

Docker Containers

Business Science

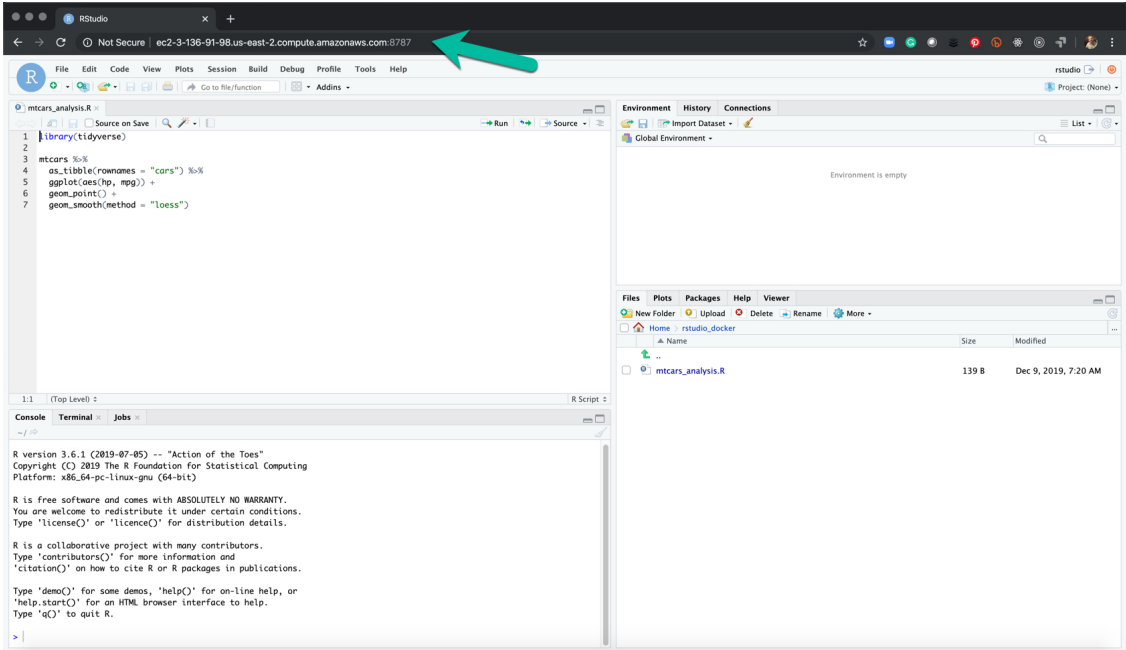
11/26/2019

Contents

| | |
|---|----------|
| Docker Containers | 1 |
| Prerequisites | 2 |
| Docker Run | 2 |
| Development | 2 |
| Deployment | 5 |
| Example Workflow for Deploying RStudio Server | 5 |
| Wrapup | 9 |

Docker Containers

This document covers the using **docker container** to run docker containerized environments on your EC2 Server. By the end of this, you will be able to deploy RStudio IDE in the cloud on AWS.



Prerequisites

Use `docker image ls` to list docker images. These are the images on our local server that we can run containers from. Make sure that you have:

- `rocker/shiny-verse` - Image with **Shiny** Server + **tidyverse** packages that runs on port 3838
- `rocker/tidyverse` - Image with **RStudio** Server (IDE) + **tidyverse** packages that runs on port 8787

Docker Run

The command `docker run` is used to run docker containers. This document is devoted to the `run` container command.

Use `docker container run --help` to see the list of the `run` options. Important Options:

- `--rm` and `-d` - for development and deployment
 - `--rm` removes the container after the container session is completed and quit with `Ctrl + C`.
 - `-d` runs the docker container in the background, which is how deployed applications will run on your EC2 Server.
- `-p <host_port>:<container_port>` - Maps a host port (e.g. server's port 80) to a container port (e.g. 3838, which runs Shiny Server)
- `-v` - Mount a Volume. Connects a virtual volume (directory where files are stored) to a local volume on your EC2 Server.

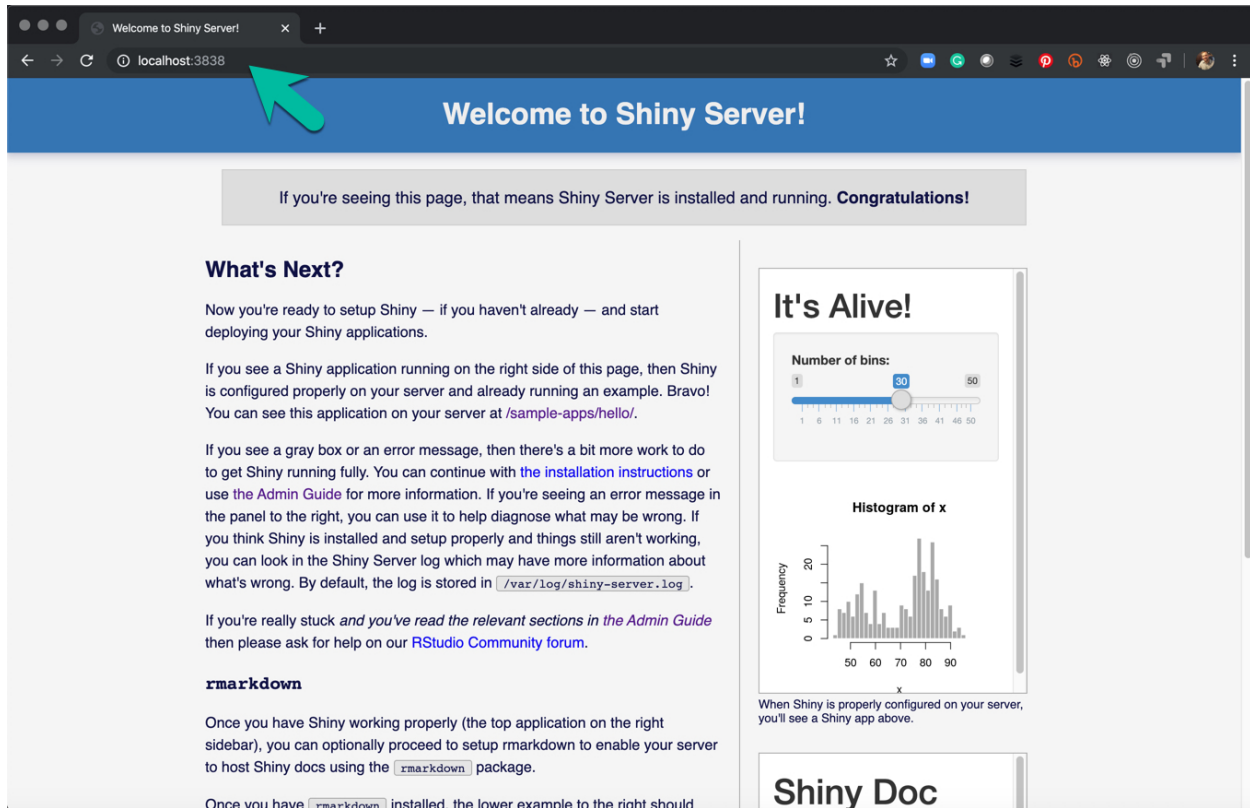
Development

When building apps, we'll be using `docker run --rm ...` where `--rm` kills the container as soon as we're finished.

Shiny Server - `rocker/shiny-verse`

Run Shiny Server with the `rocker/shiny-verse` container in development mode (`--rm` runs interactively) and map virtual port 3838 to local port 3838:

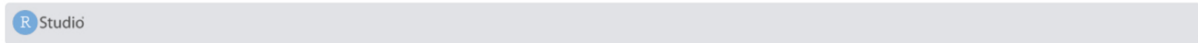
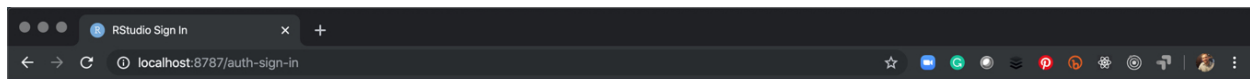
1. `sudo docker run --rm -p 3838:3838 rocker/shiny-verse`
2. Navigate to the port 3838 of your IP. If running on an EC2 server, use your `public_IPV4:3838`. If running locally (on your PC or Mac), you can use `localhost:3838` to open shiny server.
3. Congrats - You now have **Shiny** Server running.
4. Use `Ctrl + C` to kill the shiny server and shutdown the container.



RStudio Server - rocker/tidyverse

Run RStudio Server with the **rocker/tidyverse** container in development mode (`--rm` runs interactively) and map virtual port 8787 to local port 8787:

1. `sudo docker run -e PASSWORD=your_password --rm -p 8787:8787 rocker/tidyverse` - replace "your_password" with a better password
2. Navigate to the port 8787 of your IP. If running on an EC2 server, use your `public_IPV4:8787`. If running locally (on your PC or Mac), you can use `localhost:8787` to open RStudio Server IDE.
3. **Username & Password** - Use `username = "rstudio"` and `password = the password set using the environment variable -e PASSWORD=your_password`.
4. Congrats - You now have RStudio Server running.
5. Use `Ctrl + C` in the Terminal to kill the RStudio server and shutdown the container.



Sign in to RStudio

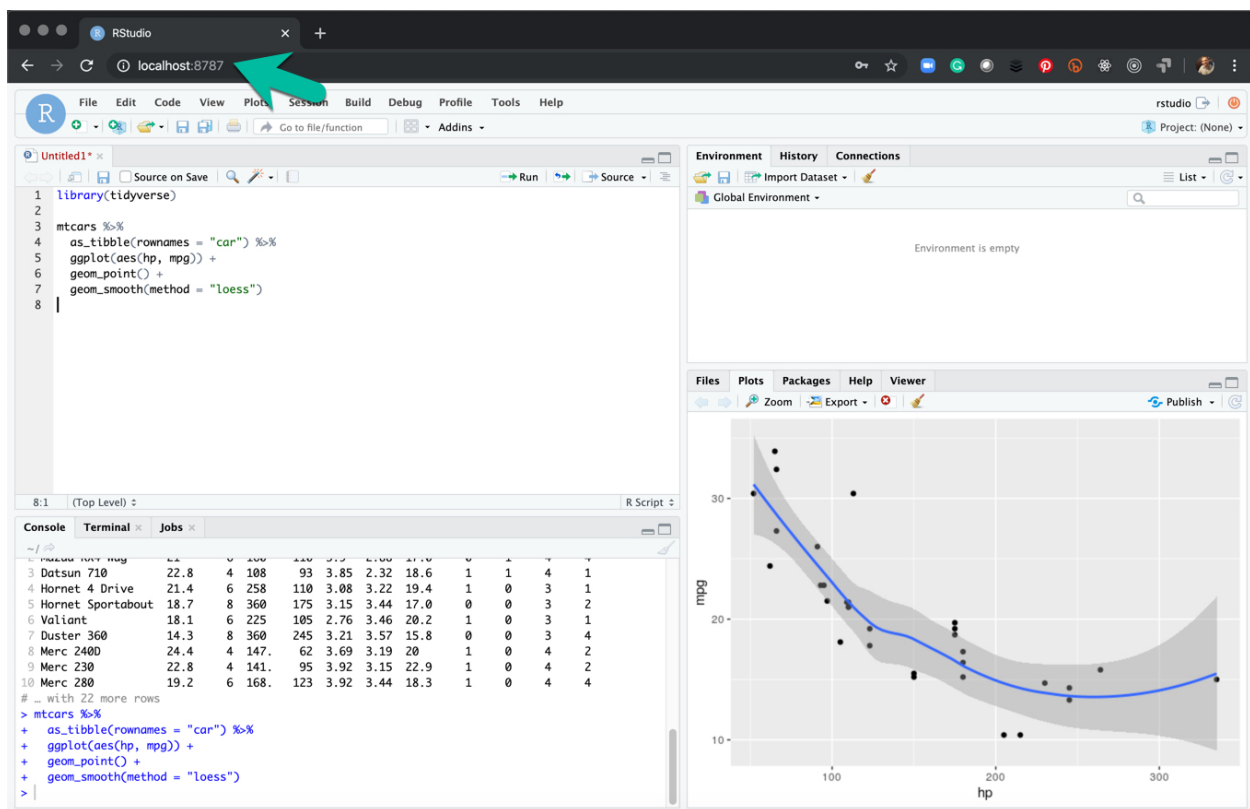
Username:

Password:

☐ Stay signed in

rstudio (arrow pointing to Username field)

your_password (arrow pointing to Password field)



Deployment

Running in Detached Mode

Running in “detach” mode (`docker container run -d`) is necessary to **run docker containers in production**. The advantage is we can have multiple docker containers running and running in detach mode does not tie up the terminal.

1. `sudo docker run -e PASSWORD=your_password -d -p 8787:8787 rocker/tidyverse` - The docker container is run in “detached” mode using `-d`, which runs the process in the background - This is exactly what is needed in production since the server’s
2. `docker container ls` - List the active containers. Note that your server can have multiple containers running concurrently.
3. `docker container stop [CONTAINER ID OR NAME]` - To stop a container.

Linking Volumes

You may notice that Docker Containers don’t save any files after the container is stopped. This is an important concept because Docker Containers are *virtual environments* - Files, directories, and any software downloaded inside the container environment does not persist.

For files and directories that (1) exist on the host server but need to be used in the container session or (2) need to be saved from the container session to the host server, we can connect the two environments by *linking volumes*.

Format to Link Volumes: `-v <host_dir>:<container_dir>`

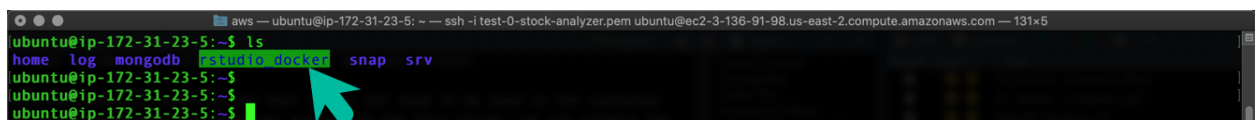
Example Workflow for Deploying RStudio Server

This is an example of deploying an RStudio Server on your AWS EC2 Server. The deployed version will:

- Be developed in development mode using `--rm`
- Have volumes linked to a directory on the EC2 Server `-v`
- Be deployed in detached mode using `-d`

Step 1 - Create a Directory

In your AWS EC2 Server, navigate to `/home/ubuntu` and `mkdir rstudio_docker`. This creates a directory that we will link to the container.



Step 2 - Change Permission

`sudo chmod 777 rstudio_docker/` - This ensures that the RStudio Server will be able to write to the new directory.

```
aws — ubuntu@ip-172-31-23-5: ~ — ssh -i test-0-stock-... rper.pem ubuntu@ec2-3-136-91-98.us-east-2.compute.amazonaws.com — 131x10
ubuntu@ip-172-31-23-5:~$ sudo chmod 777 rstudio_docker/
ubuntu@ip-172-31-23-5:~$ ls -lh
total 24K
drwxr-xr-x 3 root root 4.0K Nov 4 22:01 home
drwxr-xr-x 3 root root 4.0K Nov 4 22:18 log
drwxrwxr-x 3 ubuntu ubuntu 4.0K Nov 10 01:57 mongodb
drwxrwxrwx 2 ubuntu ubuntu 4.0K Dec 9 11:37 rstudio_docker
drwxr-xr-x 4 ubuntu ubuntu 4.0K Dec 9 11:36 snap
drwxrwxr-x 7 ubuntu ubuntu 4.0K Nov 26 16:58 srv
ubuntu@ip-172-31-23-5:~$
```

Permissions changed to **rw-rw-rw-**

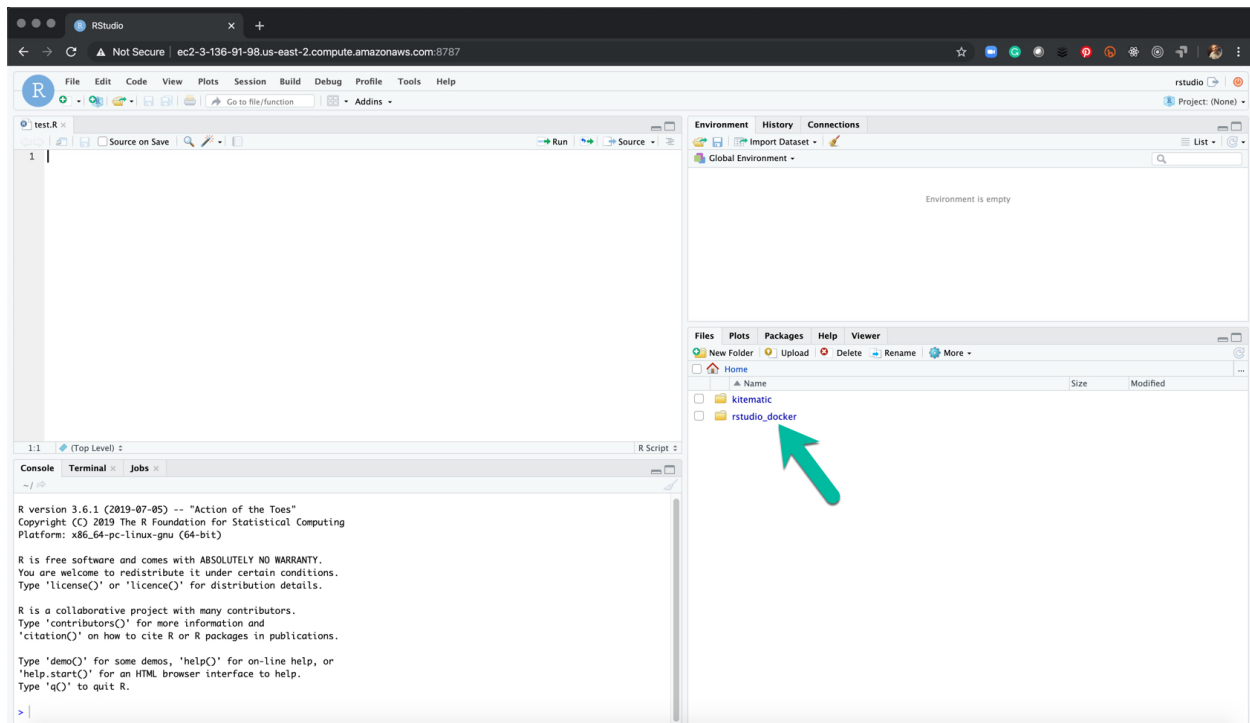
Step 3 - Run Docker Container with Linked Volumes

Run the following code. The code runs RStudio in development mode linking the your EC2 server's "rstudio_docker" directory located at: /home/ubuntu/rstudio_docker to your RStudio IDE running in a container at: home/rstudio/rstudio_docker.

```
sudo docker run -e PASSWORD=your_password --rm -p 8787:8787 \
-v /home/ubuntu/rstudio_docker:/home/rstudio/rstudio_docker rocker/tidyverse
```

Step 4 - Navigate to Your Server's RStudio Session

You should now see a folder in your "Files" pane inside of RStudio.



Step 5 - Make a File

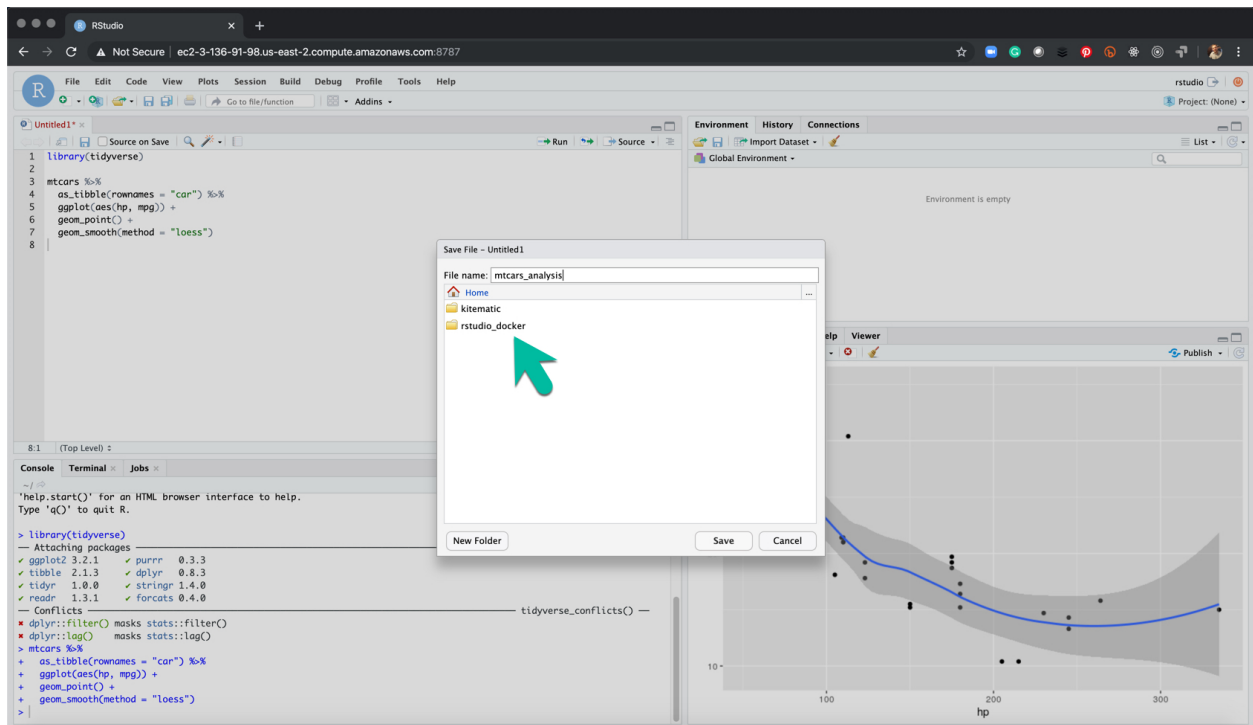
Make a quick mtcars analysis.

```
# File: mtcars_analysis.R
library(tidyverse)

mtcars %>%
  as_tibble(rownames = "cars") %>%
  ggplot(aes(hp, mpg)) +
  geom_point() +
  geom_smooth(method = "loess")
```

Step 6 - Save the Analysis in Your rstudio_docker Directory

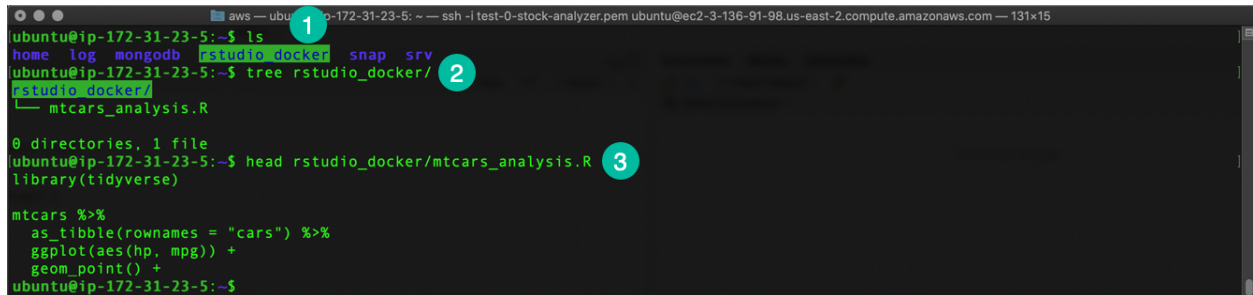
Save the analysis in the “rstudio_docker” folder. Note - You will get an ERROR if your permission was not correctly setup in Step 2.



Step 7 - Check Your File on EC2

Once finished creating an analysis file, **Ctrl + C** to exit the RStudio Session. Then verify the file was created properly by:

1. `ls` to list the files
2. `tree rstudio_docker` to run tree inside the “rstudio docker” folder
3. `head rstudio_docker/mtcars_analysis.R` to check the first 6 lines of the analysis file.



```
aws — ubuntu@ip-172-31-23-5: ~ — ssh -i test-0-stock-analyzer.pem ubuntu@ec2-3-136-91-98.us-east-2.compute.amazonaws.com — 131x15
ubuntu@ip-172-31-23-5:~$ ls
home  log  mongodb  rstudio_docker  snap  srv
ubuntu@ip-172-31-23-5:~$ tree rstudio_docker/
rstudio_docker/
├── mtcars_analysis.R
0 directories, 1 file
ubuntu@ip-172-31-23-5:~$ head rstudio_docker/mtcars_analysis.R
library(tidyverse)

mtcars %>%
  as_tibble(rownames = "cars") %>%
  ggplot(aes(hp, mpg)) +
  geom_point() +
ubuntu@ip-172-31-23-5:~$
```

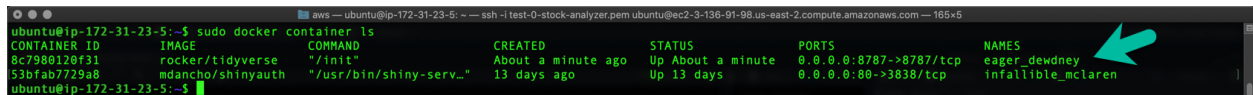
Step 8 - Run in Detached Mode

Once the volumes are linking properly, deploy the RStudio Server on port 8787 with the linked volume.

```
sudo docker run -e PASSWORD=your_password -d -p 8787:8787 \
-v /home/ubuntu/rstudio_docker:/home/rstudio/rstudio_docker rocker/tidyverse
```

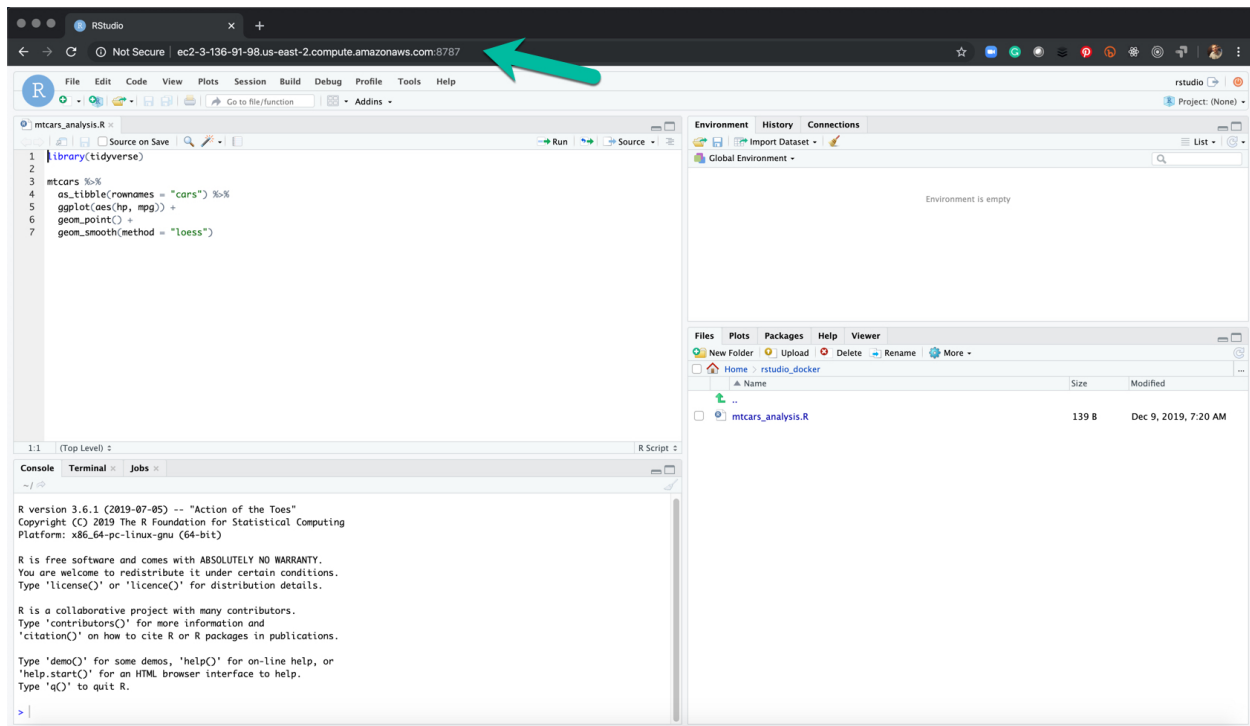
Step 9 - Verify the Deployed Container

Run `sudo docker container ls` to verify the deployed container is running.



```
aws — ubuntu@ip-172-31-23-5: ~ — ssh -i test-0-stock-analyzer.pem ubuntu@ec2-3-136-91-98.us-east-2.compute.amazonaws.com — 165x5
ubuntu@ip-172-31-23-5:~$ sudo docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
8c7980120f31       rocker/tidyverse    "/usr/bin/shiny-serv..."  About a minute ago  Up About a minute  0.0.0.0:8787->8787/tcp  eager_dewdney
53bfab7729a8       mdancho/shinyauth  "/usr/bin/shiny-serv..."  13 days ago        Up 13 days         0.0.0.0:80->3838/tcp   infallible_mclaren
```

Go to your EC2 Server's port 8787 to see the RStudio Server hosted in the cloud.



Step 10 - Adding a User Name

You can add a user name by adding a second environment variable and changing the path of the linked volumes.

1. `sudo docker container stop` - Use `stop` to stop the existing container.
2. `-e USER=your_user_name` - Update your user name for the RStudio Login.
3. `-v /home/ubuntu/rstudio_docker:/home/your_user_name/rstudio_docker` - Change the volume links to match your updated user name.

```
sudo docker run -e USER=your_user_name -e PASSWORD=your_password -d -p 8787:8787 \
-v /home/ubuntu/rstudio_docker:/home/your_user_name/rstudio_docker rocker/tidyverse
```

Wrapup

You've successfully launched your own RStudio Server in the AWS Cloud. We'll use this later for making DockerFiles.

If at any time you want to **shutdown**, just use `docker container ls` to list your active containers. Then use `docker container stop [CONTAINER ID]`.