/1     Running   0             22m
kube-system   etcd-minikube                      1/1     Running   0             22m
kube-system   kube-apiserver-minikube            1/1     Running   0             22m
kube-system   kube-controller-manager-minikube   1/1     Running   0             22m
kube-system   kube-proxy-7pfms                   1/1     Running   0             22m
kube-system   kube-scheduler-minikube            1/1     Running   0             22m
kube-system   storage-provisioner                1/1     Running   1 (21m ago)   22m
```

Listing All Running Pods Using Shorthand kubectl Command

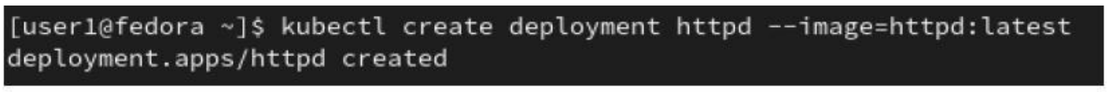
Deploying a Webserver Application to Kubernetes on MiniKube

By now, your cluster is ready for its first pod, so you'll deploy an Apache web server container to the cluster. You'll create a deployment based on an httpd image for this tutorial.

1. Run the following kubectl command to create a deployment named httpd using the latest image (httpd:latest).

kubectl create deployment httpd --image=httpd:latest

You can see below that the deployment has now been created.



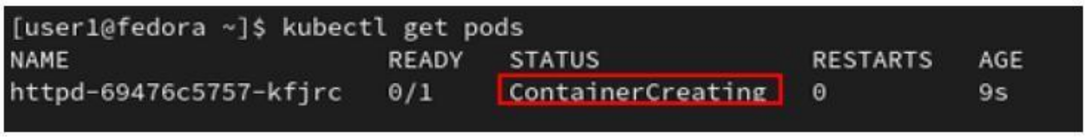
```
[user1@fedora ~]$ kubectl create deployment httpd --image=httpd:latest
deployment.apps/httpd created
```

Creating a Deployment

2. Now, run the below command to see the status of the pods created as a result of the deployment.

Running the command immediately produces the screenshot below, where the pod is still in the ContainerCreating state.

kubectl get pods



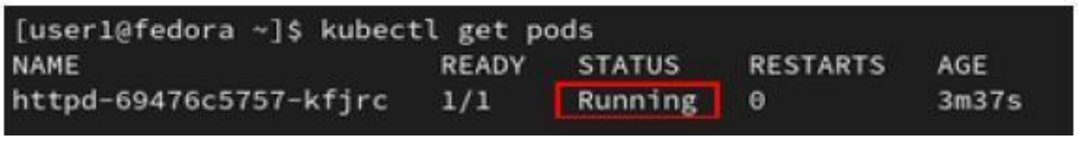
```
[user1@fedora ~]$ kubectl get pods
NAME                     READY   STATUS              RESTARTS   AGE
httpd-69476c5757-kfjrc   0/1     ContainerCreating   0          9s
```

Checking Pods Status

3. Lastly, rerun the kubectl command below as you did in step two after a few minutes.

kubectl get pods

The output below confirms the pod is Running, as shown below.



```
[user1@fedora ~]$ kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
httpd-69476c5757-kfjrc  1/1     Running   0          3m37s
```

Verifying the Pod is Running

Exposing the Deployment for External Access

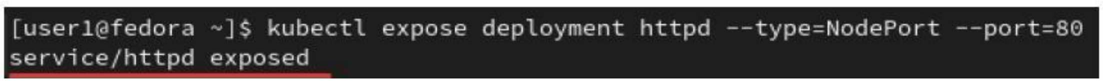
You’ve successfully deployed a web app-running pod, but the pod cannot be accessed from outside the cluster since external communication with the pod is disabled by default. You need to expressly expose (enable access to) the app for interaction via your web browser outside the cluster.

1. Run the kubectl expose command below to expose your deployment to the outside world with the standard port 80 using a NodePort service.

kubectl expose deployment httpd --type=NodePort --port=80

MiniKube is primarily used for development and testing. Using a NodePort is usually sufficient for such use cases. But in production, consider using a LoadBalancer or Ingress to expose your deployment as they are more secure and robust.

The output below confirms the port has been successfully exposed.



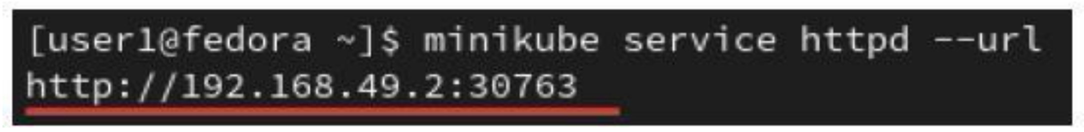
```
[user1@fedora ~]$ kubectl expose deployment httpd --type=NodePort --port=80
service/httpd exposed
```

Exposing Deployment Using a NodePort

2. Next, run the minikube service command below to obtain the external address (--url)for the deployment(httpd)

minikube service httpd --url

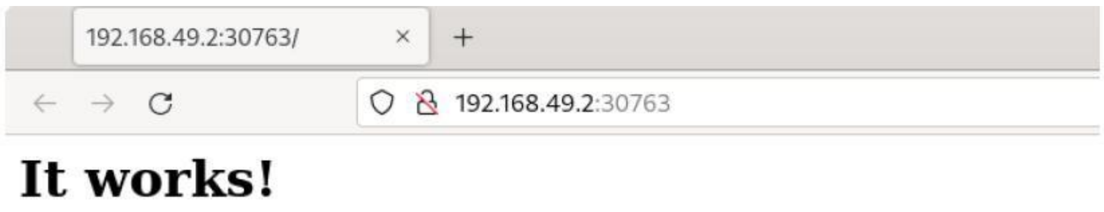
Note down the URL like in the output below, as you'll need it to access your web application (step three).



```
[user1@fedora ~]$ minikube service httpd --url
http://192.168.49.2:30763
```

Obtaining the external address of exposed deployment

3. Finally, open your preferred web browser and navigate to the URL you noted in step two. This URL redirects your browser to the default httpd container webpage, as shown below.



192.168.49.2:30763/

192.168.49.2:30763

It works!

Accessing Deployed Web Application

Pausing a Kubernetes Cluster

Your web server is now fully accessible from your local machine, so take this time to explore other features of MiniKube that can help administer your cluster. Read on about pausing the cluster for a start.

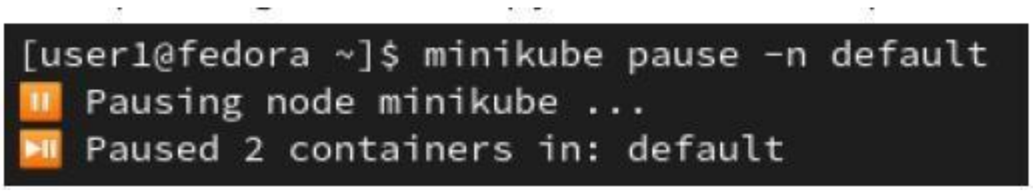
MiniKube allows you to pause running containers in a namespace with the pause command. The pause command lets you free up resources and also simulates service failure. Pausing is useful in resource-constrained development environments like MiniKube.

To pause a Kubernetes cluster:

1. Run the minikube pause command below to free up the processor and runs other containers. This command executes without losing the state of the paused containers in the default namespace (-n).

minikube pause -n default

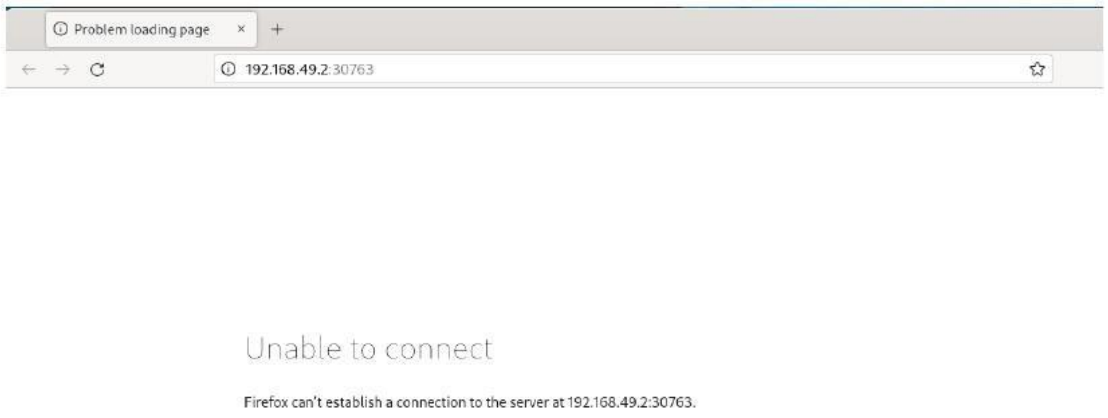
After pausing the cluster, you'll see an output similar below.



```
[user1@fedora ~]$ minikube pause -n default
⏸  Pausing node minikube ...
⏯  Paused 2 containers in: default
```

Pausing a Cluster

2. Switch back to your browser and notice an error message saying Unable to connect, as shown below. You get this error message since your browser cannot fetch the welcome page anymore after pausing the cluster.



Problem loading page

192.168.49.2:30763

Unable to connect

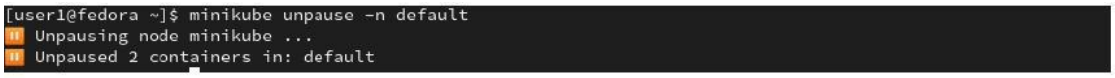
Firefox can't establish a connection to the server at 192.168.49.2:30763.

Verifying Paused Containers

3. Now, run the following minikube unpause command to resume the execution of the paused containers.

minikube unpause -n default

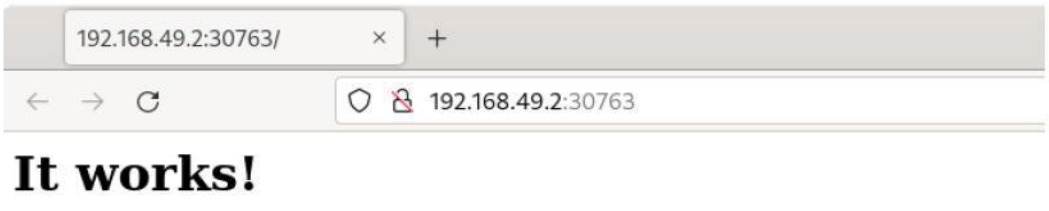
You'll see the output similar below after resuming the paused cluster.



```
[user1@fedora ~]$ minikube unpause -n default
⏸  Unpausing node minikube ...
⏸  Unpaused 2 containers in: default
```

Unpausing a Cluster

4. Lastly, switch back to your web browser and see the earlier It works message. This output indicates your web browser can connect to the web server again, as shown below.



It works!

Confirming Resumption of Containers

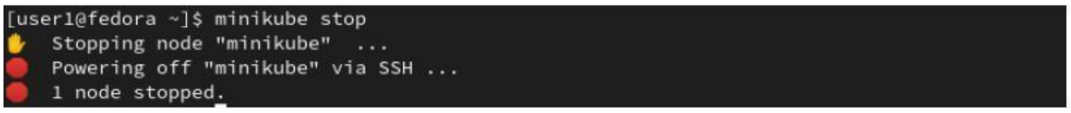
Stopping a Kubernetes Cluster

Unlike pausing a cluster, you can also stop a Kubernetes cluster entirely if it doesn’t serve any purpose anymore or if you just want to restart the cluster.

1. Run the minikube stop command below to stop the cluster altogether. Stopping the cluster preserves the current cluster configuration.

minikube stop

Accordingly, the terminal informs you of the progress as in the following screenshot.



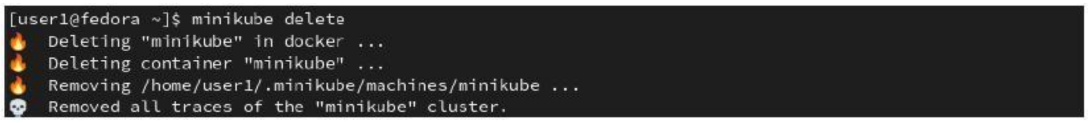
```
[user1@fedora ~]$ minikube stop
✋  Stopping node "minikube"  ...
🛑  Powering off "minikube" via SSH ...
🛑  1 node stopped.
```

Stopping MiniKube

Now, run the minikube delete command below to delete your Kubernetes cluster. This command removes the current VM or container running MiniKube, giving you the chance to start afresh.

minikube delete

You should get confirmation of the actions taken in your terminal, as in the screenshot below.



```
[user1@fedora ~]$ minikube delete
🔥  Deleting "minikube" in docker ...
🔥  Deleting container "minikube" ...
🔥  Removing /home/user1/.minikube/machines/minikube ...
💀  Removed all traces of the "minikube" cluster.
```

Deleting a Cluster