

# DATABASE MANAGEMENT AND DATA VISUALIZATION

[dmdvben/DM\\_DV: Code and files related to course in Database Management and Data Visualization](#)

## Droplet:

A Droplet is a virtual private server from DigitalOcean that lets you run applications, host websites, or store data in the cloud.

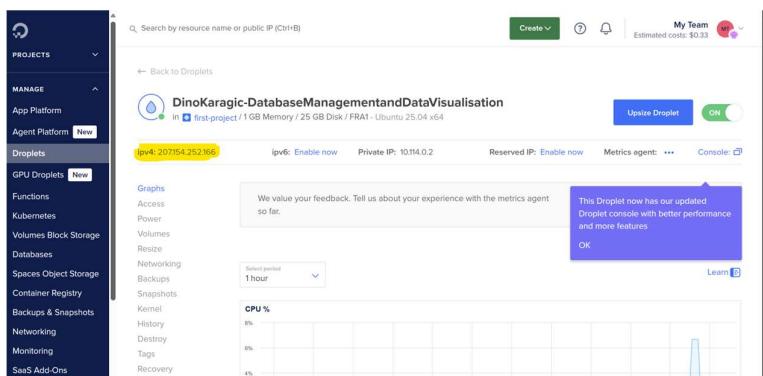
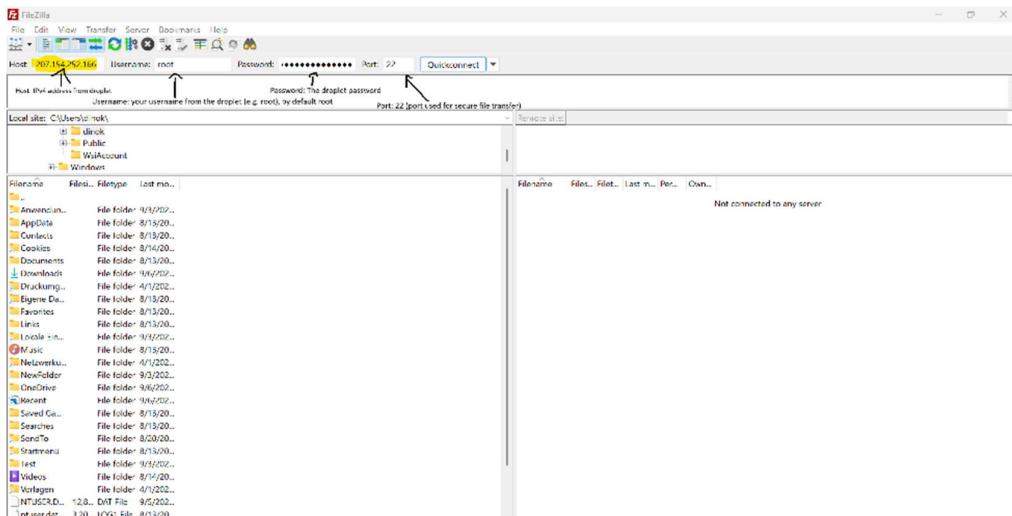
👉 Example: You can host a WordPress site on a Droplet and use FileZilla to easily upload your website files.

✓ Benefits: It's reliable (online 24/7), scalable (upgrade anytime), secure, and gives you full control compared to shared hosting or running on your own PC.

## Conect Droplet with FileZilla:

FileZilla itself is just a client. It doesn't store or change files on your machine unless you download or upload files.

When you connect FileZilla to a server (like your Droplet), it shows you the files on that server, not your local computer.



## Connect Command line Interface:

The screenshot shows the DigitalOcean Droplet interface. At the top, there's a navigation bar with 'Back to Droplets' and a droplet icon. Below it, the droplet name is 'DinoKaragic-DatabaseManagementandDataVisualisation' with status 'ON'. There are buttons for 'Upsize Droplet' and 'Metrics agent: ...'. A red box highlights the 'Console' button. A blue callout box says 'This Droplet now has our updated Droplet console with better performance and more features'. Below the interface is a terminal window showing:

```
*** System restart required ***
root@ubuntu-s-1vcpu-1gb-fra1-01:~# pwd
/root
root@ubuntu-s-1vcpu-1gb-fra1-01:~# 
```

`pwd` = current working directory.

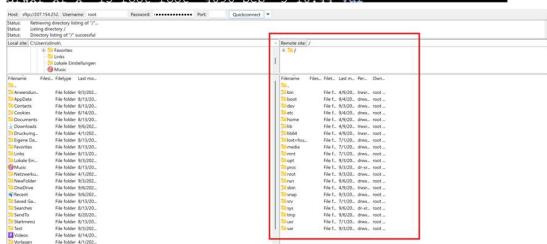
It shows you are currently in the `/root` folder, which is the home directory for the root user.

```
root@ubuntu-s-1vcpu-1gb-fra1-01:~# ls -l
total 4
drwxr-xr-x 2 root root 4096 Sep  3 11:32 Dino
```

`ls -l` = information about files and directories in the current directory

You are in the root user's home directory (`/root`) and there is **one folder named Dino** inside it. This folder is owned by root and has standard read/write/execute permissions for the owner and read/execute permissions for everyone else.

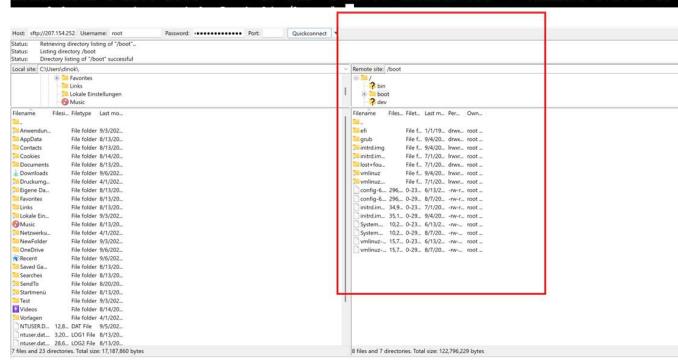
```
root@ubuntu-s-1vcpu-1gb-fra1-01:~# pwd
/root
root@ubuntu-s-1vcpu-1gb-fra1-01:~# ls -l
total 4
drwxr-xr-x 2 root root 4096 Sep  3 11:32 Dino
root@ubuntu-s-1vcpu-1gb-fra1-01:~# cd Dino
root@ubuntu-s-1vcpu-1gb-fra1-01:/# ls -l
total 60
lrwxrwxrwx 1 root root    7 Apr  9 12:06 bin -> usr/bin
drwxr-xr-x  5 root root 4096 Sep  4 06:49 boot
drwxr-xr-x  18 root root 3920 Sep  3 10:44 dev
drwxr-xr-x 117 root root 4096 Sep  4 06:51 etc
drwxr-xr-x  2 root root 4096 Apr  9 12:06 home
drwxrwxrwx  1 root root    7 Apr  9 12:06 lib -> usr/lib
drwxrwxrwx  1 root root 9 Apr  9 12:06 lib64 -> usr/lib64
drwx-----  2 root root 16384 Jul  1 19:57 lost+found
drwxr-xr-x  2 root root 4096 Jul  1 19:54 media
drwxr-xr-x  2 root root 4096 Jul  1 19:54 mnt
drwxr-xr-x  4 root root 4096 Sep  3 12:40 opt
dr-xr-xr-x 170 root root   0 Sep  3 10:44 proc
drwx-----  5 root root 4096 Sep  3 11:32 root
drwxr-xr-x  33 root root 1040 Sep  6 06:42 run
lrwxrwxrwx  1 root root   8 Apr  9 12:06 sbin -> usr/sbin
drwxr-xr-x  2 root root 4096 Sep  3 10:44 snap
drwxr-xr-x  2 root root 4096 Jul  1 19:54 srv
dr-xr-xr-x  13 root root   0 Sep  3 10:44 sys
drwxrwxrwt 11 root root 220 Sep  6 06:41 tmp
drwxr-xr-x  12 root root 4096 Jul  1 19:54 usr
drwxr-xr-x  13 root root 4096 Sep  3 10:44 var
```



cd .. moves **one level up** → from /root to / (the **top of the filesystem**).

ls -l now lists **all top-level directories and symbolic links** in /

```
root@ubuntu-s-lvcpu-lgb-frai-01:/# cd boot
root@ubuntu-s-lvcpu-lgb-frai-01:/boot# ls -l
total 119953
-rw----- 1 root root 10284289 Jun 13 18:24 System.map-6.14.0-23-generic
-rw----- 1 root root 10287886 Aug  7 14:44 System.map-6.14.0-29-generic
-rw-r--r-- 1 root root 296789 Jun 13 18:24 config-6.14.0-23-generic
-rw-r--r-- 1 root root 296778 Aug  7 14:44 config-6.14.0-29-generic
drwx---- 3 root root 512 Jan  1 1970 efi
drwxr-xr-x 6 root root 4096 Sep  4 06:49 grub
lrwxrwxrwx 1 root root 28 Sep  4 06:49 initrd.img -> initrd.img-6.14.0-29-generic
-rw-r--r-- 1 root root 34931204 Jul  1 19:56 initrd.img-6.14.0-23-generic
-rw-r--r-- 1 root root 35159299 Sep  4 06:49 initrd.img-6.14.0-29-generic
lrwxrwxrwx 1 root root 28 Jul  1 19:56 initrd.img.old -> initrd.img-6.14.0-23-generic
drwx---- 2 root root 16384 Jul  1 19:57 lost+found
lrwxrwxrwx 1 root root 25 Sep  4 06:49 vmlinuz -> vmlinuz-6.14.0-29-generic
-rw----- 1 root root 15767944 Jun 13 15:19 vmlinuz-6.14.0-23-generic
-rw----- 1 root root 15772040 Aug  7 15:34 vmlinuz-6.14.0-29-generic
lrwxrwxrwx 1 root root 25 Jul  1 19:56 vmlinuz.old -> vmlinuz-6.14.0-23-generic
```



You successfully navigated to /boot and listed all information about files and directories in the boot directory.

```
root@ubuntu-s-lvcpu-lgb-frai-01:/boot# cd ..
root@ubuntu-s-lvcpu-lgb-frai-01:/# ls -l
ls: cannot access '1-': No such file or directory
root@ubuntu-s-lvcpu-lgb-frai-01:/# ls -l
total 60
lrwxrwxrwx 1 root root 7 Apr  9 12:06 bin -> usr/bin
drwxr-xr-x 5 root root 4096 Sep  4 06:49 boot
drwxr-xr-x 18 root root 3920 Sep  3 10:44 dev
drwxr-xr-x 117 root root 4096 Sep  4 06:51 etc
drwxr-xr-x 2 root root 4096 Apr  9 12:06 home
lrwxrwxrwx 1 root root 7 Apr  9 12:06 lib -> usr/lib
lrwxrwxrwx 1 root root 9 Apr  9 12:06 lib64 -> usr/lib64
drwx---- 2 root root 16384 Jul  1 19:57 lost+found
drwxr-xr-x 2 root root 4096 Jul  1 19:54 media
drwxr-xr-x 2 root root 4096 Jul  1 19:54 mnt
drwxr-xr-x 4 root root 4096 Sep  3 12:40 opt
dr-xr-xr-x 167 root root 0 Sep  3 10:44 proc
drwx---- 5 root root 4096 Sep  3 11:32 root
drwxr-xr-x 33 root root 1040 Sep  6 06:58 run
lrwxrwxrwx 1 root root 8 Apr  9 12:06 sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Sep  3 10:44 snap
drwxr-xr-x 2 root root 4096 Jul  1 19:54 srv
dr-xr-xr-x 13 root root 0 Sep  6 06:51 sys
drwxrwxrwt 11 root root 220 Sep  6 06:41 tmp
drwxr-xr-x 12 root root 4096 Jul  1 19:54 usr
drwxr-xr-x 13 root root 4096 Sep  3 10:44 var
```

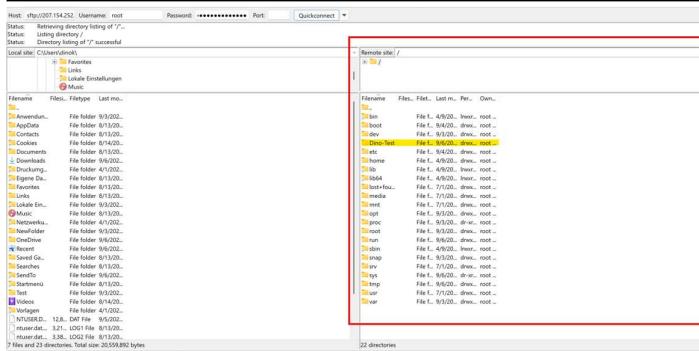
Cd moves back to parent folder

ls -l list all information about files and directories in the / directory.

```

root@ubuntu-s-1vcpu-1gb-fra1-01:/# mkdir Dino-Test
root@ubuntu-s-1vcpu-1gb-fra1-01:/# ls -l
total 64
drwxr-xr-x  2 root root  4096 Sep  6 07:03 Dino-Test
lwxrwxrwx  1 root root     7 Apr  9 12:06 bin -> usr/bin
drwxr-xr-x  5 root root  4096 Sep  4 06:49 boot
drwxr-xr-x 18 root root 3920 Sep  3 10:44 dev
drwxr-xr-x 117 root root 4096 Sep  4 06:51 etc
drwxr-xr-x  2 root root 4096 Apr  9 12:06 home
lwxrwxrwx  1 root root     7 Apr  9 12:06 lib -> usr/lib
lwxrwxrwx  1 root root     9 Apr  9 12:06 lib64 -> usr/lib64
drwx----- 2 root root 16384 Jul  1 19:57 lost+found
drwxr-xr-x  2 root root 4096 Jul  1 19:54 media
drwxr-xr-x  2 root root 4096 Jul  1 19:54 mnt
drwxr-xr-x  4 root root 4096 Sep  3 12:40 opt
dr-xr-xr-x 167 root root    0 Sep  3 10:44 proc
drwx----- 5 root root 4096 Sep  3 11:32 root
drwxr-xr-x 33 root root 1040 Sep  6 06:58 run
lwxrwxrwx  1 root root     8 Apr  9 12:06 sbin -> usr/sbin
drwxr-xr-x  2 root root 4096 Sep  3 10:44 snap
drwxr-xr-x  2 root root 4096 Jul  1 19:54 srv
dr-xr-xr-x 13 root root    0 Sep  6 06:51 sys
drwxrwxrwt 11 root root 220 Sep  6 06:41 tmp
drwxr-xr-x 12 root root 4096 Jul  1 19:54 usr
drwxr-xr-x 13 root root 4096 Sep  3 10:44 var
root@ubuntu-s-1vcpu-1gb-fra1-01:/#

```

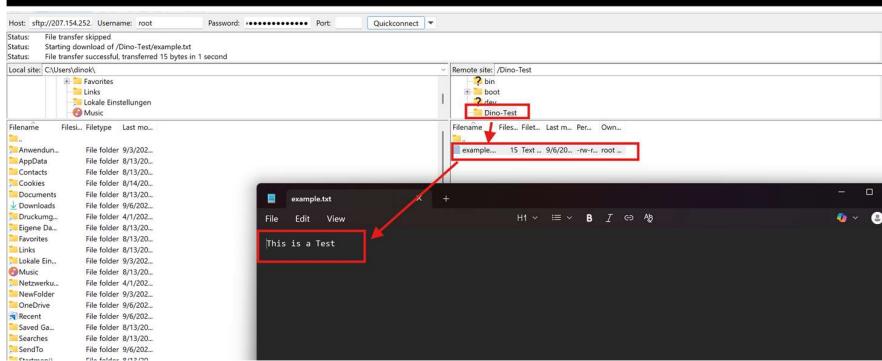


### Mkdir – Creates new folder (here created Dino-Test)

```

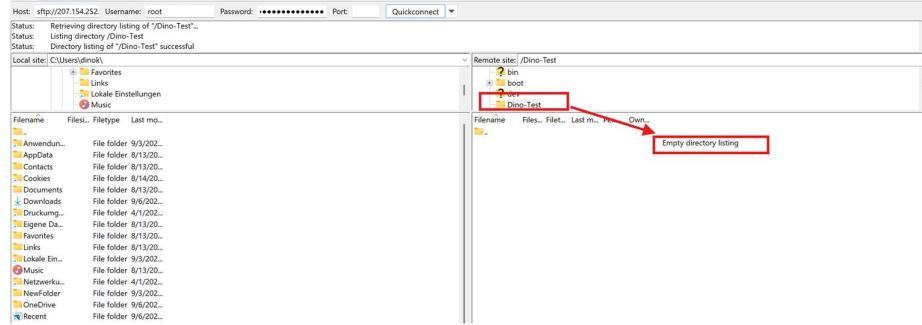
root@ubuntu-s-1vcpu-1gb-fra1-01:/# cd Dino-Test
root@ubuntu-s-1vcpu-1gb-fra1-01:/Dino-Test# ls -l
total 4
-rw-r--r-- 1 root root 12 Sep  6 07:16 example.txt
root@ubuntu-s-1vcpu-1gb-fra1-01:/Dino-Test# echo "This is a Test" > example.txt
root@ubuntu-s-1vcpu-1gb-fra1-01:/Dino-Test# cat example.txt
This is a Test
root@ubuntu-s-1vcpu-1gb-fra1-01:/Dino-Test# 

```



Moved to director Dino – Test, listed all the folders, example.txt was there, created in the example.txt a text “This is a Test” (“This is a Text” > example.txt) showed the text in the example test (cat example.txt)

```
root@ubuntu-s-lvcpu-1gb-fra1-01:/Dino-Test# rm example.txt
root@ubuntu-s-lvcpu-1gb-fra1-01:/Dino-Test# ls -l
total 0
```



Rm – removes the example.txt

```
root@ubuntu-s-lvcpu-1gb-fra1-01:/# rm -r Dino-Test
root@ubuntu-s-lvcpu-1gb-fra1-01:/# ls -l
total 60
lrwxrwxrwx  1 root root    7 Apr  9 12:06 bin  -> usr/bin
drwxr-xr-x  5 root root  4096 Sep  4 06:49 boot
drwxr-xr-x 18 root root  3920 Sep  3 10:44 dev
drwxr-xr-x 117 root root  4096 Sep  4 06:51 etc
drwxr-xr-x  2 root root  4096 Apr  9 12:06 home
lrwxrwxrwx  1 root root    7 Apr  9 12:06 lib  -> usr/lib
lrwxrwxrwx  1 root root    9 Apr  9 12:06 lib64 -> usr/lib64
drwxr----- 2 root root 16384 Jul  1 19:57 lost+found
drwxr-xr-x  2 root root  4096 Jul  1 19:54 media
drwxr-xr-x  2 root root  4096 Jul  1 19:54 mnt
drwxr-xr-x  4 root root  4096 Sep  3 12:40 opt
dr-xr-xr-x 166 root root    0 Sep  3 10:44 proc
drwxr----- 5 root root  4096 Sep  3 11:32 root
drwxr-xr-x  33 root root 1040 Sep  6 07:32 run
lrwxrwxrwx  1 root root   8 Apr  9 12:06 sbin -> usr/sbin
drwxr-xr-x  2 root root  4096 Sep  3 10:44 snap
drwxr-xr-x  2 root root  4096 Jul  1 19:54 srv
dr-xr-xr-x  13 root root    0 Sep  6 06:51 sys
drwxrwxrwt 11 root root  220 Sep  6 07:18 tmp
drwxr-xr-x  12 root root  4096 Sep  6 07:35 usr
drwxr-xr-x  13 root root  4096 Sep  3 10:44 var
root@ubuntu-s-lvcpu-1gb-fra1-01:/#
```

rm -r MyAssignments, deletes the directory and everything inside it.

## Docker:

Docker is a tool that lets you run apps in **containers** — like little boxes that have everything the app needs to work.

### What is a container?

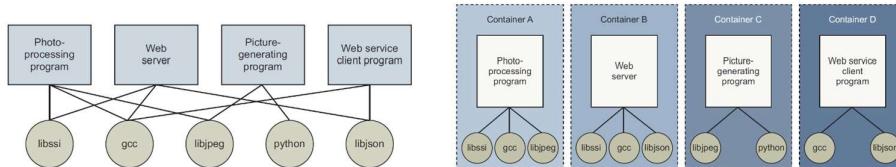
- A container is like a mini-computer inside your computer.
- It has its own software, settings, and files, separate from your main system.

## Example:

Imagine you want to run a website:

- Normally, you'd install a web server, database, and code on your PC.
- With Docker, you just get a **container** with all of that already inside.
- You can start it with one command, and it works instantly.

Complicated dependencies without Docker      Programs organized in Docker containers



#### ▪ **docker ps**: See containers in running state

```
root@test:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS        NAMES
root@test:~$ []
```

No containers running

#### ▪ **docker ps -a**: See all containers (both in running and created state)

```
root@test:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS        NAMES
1a0accfa4nee      hello-world        "/hello"          About a minute ago   Exited (0) About a minute ago
1dcace59b419       hello-world        "/hello"          58 minutes ago     Exited (0) 58 minutes ago
*   -
```

Two containers in the created state

#### ▪ **Docker remove [name of container]**: Remove a container in created state

```
root@test:~$ docker rm romantic_jepsen
root@test:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS        NAMES
1dcace59b419       hello-world        "/hello"          59 minutes ago     Exited (0) 59 minutes ago
*   -
```

17

After removing one container, only one container is in created state

## Docker Volume:

A volume is a special folder that lives outside the container but can be used by one or more containers.

It's used to store data persistently.

- Normally, if a container is deleted, all its data disappears.
- With a volume, the data stays safe even if the container is removed.

## What is a Shared Volume?

- A shared volume is a volume that multiple containers can access at the same time.
- This allows containers to read and write the same data, making it useful for collaboration between containers.

## Example

Imagine you have two containers:

1. A web server container
2. A backup container

You can create a shared volume /data:

### Volume: /data

```
Volume: /data
├─ Container 1: Web server writes logs
└─ Container 2: Backup container reads logs to save them
```

- Both containers can access /data, so logs are shared safely

## Create Volumes:

```
root@ubuntu-s-1vcpu-1gb-fral-01:~# docker volume create rstudio_data
rstudio_data
root@ubuntu-s-1vcpu-1gb-fral-01:~# docker volume ls
DRIVER      VOLUME NAME
local      postgres_data
local      rstudio_data
local      shiny_app_data
root@ubuntu-s-1vcpu-1gb-fral-01:~# docker volume inspect --format '{{ .Mountpoint }}' $(docker volume ls -q)
/var/lib/docker/volumes/postgres_data/_data
/var/lib/docker/volumes/rstudio_data/_data
/var/lib/docker/volumes/shiny_app_data/_data
root@ubuntu-s-1vcpu-1gb-fral-01:~# chmod o+rws /var/lib/docker/volumes/postgres_data/_data
root@ubuntu-s-1vcpu-1gb-fral-01:~# chmod o+rws /var/lib/docker/volumes/rstudio_data/_data
root@ubuntu-s-1vcpu-1gb-fral-01:~# chmod o+rws /var/lib/docker/volumes/shiny_app_data/_data
root@ubuntu-s-1vcpu-1gb-fral-01:~#
```

docker volume create rstudio\_data

Creates a volume on the host machine which can be accessed from the container  
docker volume ls

Returns a list of shared volume  
docker volume inspect --format '{{ .Mountpoint }}' \$(docker volume ls -q)  
Returns a list of shared volumes and their location on the host machi  
chmod o+rws /var/lib/docker/volumes/postgres\_data/\_data  
chmod o+rws /var/lib/docker/volumes/rstudio\_data/\_data  
chmod o+rws /var/lib/docker/volumes/shiny\_app\_data/\_data  
Create read and write permission for other users than the root user in host path  
associated with the shared volume.  
The path /var/lib/docker/volumes/rstudio\_data/\_data should be changed if the  
location of the shared volume is different

## Rstudio Docker:

**GET RSTUDIO CONTAINER RUNNING**

- `docker run -d -p 8787:8787 -e PASSWORD=ThisIs4ThePassword --name rstudio -v rstudio_data:/home/rstudio rocker/rstudio`
- `docker run`: start a container
- `-d`: run in detach mode (the container runs in the background and do not tie up your terminal)
- `-p 8787:8787`: The first 8787 is the host machine port, and the next 8787 is the port for the container. We map from the host machine port to the container port. Hence, when you access the host and specify that you connect to port 8787 (E.g., "droplet\_IP:8787") Docker will redirect traffic to the container.
- `-e PASSWORD = [your_password]`: Set the password for logging in to Rstudio
- `--name rstudio`: The name of the container
- `-v rstudio_data:/home/rstudio`: Mount the volume "rstudio\_data" to the container path "/home/rstudio"
- `rocker/rstudio`: The name of the image that you use to install and run the docker container
- See more information about rocker images: <https://rocker-project.org/images/versioned/rstudio.html>
- Find the ipv4 address of your digital ocean droplet, and access your Rstudio server in the browser using port 8787 (e.g., 165.22.92.178:8787)
- The username is "rstudio" and the password is as you gave in the docker run command

The screenshot shows the DigitalOcean interface. On the left, there's a sidebar with 'PROJECTS' and 'MANAGE' sections. Under 'Droplets', a list includes 'GPU Droplets', 'Functions', 'Kubernetes', 'Volumes Block Storage', and 'Databases'. A specific droplet is selected with the name 'DinoKaragic-DatabaseManagementandDataVisualisation'. Below it, details show 'IPv4: 207.154.252.166' and 'IPv6: Enable now'. A message box is overlaid on the page, stating 'This Droplet now has our updated Droplet console with better performance and more features' with an 'OK' button. At the bottom, a browser window shows the RStudio sign-in page at the URL 207.154.252.166:8787.

## VERIFY PERSISTENT STORAGE IN SHARED VOLUME

- Login to rstudio from your browser
- In R: Install the R package "pryr" with `install.packages("pryr")`
- Create a new R script where you load the R package: `library(pryr)`
  - Verify that `library(pryr)` works
- Then stop and remove the rstudio container (from the Linux CLI)
  - `docker stop rstudio`
  - `docker rm rstudio`
- Use filezilla to connect to the server and go to the host folder where the shared volume for `rstudio_data` is located and verify that the R script is in that folder (despite the rstudio container has been removed)
  - Remember: when you created the shared volume you specified the host path for the shared volume
- Spin up the rstudio container again and verify that the rscript is still there
  - `docker run -d -p 8787:8787 -e PASSWORD=ThisIs4ThePassword --name rstudio -v rstudio_data:/home/rstudio rocker/rstudio`
- Open the R script and try to run `library(pryr)`
  - You should get an error, as the library is not installed
  - The reason is that the rstudio container is not built with this package! It is only the data in the shared volume which is persisted
  - This is an advantage when different users use the same image. We can create our own images with the packages we wish to have installed →

## INSPECTING AND ENTERING A CONTAINER

- **docker exec -it rstudio /bin/bash**
  - **docker exec**: execute a command in a container
  - **-it**: the i stands for *interactive*, and t for *terminal*.
  - **/bin/bash**: ls the command executed and we are starting an interactive shell within the container
- **id rstudio**: In the shell of the rstudio container, we inspect the user id and group id of the user called rstudio. Rstudio is the default user when using the rocker/rstudio image.
- **exit**: leave the interactive terminal in the container and return to host

```
root@ubuntu-s-1vcpu-1gb-fra1-01:~# docker exec -it rstudio /bin/bash
root@5a2be9cd8a04:/# id rstudio
uid=1000(rstudio) gid=1000(rstudio) groups=1000(rstudio),50(staff)
root@5a2be9cd8a04:/# exit
exit
root@ubuntu-s-1vcpu-1gb-fra1-01:~# █
```

**docker exec -it rstudio /bin/bash** = This command lets you **enter the RStudio container** and manipulate its filesystem, install packages, or inspect files just like you would on a normal server.

**Id rstudio** = inspects user id and group id of the user called rstudio

**Exit** = Leaves the container

## Shiny Docker:

```
docker run -d -p 3838:3838 --name rshiny -v shiny_app_data:/srv/shiny-server
rocker/shiny
```

```
[root@ubuntu-a-1vcpu-1gb-fral-01: ~]# docker run -d -p 3838:3838 --name rshiny -v shiny_app_data:/srv/shiny-server rocker/shiny
latest: Pulling from rocker/shiny
9d3d52c1bb0d: Already exists
a725794c0e5a: Already exists
a172b5443207: Already exists
ea32954d4a0f: Already exists
aa8e97b0a8b1: Already exists
00411d571e02: Already exists
17191a55fd34: Pull complete
fc0b049fbfbcc: Pull complete
3f8f7c7e5545: Pull complete
00411d571e02: Pull complete
15752b1b061: Pull complete
0062c1982440: Pull complete
Digest: sha256:74367d163ce2f2d5ed35efd74f11fd43edb1f4dd92c26936f323b31d435658a2a
Status: Downloaded newer image for rocker/shiny:latest
7e30e982ce938614d413ab0a2d30b82eb3d64c0bf694152ac6984a1322b7f
[root@ubuntu-a-1vcpu-1gb-fral-01: ~]
```

- **docker pull rocker/shiny:** Download the image without creating the container
  - If we were to use the `docker run` command below, then the image would automatically be downloaded
  - Hence, I only show this command so you know we can download images without running them immediately
- **docker run -d -p 3838:3838 --name rshiny -v shiny\_app\_data:/srv/shiny-server rocker/shiny**
- Find the ipv4 address of your digital ocean droplet, and access Shiny server in the browser using port 3838 (e.g., 165.22.92.178:3838)

← Back to Droplets

DinoKaragic-DatabaseManagementandDataVisualisation  
in first-project / 1 GB Memory / 25 GB Disk / FRA1 - Ubuntu 25.04 x64

Upsize Droplet **ON**

ipv4: 207.154.252.166    ipv6: Enable now    Private IP: 10.114.0.2    Reserved IP: Enable now    Metrics agent: ...    Console:

Graphs    Access    Power    Volumes    Resize    Networking    Backups    Select period: 1 hour    Learn

We value your feedback. Tell us about your experience with the metrics agent so far.

This Droplet now has our updated Droplet console with better performance and more features

OK

Welcome to Shiny Server!  
If you're seeing this page, that means Shiny Server is installed and running. Congratulations!  
What's Next?  
Now you're ready to setup Shiny — if you haven't already — and start deploying your Shiny applications.  
If you see a Shiny application running on the right side of this page, then Shiny is configured properly on your server and already running an example. Bravo! You can see this application on your server at [/sample-apps/hello](#).  
If you see a gray box or an error message, then there's a bit more work to do to get Shiny running fully. You can continue with the installation instructions or use the Admin Guide for more information. If you're seeing an error message in the panel to the right, you can use it to help diagnose what may be wrong. If you think Shiny is installed and setup properly and things still aren't working, you can look in the Shiny Server log which may have more information about what's wrong. By default, the log is stored in [/var/log/shiny-server.log](#).  
If you're really stuck and you've read the relevant sections in the Admin Guide then please ask for help on our RStudio Community forum.  
rmarkdown  
Once you have Shiny working properly (the top application on the right)

It's Alive!

Number of bins: 30

Histogram of x

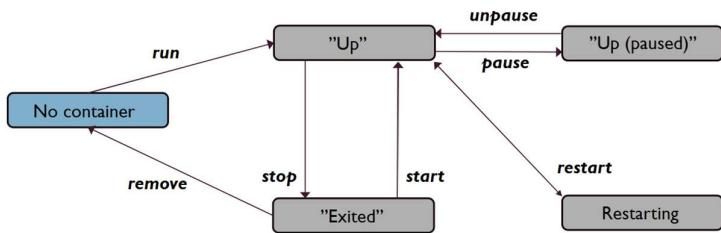
Frequency

x

When Shiny is properly configured on your server, you'll see a Shiny app above.

## Docker Container States:

### CONTAINER STATES



**Docker ps -a:** Inspect containers in "running", "exited" and "Up (paused)" states

**Docker ps:** Inspect containers in only "running" and "Up (paused)" states

```

root@ubuntu-s-1vcpu-1gb-fral-01:~# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
7e30e982ce93 rocker/shiny "/init" 4 minutes ago Up 4 minutes 0.0.0.0:3838->3838/tcp, [::]:3838->3838/tcp rshiny
5a2be9cd8a04 rocker/rstudio "init" 14 minutes ago Up 14 minutes 0.0.0.0:8787->8787/tcp, [::]:8787->8787/tcp rstudio

root@ubuntu-s-1vcpu-1gb-fral-01:~# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e30e982ce93 rocker/shiny "/init" 5 minutes ago Up 5 minutes 0.0.0.0:3838->3838/tcp, [::]:3838->3838/tcp rshiny
a2be9cd8a04 rocker/rstudio "init" 15 minutes ago Up 15 minutes 0.0.0.0:8787->8787/tcp, [::]:8787->8787/tcp rstudio
e9acf4f42f08 hello-world "/hello" 2 hours ago Exited (0) 2 hours ago wonderful_fermi
8dcf498a0ff hello-world "/hello" 2 days ago Exited (0) 2 days ago intelligent_curran
  
```

## Remove a container:

First stop container with:

Docker stop container\_name

Then remove container:

Docker rm container\_name

Networks:

Protocol

- A protocol is like a language computers use to talk to each other.
- Example:
  - HTTP → the language for web pages
  - If both sides “speak” HTTP, they can exchange web data correctly.

Interface

- An interface is like a door to a computer.
- Each interface has an IP address, which is like the house address of the computer.
- Example:
  - 192.168.0.1 → a specific computer on your network

Port

- A port is like a room inside the house (computer) for a specific service.
- Example:
  - RStudio Server → port 8787
  - RShiny → port 3838
  - PostgreSQL → port 5432
- A program listens on its port so clients know where to send data.

Putting it together

Example:

- You want to access a website on Wendy’s computer:
  - IP address (door) : 192.168.0.1
  - Port (room) : 80 (web server)
  - Protocol (language): HTTP
- Your web browser “knocks” on the door (IP) and asks for data in the correct room (port) using HTTP.

Two Networks:

Host network

- This is the network your computer or server is connected to.
- Example:
  - Your laptop connected to Wi-Fi at home → that’s your host network.
  - A server connected to the internet → that’s its host network.
- It defines how the machine communicates with other computers.

Docker virtual network (Bridge)

- Docker creates its own virtual network to let containers talk to each other and the host.
- The bridge network is like a virtual LAN inside your host machine.
- Containers connected to the bridge can:
  - Communicate with each other using container names
  - Connect to the outside world via the ho

## Inspecting Docker Network: (docker network ls)

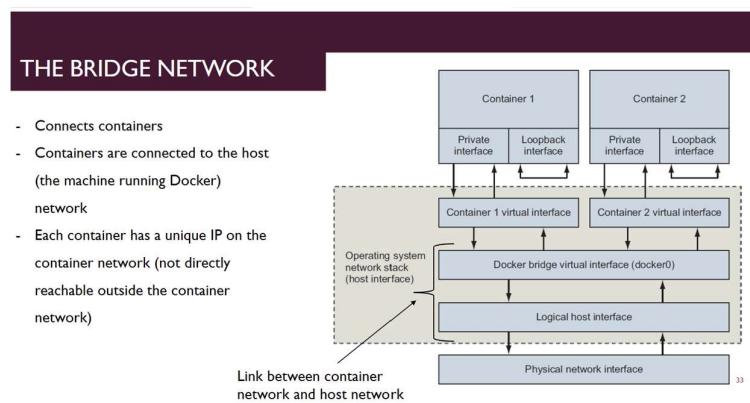
```
root@ubuntu-s-1vcpu-1gb-fra1-01:~# docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
3aa02e02c60a    bridge    bridge      local
b8cf7fbef2f2   host      host       local
63b3015476f2   none      null      local
root@ubuntu-s-1vcpu-1gb-fra1-01:~#
```

Bridge: Provides intercontainer connectivity on the same machine

Host: Containers connected to host interact with the host network

None: Containers connected to this does not have network connectivity outside themselves

Local scope: The network is constrained to the host running Dock



## Build a New Network

Creating a network is helpful so we can have different containers in the network.

```
docker network create \
--driver bridge \
--attachable \
--scope local \
--subnet 10.0.42.0/24 \
--ip-range 10.0.42.128/25 \
db_r_shiny

root@ubuntu-s-1vcpu-1gb-fra1-01:~# docker network create \
--driver bridge \
--attachable \
--scope local \
--subnet 10.0.42.0/24 \
--ip-range 10.0.42.128/25 \
db_r_shiny
72bf3977433b7dad563e545a13ec866902d910799a362fdc3b68c12ba8192140
root@ubuntu-s-1vcpu-1gb-fra1-01:~#
```

Docker network create:

Command to build a new network

driver bridge:

Using the bridge driver to create a bridge network

attachable:

Allows us to attach and detach containers at any time

scope local:

A network only available on the host

subnet 10.0.42.0/24:

Considers all IP-addresses from 10.0.42.0 to 10.0.42.255 as local

ip-range 10.0.42.128/25:

Only allocate IP-addresses to new containers in the range from 10.0.42.128 to 10.0.42.255

Hence we reserve the first IP-addresses for other purposes like manual assignment of IP address

Attach a Container to the Network: (docker network connect db\_r\_shiny rstudio)

```
root@ubuntu-s-1vcpu-1gb-fra1-01:~# docker network connect db_r_shiny rstudio
```

Docker network connect: Command to connect a container to network  
db\_r\_shiny: The network name  
rstudio: The container nam

## Postgres SQL Container and Connect to Network

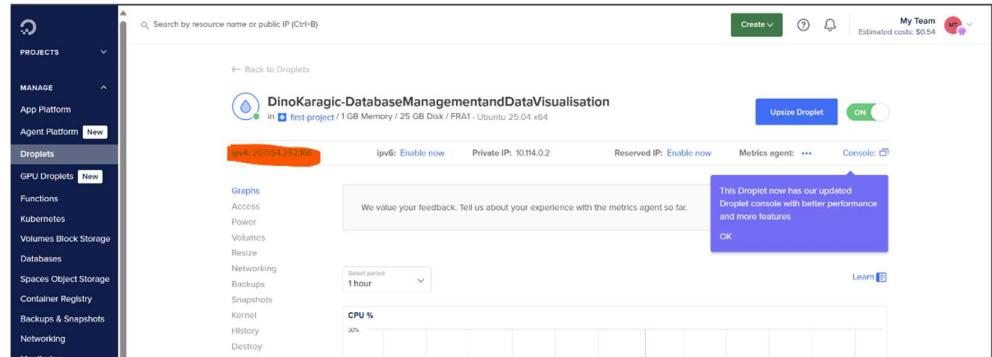
### Install Docker

```
docker run -p 5432:5432 --name postgres -e
```

```
POSTGRES_PASSWORD=ThisIs4ThePassword -e POSTGRES_USER=postgres -d -v
```

```
postgres_data:/var/lib/postgresql/data postgres
```

```
root@ubuntu-s-1vcpu-1gb-fra1-01:~# docker run -p 5432:5432 --name postgres -e POSTGRES_PASSWORD=ThisIs4ThePassword -e POSTGRES_USER=postgres -d -v postgres_data:/var/lib/postgresql/data postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
396b1da7636e: Pull complete
b72d8ac07e9d: Pull complete
94e1cd2b7c29: Pull complete
33d8bc47040c: Pull complete
5f1c297d801d: Pull complete
073d4679511b: Pull complete
e69073b30e46: Pull complete
2705262d29aa: Pull complete
2ad4158055ed: Pull complete
102a5a231171: Pull complete
b9d50a463683: Pull complete
41da6c673ba5: Pull complete
04777d59042a: Pull complete
756eeef103c01: Pull complete
Digest: sha256:d17be73073d6e73a005c239c67b8c9f0331925d05bd874964f445a9f698fbf72
Status: Downloaded newer image for postgres:latest
1b8ccb674df93dd5fc24a5c42ef58ccb0dea3ad673bca3cdc0da9dc98da2b6d8
```



## Connect postgress sql to network:

```
docker network connect db_r_shiny postgres
```

```
root@ubuntu-s-1vcpu-1gb-fra1-01:~# docker network connect db_r_shiny postgres
```

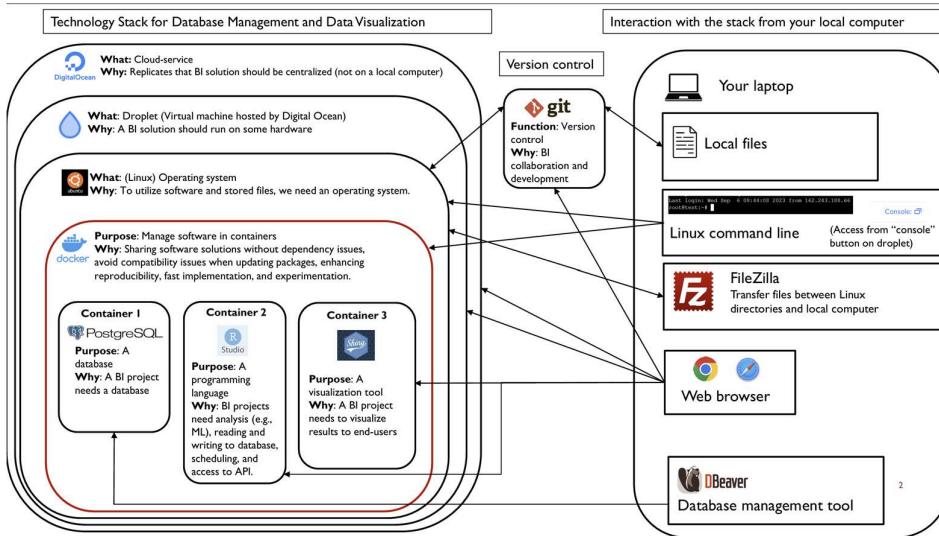
## Inspect the network:

```
docker network inspect db_r_shiny
```

```

root@Ubuntu-s-1vcpu-1gb-fral-01:~# docker network inspect db_r_shiny
[{"Name": "db_r_shiny", "Id": "72d137743107dad563e545a13ec866902d910799a362fcd3b68c12ba8192140", "Created": "2025-09-06T10:02:07.167438936Z", "Scope": "local", "Driver": "bridge", "EnableIPv6": true, "LinkLocalIPv6PrefixLen": 0, "IPAM": {"Driver": "default", "Options": {}, "Config": [{"Subnet": "10.0.42.0/24", "IPRange": "10.0.42.128/25"}]}, "Internal": false, "Attachable": true, "Ingress": false, "ConfigFrom": "", "Network": "", "Containerony": false, "Containers": [{"Name": "postgres", "Id": "5a2be9c656c43df1b49f46bfaf017e8abc5a7e1c2ce93fe67cbced959d07e2b16ea", "MacAddress": "32:ad:00:f8:4d:62", "IPV4Address": "10.0.42.130/24", "IPV6Address": ""}, {"Name": "rstudio", "Id": "5a2be9c656c43df1b49f46bfaf017e8abc5a7e1c2ce93fe67cbced959d07e2b16ea", "MacAddress": "fa:c7:1a:91:76:32", "IPV4Address": "10.0.42.129/24", "IPV6Address": ""}], "Options": {}, "Labels": {}}

```



## Images-Docker

A **Docker image** is like a **recipe** or **blueprint**.

It contains **everything needed to run a program**, such as:

- the application code,
- libraries,
- dependencies,
- and settings.

A **container** is basically the **environment where your application actually runs**.

You can use the same image to create many running copies → called **containers**.

## Analogy

- **Image = Cake recipe** 🍰
- **Container = Actual cake** 🍰
  - You can bake many cakes (containers) from one recipe (image).

## Inspect Image

(`docker image ls`)

```
*** System restart required ***
root@ubuntu-s-1vcpu-1gb-fral-01:~# docker image ls
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
postgres        latest   47fa19484031  2 days ago   454MB
hello-world     latest   1b44b5a3e06a  4 weeks ago  10.1kB
rocker/shiny     latest   26a63566fbca  2 months ago  1.64GB
rocker/rstudio   latest   35c3cf057620  2 months ago  2.32GB
root@ubuntu-s-1vcpu-1gb-fral-01:~# █
```

`docker rmi repository:TAG`

Removes an image – remember to specify the TAG of the image

Dockerfile:

- A Dockerfile is just a recipe (text file) that tells Docker how to build an image.
- Each line in the Dockerfile is one instruction (like “add ingredients” or “turn on the oven”).
- Docker reads it top to bottom and creates the image step by step.

### DOCKERFILE EXAMPLE

```
FROM gives the base
image we wish to build
upon (fetched from
Docker hub)
# Start from the rocker/rstudio base image which has R and RStudio pre-installed
FROM rocker/rstudio
# The RUN command executes shell commands during the image building process.
RUN apt-get update && apt-get install -y \
    git
# Install R packages using R's built-in 'install.packages' function.
RUN R -e 'install.packages(c("DBI", "RPostgres"))'

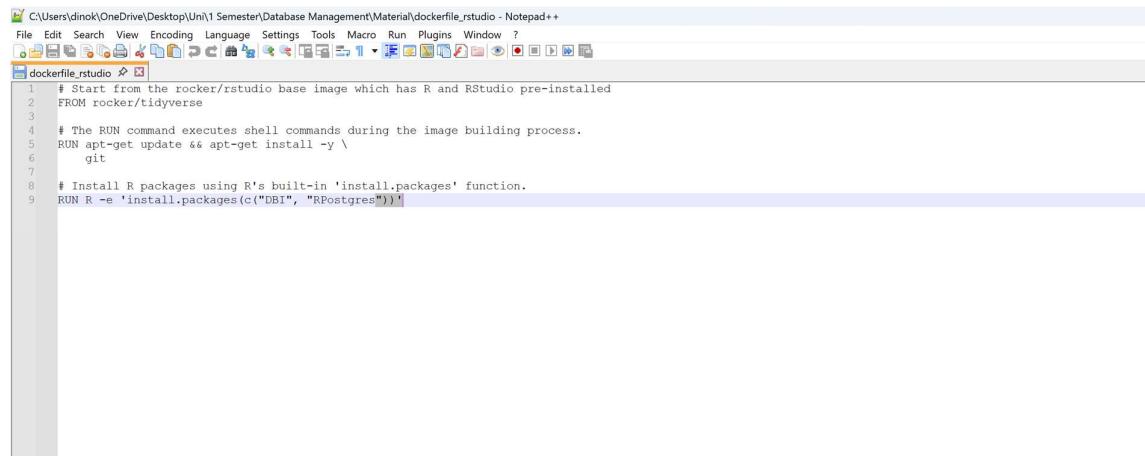
- RUN: Execute commands in our Linux terminal
- apt-get update: Updates packages meta-information (does not upgrade existing
  packages)
- &&: the command after the && will be run if the preceding command was successful
- apt-get install -y: Installs one or more packages and specifies that we wish to
  answer 'yes' to any prompts.
- \: This is a continuation symbol to ease readability. We might as well have written
  "install -y git"
- git: is an Ubuntu package we wish to install. If we wish to install other packages, like
  e.g. libxml2-dev, we could continue after a space like git libxml2-dev.
```

Include comments with "#"

The terminal initialize R with RUN R. Using -e 'install.packages(...)', instructs R to evaluate the string as an R expression.

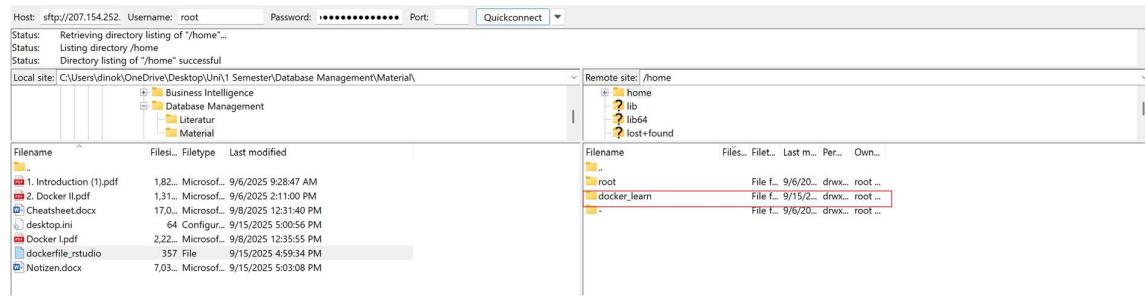
## Build Images from Dockerfile

We have the file dockerfile\_rstudio



```
C:\Users\dnok\OneDrive\Desktop\Uni\1 Semester\Database Management\Material\dockerfile_rstudio - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
dockefile_rstudio  
1 # Start from the rocker/rstudio base image which has R and RStudio pre-installed  
2 FROM rocker/tidyverse  
3  
4 # The RUN command executes shell commands during the image building process.  
5 RUN apt-get update && apt-get install -y \  
6 git  
7  
8 # Install R packages using R's built-in 'install.packages' function.  
9 RUN R -e 'install.packages(c("DBI", "RPostgres"))'
```

Create a file on remote environment right click mouse



Host: sftp://207.154.252.166 Username: root Password: \*\*\*\*\* Port: Quickconnect

Status: Retrieving directory listing of "/home"...

Status: Listing directory /home

Status: Directory listing of "/home" successful

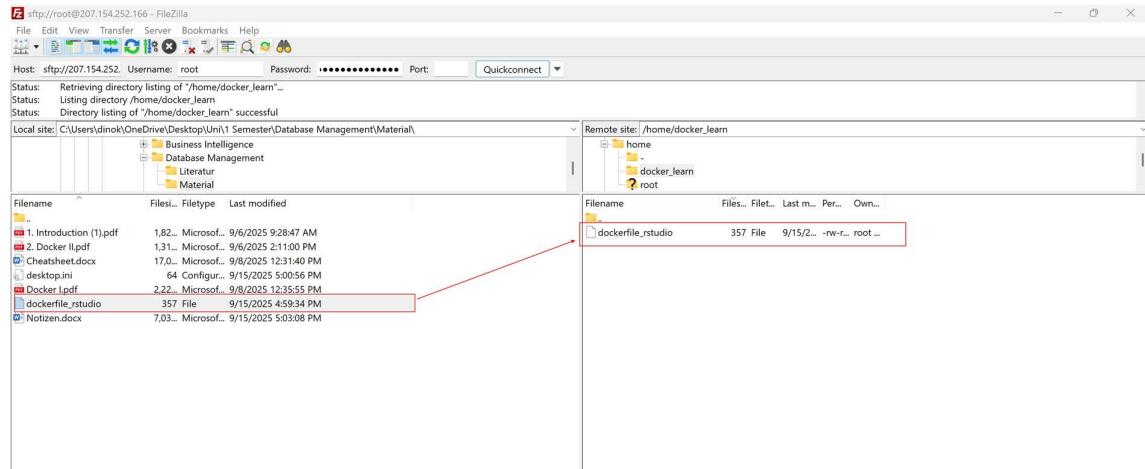
Local site: C:\Users\dnok\OneDrive\Desktop\Uni\1 Semester\Database Management\Material\

Filename	Files...	Filetype	Last modified
1. Introduction (1).pdf	1.82...	Microsoft...	9/6/2025 9:28:47 AM
2. Docker II.pdf	1.31...	Microsoft...	9/6/2025 2:11:00 PM
Cheatsheet.docx	17.0...	Microsoft...	9/8/2025 12:31:40 PM
desktop.ini	64	Configur...	9/15/2025 5:00:56 PM
Docker I.pdf	2.22...	Microsoft...	9/8/2025 12:35:55 PM
dockerfile_studio	357	File	9/15/2025 4:59:34 PM
Notizen.docx	7.03...	Microsoft...	9/15/2025 5:03:08 PM

Remote site: /home

Filename	Files...	Filetype	Last m...	Per...	Own...
root	File f...	9/6/20...	drwx...	root	...
docker_learn	File f...	9/15/2...	drwx...	root	...
root	File f...	9/6/20...	drwx...	root	...

Drag and drop the file from your local environment to the remote environment



FileZilla - sftp://207.154.252.166 - FileZilla

File Edit View Transfer Server Bookmarks Help

Host: sftp://207.154.252.166 Username: root Password: \*\*\*\*\* Port: Quickconnect

Status: Retrieving directory listing of "/home/docker\_learn"...

Status: Listing directory /home/docker\_learn

Status: Directory listing of "/home/docker\_learn" successful

Local site: C:\Users\dnok\OneDrive\Desktop\Uni\1 Semester\Database Management\Material\

Filename	Files...	Filetype	Last modified
1. Introduction (1).pdf	1.82...	Microsoft...	9/6/2025 9:28:47 AM
2. Docker II.pdf	1.31...	Microsoft...	9/6/2025 2:11:00 PM
Cheatsheet.docx	17.0...	Microsoft...	9/8/2025 12:31:40 PM
desktop.ini	64	Configur...	9/15/2025 5:00:56 PM
Docker I.pdf	2.22...	Microsoft...	9/8/2025 12:35:55 PM
dockerfile_studio	357	File	9/15/2025 4:59:34 PM
Notizen.docx	7.03...	Microsoft...	9/15/2025 5:03:08 PM

Remote site: /home/docker\_learn

Filename	Files...	Filetype	Last m...	Per...	Own...
root	File f...	9/6/20...	drwx...	root	...
dockerfile_rstudio	357	File	9/15/2...	-rw-r...	root

Change the cd to where the file is located

```
root@ubuntu-s-lvcpu-1gb-fral-01:/# ls -l
total 60
drwxrwxrwx  1 root root    7 Apr  9 12:06 bin -> usr/bin
drwxr-xr-x  5 root root  4096 Sep  4 06:49 boot
drwxr-xr-x 18 root root  3920 Sep  3 10:44 dev
drwxr-xr-x 117 root root  4096 Sep 12 06:23 etc
drwxr-xr-x  5 root root  4096 Sep  6 12:44 home
drwxrwxrwx  1 root root    7 Apr  9 12:06 lib -> usr/lib
drwxrwxrwx  1 root root    9 Apr  9 12:06 lib64 -> usr/lib64
drwx----- 2 root root 16384 Jul  1 19:57 lost+found
drwxr-xr-x  2 root root  4096 Jul  1 19:54 media
drwxr-xr-x  2 root root  4096 Jul  1 19:54 mnt
drwxr-xr-x  4 root root  4096 Sep  3 12:40 opt
dr-xr-xr-x 186 root root   0 Sep  3 10:44 proc
drwx----- 6 root root  4096 Sep 15 08:52 root
drwxr-xr-x 34 root root 1060 Sep 15 15:12 run
drwxrwxrwx  1 root root   8 Apr  9 12:06 sbin -> usr/sbin
drwxr-xr-x  2 root root  4096 Sep  3 10:44 snap
drwxr-xr-x  2 root root  4096 Jul  1 19:54 srv
dr-xr-xr-x 13 root root   0 Sep  6 06:51 sys
drwxrwxrwt  6 root root 120 Sep 15 14:56 tmp
drwxr-xr-x 12 root root  4096 Sep  6 07:35 usr
drwxr-xr-x 13 root root  4096 Sep  3 10:44 var
root@ubuntu-s-lvcpu-1gb-fral-01:/# cd home
root@ubuntu-s-lvcpu-1gb-fral-01:/home# ls -l
total 12
drwxr-xr-x 2 root root 4096 Sep  6 12:44 -
drwxr-xr-x 2 root root 4096 Sep 15 15:05 docker_learn
drwxr-xr-x 3 root root 4096 Sep  6 09:10 root
root@ubuntu-s-lvcpu-1gb-fral-01:/home# cd docker_learn
root@ubuntu-s-lvcpu-1gb-fral-01:/home/docker_learn# 
```

To build the image run:

```
docker image build --tag rstudio:1.0.0 -f dockerfile_rstudio .
```

- docker image build: Initiate building of the image
- --tag rstudio:1.0.0: Gives the name and tag of the image ([name]:[tag])
- -f dockerfile\_rstudio: Gives the name of the Dockerfile
- . : The „space-dot“ indicates that the dockerfile is to be found in your current directory

```
root@ubuntu-s-lvcpu-1gb-fral-01:/home/docker_learn# docker image build --tag rstudio:1.0.0 -f dockerfile_rstudio .
[+] Building 0.8s (7/7) FINISHED
=> [internal] load build definition from dockerfile_rstudio
=> => transferring dockerfile: 404B
=> [internal] load metadata for docker.io/rocker/tidyverse:latest
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/rocker/tidyverse:latest@sha256:6a7e93b235cbdbdf63ca81b1aa7e771ccebd1b54b30dd8f92f560475155119
=> CACHED [2/3] RUN apt-get update && apt-get install -y git
=> CACHED [3/3] RUN R -e 'install.packages(c("DBI", "RPostgres"))'
=> exporting to image
=> => exporting layers
=> => writing image sha256:alcdaf05557396d579db6f6562db79706352397f24f3ae56da02aae4cd96d54b
=> => naming to docker.io/library/rstudio:1.0.0
```

Now run a container with the new image:

First delete the old container so we can create new container with the name r\_studio  
(we cannot have two containers with the same name)

Delete container by:

docker rm -f rstudio

- f is a flag that force the container to be stopped if it is running

```
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
55d33d269eec rocker/rstudio "/init" 15 seconds ago Up 14 seconds 0.0.0.0:8787->8787/tcp, [::]:8787->8787/tcp rstudio
1b8cb674df93 postgres "docker-entrypoint.s..." 9 days ago Up 9 days 0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp postgres
7e30e982ce93 rocker/shiny "/init" 9 days ago Up 9 days 0.0.0.0:3838->3838/tcp, [::]:3838->3838/tcp rshiny
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker rm -f rstudio
rstudio
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1b8cb674df93 postgres "docker-entrypoint.s..." 9 days ago Up 9 days 0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp postgres
7e30e982ce93 rocker/shiny "/init" 9 days ago Up 9 days 0.0.0.0:3838->3838/tcp, [::]:3838->3838/tcp rshiny
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# 
```

We now build a new rstudio container using the image we just created:

docker run -d --network db\_r\_shiny -p 8787:8787 -e PASSWORD=xxxxxxxxx --name rstudio -v rstudio\_data:/home/rstudio **rstudio:1.0.0**

- network db\_r\_shiny: This option connects your container to the bridge network db\_r\_shiny
  - v rstudio\_data:/home/rstudio
- This is the shared volume

- Rstudio:1.0.0** is the name of the image we created above

```
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1b8cb674df93 postgres "docker-entrypoint.s..." 9 days ago Up 9 days 0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp postgres
7e30e982ce93 rocker/shiny "/init" 9 days ago Up 9 days 0.0.0.0:3838->3838/tcp, [::]:3838->3838/tcp rshiny
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker run -d --network db_r_shiny -p 8787:8787 -e PASSWORD=xxxxxxxxx --name rstudio -v rstudio_data:/home/rstudio rstudio:1.0.0
7a6295d1fa8613b94c85f55ac6467c96440ab71a1ccdb07ba47f5dcd9
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
47a6295d1fa8613b94c85f55ac6467c96440ab71a1ccdb07ba47f5dcd9
postgres "docker-entrypoint.s..." 22 seconds ago Up 22 seconds 0.0.0.0:8787->8787/tcp, [::]:8787->8787/tcp rstudio
7e30e982ce93 rocker/shiny "/init" 9 days ago Up 9 days 0.0.0.0:3838->3838/tcp, [::]:3838->3838/tcp rshiny
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# 
```

Now when we go to R we can just library(DBI) and library(RPostgres) as we have already installed the packages while creating the image with the file

The screenshot shows the RStudio interface with the console tab active. The console output is as follows:

```
R - R 4.5.1 ... 
* installing *binary* package 'stringr' ...
* DONE (stringr)
* installing *binary* package 'pryr' ...
* DONE (pryr)

The downloaded source packages are in
  '/tmp/RtmpGkv3D2/downloaded_packages'
> (DBI)
Error: object 'DBI' not found

> library(DBI)
Session restored from your saved work on 2025-Sep-15 11:59:36 UTC (3 hours ago)
> library(DBI)
> library(RPostgres)
> 
```

To upload a new image from dockerfile always needs to delete the container

Typically, Images are distributed through Docker hub

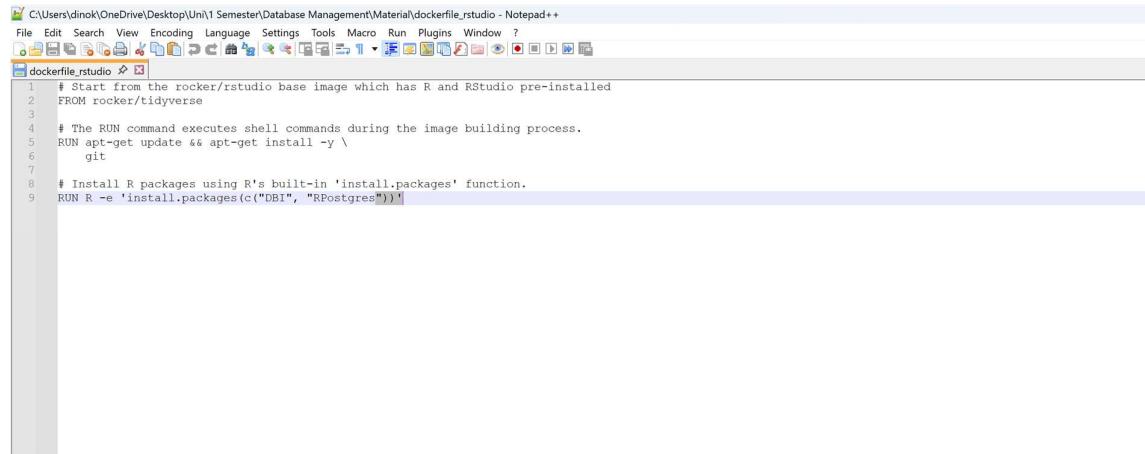
Build an Image from dockerfile as .tar file

First we need to build an Image again if we haven't done so yet if we already have an

Image created skip down to part "Then save image as tar by running "

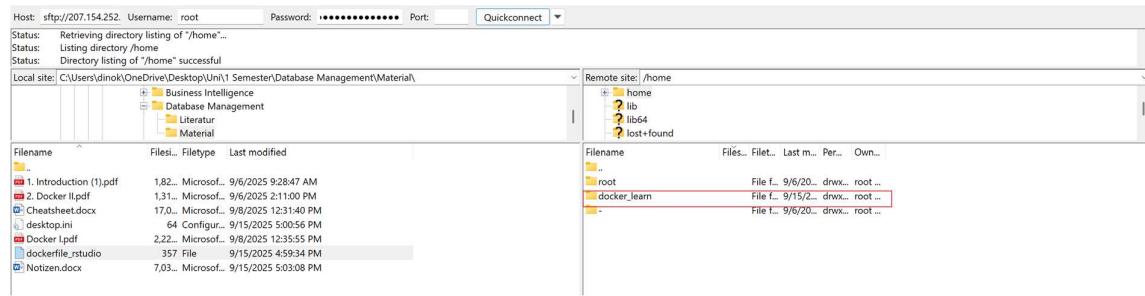
## Build Images from Dockerfile

We have the file dockerfile\_rstudio



```
C:\Users\dnok\OneDrive\Desktop\Uni\1 Semester\Database Management\Material\dockerfile_rstudio - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
dockefile_rstu... dockerfile_rstu...  
1 # Start from the rocker/rstudio base image which has R and RStudio pre-installed  
2 FROM rocker/tidyverse  
3  
4 # The RUN command executes shell commands during the image building process.  
5 RUN apt-get update && apt-get install -y \  
6 git  
7  
8 # Install R packages using R's built-in 'install.packages' function.  
9 RUN R -e 'install.packages(c("DBI", "RPostgres"))'
```

Create a file on remote environment right click mouse



Host: sftp://207.154.252.166 Username: root Password: \*\*\*\*\* Port: Quickconnect

Status: Retrieving directory listing of "/home"...

Status: Listing directory /home

Status: Directory listing of "/home" successful

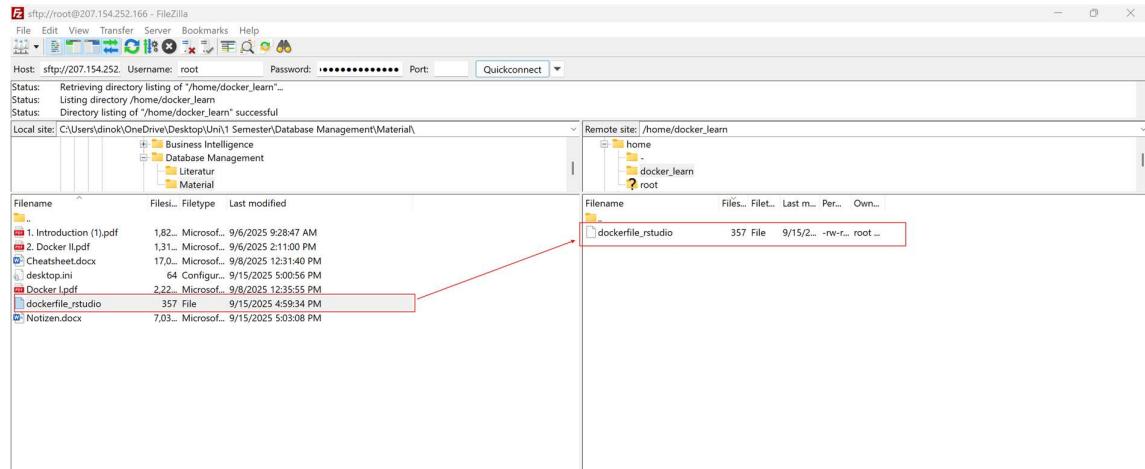
Local site: C:\Users\dnok\OneDrive\Desktop\Uni\1 Semester\Database Management\Material\

Filename	Files...	Filetype	Last modified
1. Introduction (1).pdf	1.82...	Microso...	9/6/2025 9:28:47 AM
2. Docker II.pdf	1.31...	Microso...	9/6/2025 2:11:00 PM
Cheatsheet.docx	17.0...	Microso...	9/8/2025 12:31:40 PM
desktop.ini	64 Configur...	File	9/15/2025 5:00:56 PM
Docker I.pdf	2.22...	Microso...	9/8/2025 12:35:55 PM
dockerfile_studio	357	File	9/15/2025 4:59:34 PM
Notizen.docx	7.03...	Microso...	9/15/2025 5:03:08 PM

Remote site: /home

Filename	Files...	Filet...	Last m...	Per...	Own...
..					
root	File f...	File	9/6/20...	drwx...	root ...
docker_learn	File f...	File	9/15/2...	drwx...	root ...
lost+found	File f...	File	9/6/20...	drwx...	root ...

Drag and drop the file from your local environment to the remote environment



FileZilla - sftp://207.154.252.166 - FileZilla

File Edit View Transfer Server Bookmarks Help

Host: sftp://207.154.252.166 Username: root Password: \*\*\*\*\* Port: Quickconnect

Status: Retrieving directory listing of "/home/docker\_learn"...

Status: Listing directory /home/docker\_learn

Status: Directory listing of "/home/docker\_learn" successful

Local site: C:\Users\dnok\OneDrive\Desktop\Uni\1 Semester\Database Management\Material\

Filename	Files...	Filetype	Last modified
1. Introduction (1).pdf	1.82...	Microso...	9/6/2025 9:28:47 AM
2. Docker II.pdf	1.31...	Microso...	9/6/2025 2:11:00 PM
Cheatsheet.docx	17.0...	Microso...	9/8/2025 12:31:40 PM
desktop.ini	64 Configur...	File	9/15/2025 5:00:56 PM
Docker I.pdf	2.22...	Microso...	9/8/2025 12:35:55 PM
dockerfile_studio	357	File	9/15/2025 4:59:34 PM
Notizen.docx	7.03...	Microso...	9/15/2025 5:03:08 PM

Remote site: /home/docker\_learn

Filename	Files...	Filet...	Last m...	Per...	Own...
..					
root	File f...	File	9/6/20...	drwx...	root ...
docker_learn	File f...	File	9/15/2...	drwx...	root ...
lost+found	File f...	File	9/6/20...	drwx...	root ...

A red arrow points from the "dockerfile\_studio" file in the local site list to the "docker\_learn" directory in the remote site list.

Change the cd to where the file is located

```
root@ubuntu-s-lvcpu-1gb-fral-01:/# ls -l
total 60
-rwxrwxrwx 1 root root 7 Apr 9 12:06 bin -> usr/bin
-rwxr-xr-x 5 root root 4096 Sep 4 06:49 boot
-rwxr-xr-x 18 root root 3920 Sep 3 10:44 dev
-rwxr-xr-x 117 root root 4096 Sep 12 06:23 etc
-rwxr-xr-x 5 root root 4096 Sep 6 12:44 home
-rwxrwxrwx 1 root root 7 Apr 9 12:06 lib -> usr/lib
-rwxrwxrwx 1 root root 9 Apr 9 12:06 lib64 -> usr/lib64
drwx----- 2 root root 16384 Jul 1 19:57 lost+found
drwxr-xr-x 2 root root 4096 Jul 1 19:54 media
drwxr-xr-x 2 root root 4096 Jul 1 19:54 mnt
drwxr-xr-x 4 root root 4096 Sep 3 12:40 opt
dr-xr-xr-x 186 root root 0 Sep 3 10:44 proc
drwx----- 6 root root 4096 Sep 15 08:52 root
drwxr-xr-x 34 root root 1060 Sep 15 15:12 run
-rwxrwxrwx 1 root root 8 Apr 9 12:06 sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Sep 3 10:44 snap
drwxr-xr-x 2 root root 4096 Jul 1 19:54 srv
dr-xr-xr-x 13 root root 0 Sep 6 06:51 sys
drwxrwxrwt 6 root root 120 Sep 15 14:56 tmp
drwxr-xr-x 12 root root 4096 Sep 6 07:35 usr
drwxr-xr-x 13 root root 4096 Sep 3 10:44 var
root@ubuntu-s-lvcpu-1gb-fral-01:/# cd home
root@ubuntu-s-lvcpu-1gb-fral-01:/home# ls -l
total 12
drwxr-xr-x 2 root root 4096 Sep 6 12:44 -
drwxr-xr-x 2 root root 4096 Sep 15 15:05 docker_learn
drwxr-xr-x 3 root root 4096 Sep 6 09:10 root
root@ubuntu-s-lvcpu-1gb-fral-01:/home# cd docker_learn
root@ubuntu-s-lvcpu-1gb-fral-01:/home/docker_learn# 
```

To build the image run:

```
docker image build --tag rstudio:1.0.0 -f dockerfile_rstudio .
```

- docker image build: Initiate building of the image
- --tag rstudio:1.0.0: Gives the name and tag of the image ([name]:[tag])
- -f dockerfile\_rstudio: Gives the name of the Dockerfile
- . : The „space-dot“ indicates that the dockerfile is to be found in your current directory

```
root@ubuntu-s-lvcpu-1gb-fral-01:/home/docker_learn# docker image build --tag rstudio:1.0.0 -f dockerfile_rstudio .
[+] Building 0.8s (7/7) FINISHED
=> [internal] load build definition from dockerfile_rstudio
=> => transferring dockerfile: 404B
=> [internal] load metadata for docker.io/rocker/tidyverse:latest
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/rocker/tidyverse:latest@sha256:6a7e93b235cbdbdf63ca81b11aa7e771ccebd1b54b30dd8f92f560475155119
=> CACHED [2/3] RUN apt-get update && apt-get install -y git
=> CACHED [3/3] RUN R -e 'install.packages(c("DBI", "RPostgres"))'
=> exporting to image
=> => exporting layers
=> => writing image sha256:alcda05557396d579db6f6562db79706352397f24f3ae56da02aae4cd96d54b
=> => naming to docker.io/library/rstudio:1.0.0
```

(when going to R and writing library the packages are not yet installed for that to be the case we need to run the container first)

Then go to the path were the file should be saved in this case docker\_learn

```
cd docker_learn
```

```
root@ubuntu-s-1vcpu-1gb-frai-01:/# ls -l
total 60
lrwxrwxrwx 1 root root 7 Apr 9 12:06 bin -> usr/bin
drwxr-xr-x 5 root root 4096 Sep 4 06:49 boot
drwxr-xr-x 18 root root 3920 Sep 3 10:44 dev
drwxr-xr-x 117 root root 4096 Sep 12 06:23 etc
drwxr-xr-x 5 root root 4096 Sep 6 12:44 home
lrwxrwxrwx 1 root root 7 Apr 9 12:06 lib -> usr/lib
lrwxrwxrwx 1 root root 9 Apr 9 12:06 lib64 -> usr/lib64
drwxr----- 2 root root 16384 Jul 1 19:57 lost+found
drwxr-xr-x 2 root root 4096 Jul 1 19:54 media
drwxr-xr-x 2 root root 4096 Jul 1 19:54 mnt
drwxr-xr-x 4 root root 4096 Sep 3 12:40 opt
dr-xr-xr-x 191 root root 0 Sep 3 10:44 proc
drwxr----- 6 root root 4096 Sep 15 08:52 root
drwxr-xr-x 34 root root 1060 Sep 15 17:50 run
lrwxrwxrwx 1 root root 8 Apr 9 12:06 sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Sep 3 10:44 snap
drwxr-xr-x 2 root root 4096 Jul 1 19:54 srv
dr-xr-xr-x 13 root root 0 Sep 6 06:51 sys
drwxrwxrwt 6 root root 120 Sep 15 17:41 tmp
drwxr-xr-x 12 root root 4096 Sep 6 07:35 usr
drwxr-xr-x 13 root root 4096 Sep 3 10:44 var
root@ubuntu-s-1vcpu-1gb-frai-01:/# cd home
root@ubuntu-s-1vcpu-1gb-frai-01:/home# ls -l
total 12
drwxr-xr-x 2 root root 4096 Sep 6 12:44 -
drwxr-xr-x 2 root root 4096 Sep 15 17:48 docker_learn
drwxr-xr-x 3 root root 4096 Sep 6 09:10 root
root@ubuntu-s-1vcpu-1gb-frai-01:/home# cd docker_learn
```

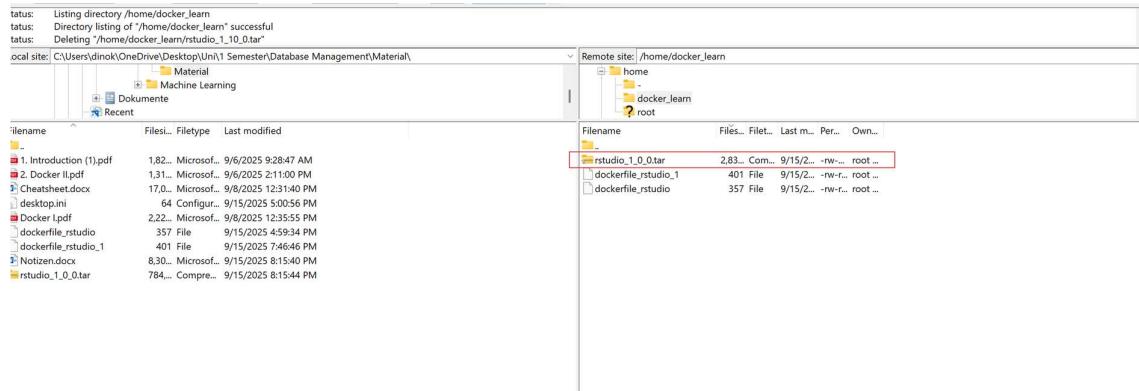
Then save image as tar by running

```
docker save -o rstudio_1_0_0.tar rstudio:1.0.0
```

- rstudio\_1\_0\_0.tar is the name the new file is gonna be given
- rstudio:1.0.0 is the image that will be saved as tar

```
root@ubuntu-s-1vcpu-1gb-frai-01:/home/docker_learn# docker save -o rstudio_1_0_0.tar rstudio:1.0.0
```

Check FileZilla



Then load the image by running

```
docker load -i /path/to/your-image-file.tar
```

- -i = input file (the tarball containing your image).

- If you are already in the same directory as myimage.tar, you don't need the full path — just the filename.

First go to the path where the file is saved in this case docker\_learn

```
cd docker_learn
```

```
root@ubuntu-s-1vcpu-1gb-fra1-01:/# ls -l
total 60
lrwxrwxrwx 1 root root 7 Apr 9 12:06 bin -> usr/bin
drwxr-xr-x 5 root root 4096 Sep 4 06:49 boot
drwxr-xr-x 18 root root 3920 Sep 3 10:44 dev
drwxr-xr-x 117 root root 4096 Sep 12 06:23 etc
drwxr-xr-x 5 root root 4096 Sep 6 12:44 home
lrwxrwxrwx 1 root root 7 Apr 9 12:06 lib -> usr/lib
lrwxrwxrwx 1 root root 9 Apr 9 12:06 lib64 -> usr/lib64
drwx----- 2 root root 16384 Jul 1 19:57 lost+found
drwxr-xr-x 2 root root 4096 Jul 1 19:54 media
drwxr-xr-x 2 root root 4096 Jul 1 19:54 mnt
drwxr-xr-x 4 root root 4096 Sep 3 12:40 opt
dr-xr-xr-x 191 root root 0 Sep 3 10:44 proc
drwx----- 6 root root 4096 Sep 15 08:52 root
drwxr-xr-x 34 root root 1060 Sep 15 17:50 run
lrwxrwxrwx 1 root root 8 Apr 9 12:06 sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Sep 3 10:44 snap
drwxr-xr-x 2 root root 4096 Jul 1 19:54 srv
dr-xr-xr-x 13 root root 0 Sep 6 06:51 sys
drwxrwxrwt 6 root root 120 Sep 15 17:41 tmp
drwxr-xr-x 12 root root 4096 Sep 6 07:35 usr
drwxr-xr-x 13 root root 4096 Sep 3 10:44 var
root@ubuntu-s-1vcpu-1gb-fra1-01:/# cd home
root@ubuntu-s-1vcpu-1gb-fra1-01:/home# ls -l
total 12
drwxr-xr-x 2 root root 4096 Sep 6 12:44 -
drwxr-xr-x 2 root root 4096 Sep 15 17:48 docker_learn
drwxr-xr-x 3 root root 4096 Sep 6 09:10 root
root@ubuntu-s-1vcpu-1gb-fra1-01:/home# cd docker_learn
```

Now run a container with the new image:

First delete the old container so we can create new container with the name r\_studio

```
root@ubuntu-s-1vcpu-1gb-fra1-01:/home/docker_learn# docker stop rstudio
rstudio
root@ubuntu-s-1vcpu-1gb-fra1-01:/home/docker_learn# docker rm rstudio
rstudio
```

Then remove the old image which we was saved as tar

```
root@ubuntu-s-1vcpu-1gb-fra1-01:/home/docker_learn# docker rmi rstudio:1.0.0
Untagged: rstudio:1.0.0
Deleted: sha256:d82d32f5006405d1b6c7c102ff25eb49d54670442a3aad75e2eee51383b5dae0
```

Now load an image from the tar file using

```
docker load -i /path/to/your-image-file.tar
```

- If we are in the input files directory, we do not need to specify the path, only the filename in this case: docker load -i rstudio\_1\_0\_0.tar

```
root@ubuntu-s-1vcpu-1gb-fra1-01:/home/docker_learn# docker load -i rstudio_1_0_0.tar
Loaded image: rstudio:1.0.0
root@ubuntu-s-1vcpu-1gb-fra1-01:/home/docker_learn# docker image ls
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
rstudio          1.0.0    d82d32f50064  11 minutes ago  2.81GB
rstudio          1.0.1    81724e39a678  About an hour ago  2.82GB
postgres         latest   47fa19484031  11 days ago   454MB
hello-world      latest   1b44b5a3e06a  5 weeks ago   10.1kB
rocker/shiny     latest   26a63566fbca  3 months ago  1.64GB
rocker/rstudio   latest   35c3cf057620  3 months ago  2.32GB
```

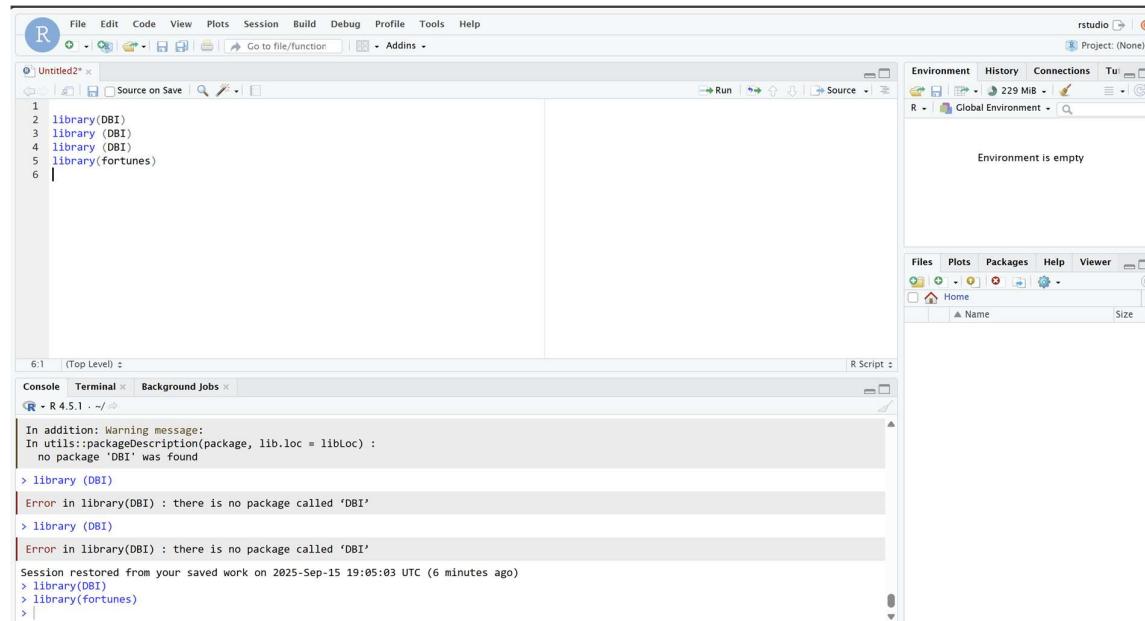
We now build a new rstudio container using the image we just created:

```
docker run -d --network db_r_shiny -p 8787:8787 -e PASSWORD=xxxxxx --name rstudio  
-v rstudio_data:/home/rstudio rstudio:1.0.0
```

- **--network db\_r\_shiny:** This option connects your container to the bridge network **db\_r\_shiny**
- **Rstudio:1.0.0** is the name of the image we created above

```
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker run -d --network db_r_shiny -p 8787:8787 -e PASSWORD=gArField.l3_ll_2001G --name rstudio -v rstudio_data:/home/rstudio rstudio:1.0.0  
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker ps -a  
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES  
013f3b0fe49f        rstudio:1.0.0   "/init"            10 seconds ago    Up 10 seconds    0.0.0.0:8787->8787/tcp, ::1:8787->8787/tcp   rstudio  
1b8cb674df93        postgres           "docker-entrypoint.s..."  9 days ago       Up 9 days          0.0.0.0:5432->5432/tcp, ::1:5432->5432/tcp   postgres  
7e30e982ce93        rocker/shiny        "/init"            9 days ago       Up 9 days          0.0.0.0:3838->3838/tcp, ::1:3838->3838/tcp   rshiny
```

Now check R to see if it worked



## SAVE IMAGE TO .TAR FILE

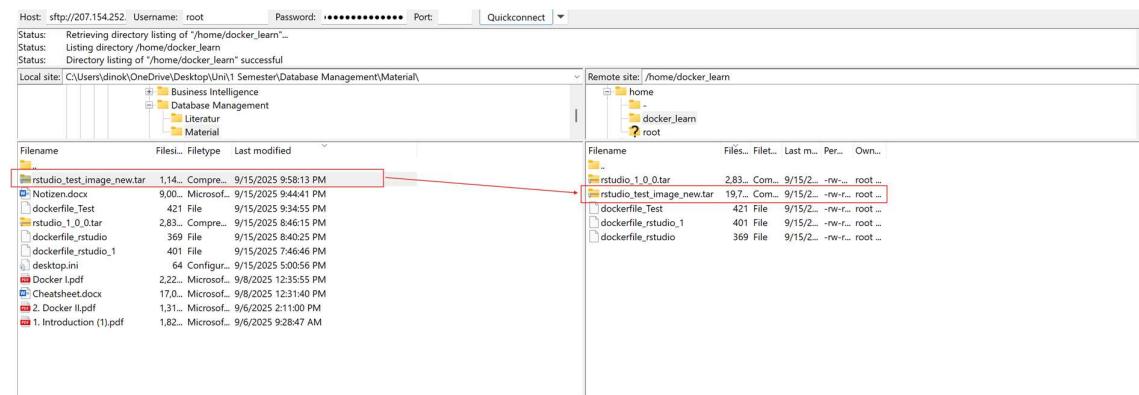
```
docker save -o rstudio_1_0_0.tar rstudio:1.0.0
```

- `rstudio_1_0_0.tar` is the name the new file is gonna be given
- `rstudio:1.0.0` is the image that will be saved as tar

```
root@ubuntu-s-lvcpu-lgb-fral-01:/home/docker_learn# docker save -o rstudio_1_0_0.tar rstudio:1.0.0
```

## Loading Data from extern .Tar file (extern docker image)

First download the .tar and drag and drop it from local machine to remote machine



Now load an Image from the tar file using  
`docker load -i /path/to/your-image-file.tar`

- If we are in the input files directory, we do not need to specify the path, only the filename in this case: `docker load -i rstudio_test_image_new.tar`

The image will be created automatically `rstudio_test_image`

```
root@ubuntu-s-lvcpu-lgb-fral-01:/home/docker_learn# docker load -i rstudio_test_image_new.tar
Loaded image: rstudio_test_image:latest
root@ubuntu-s-lvcpu-lgb-fral-01:/home/docker_learn# docker image ls
REPOSITORY          TAG       IMAGE ID    CREATED             SIZE
rstudio_test_image  latest    491805f23105  41 minutes ago   2.83GB
rstudio             1.0.0    d82d32f50064  About an hour ago  2.81GB
rstudio             1.0.1    81724e39a678  2 hours ago      2.82GB
postgres            latest   47fa19484031  11 days ago      454MB
hello-world         latest   1b44b5a3e06a  5 weeks ago      10.1kB
rocker/shiny        latest   26a63566fbca  3 months ago     1.64GB
rocker/rstudio      latest   35c3cf057620  3 months ago     2.32GB
```

This does not install packages or anything in R for that we need to restart the container

Delete container by:

```
docker rm -f rstudio
```

- `-f` is a flag that force the container to be stopped if it is running

```
root@ubuntu-s-lvcpu-lgb-fral-01:/home/docker_learn# docker rm -f rstudio
```

```
docker run -d --network db_r_shiny -p 8787:8787 -e PASSWORD=xxxxxx --name rstudio  
-v rstudio_data:/home/rstudio rstudio_test_image
```

- --network db\_r\_shiny: This option connects your container to the bridge network db\_r\_shiny
- rstudio\_test\_image is the name of the image we created above

```
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker run -d --network db_r_shiny -p 8787:8787 -e PASSWORD=gArfield_13_11_2001G --name  
rstudio -v rstudio_data:/home/rstudio rstudio test_image  
986726c27a4e3e4f937639e51e149c37d7e8137dfcea393ab2894cd7bfdfe41a
```

## Check R

The screenshot shows the RStudio IDE interface. The top bar includes tabs for 'Untitled2\*', 'Source on Save', 'Run', and 'Source'. A message in the status bar indicates a package is required but not installed: 'Package fortunes required but is not installed. [Install](#) Don't Show Again'. The main area contains an R script with the following code:

```
1  
2  
3  
4 library(DBI)  
5 library(RPostgres)  
6 library(tidyverse)  
7 library(httr2)  
8 library(shiny)  
9 library(jsonlite)  
10 library(janitor)  
11 library(skimr)
```

The bottom panel shows the R console output:

```
4:1 | (Top Level) | R Script |  
Console Terminal Background Jobs  
R 4.5.1 . ~/|> library(jsonlite)  
  
Attaching package: 'jsonlite'  
The following object is masked from 'package:shiny':  
validate  
The following object is masked from 'package:purrr':  
flatten  
> library(janitor)  
> library(skimr)  
> |
```

### Dangling:

- When building an image with a name and tag similar to an existing image (e.g. rstudio:1.0.0), the old image will be "dangling".
- A dangling image is an image that has no tag, and where no containers are referencing (using) the image
- To remove dangling image:  
docker image prune

```
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# docker image prune
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
root@ubuntu-s-1vcpu-1gb-fral-01:/home/docker_learn# 
```

➔ goal to free up disk space

docker switch passcode of postgres:

Option 1: Reset the password without removing data

If you want to keep your data, you can change the password from inside the container:

    docker exec -it <container\_name> psql -U postgres

Then in the Postgres prompt:

    ALTER USER postgres WITH PASSWORD 'xxxxxxxxxxxx';

    \q