

Setting up Docker to Work with Local Files on Windows

This is my attempt at helping my fellow Windows users to get Docker up-and-running. It might seem a little duct-taped together – that's because it pretty much is. But as long as it works, I figure this setup should be fine for our purposes.

Feel free to reach out to me (kaleb.erickson@gmail.com) if you have further questions or need help. I am by no means an expert, but I'll do what I can to help you get things working.

If you've already got Docker and Kitematic working on your machine, then you can skip to Step 4 for instructions on how to get the local files working. Just make sure that you have Kitematic installed and working on your computer.

Step 1: Download Docker Toolbox

For some reason, the normal Docker application only works on Windows 10 Enterprise edition. So, if you don't have that, you'll have to go with the legacy solution, which is Docker Toolbox.

Here is the download link: <https://download.docker.com/win/stable/DockerToolbox.exe>

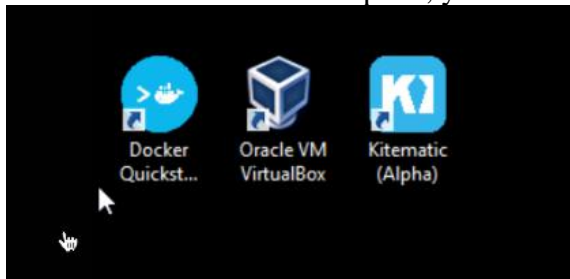
Step 2: Install Docker Toolbox

Once you've got that downloaded, you'll want to head over to this tutorial on how to install Docker Toolbox: https://docs.docker.com/toolbox/toolbox_install_windows/

Be sure to follow the directions here, especially the part about installing the VirtualBox. If you don't have it installed, you will want it here. Otherwise, make sure you uncheck it.

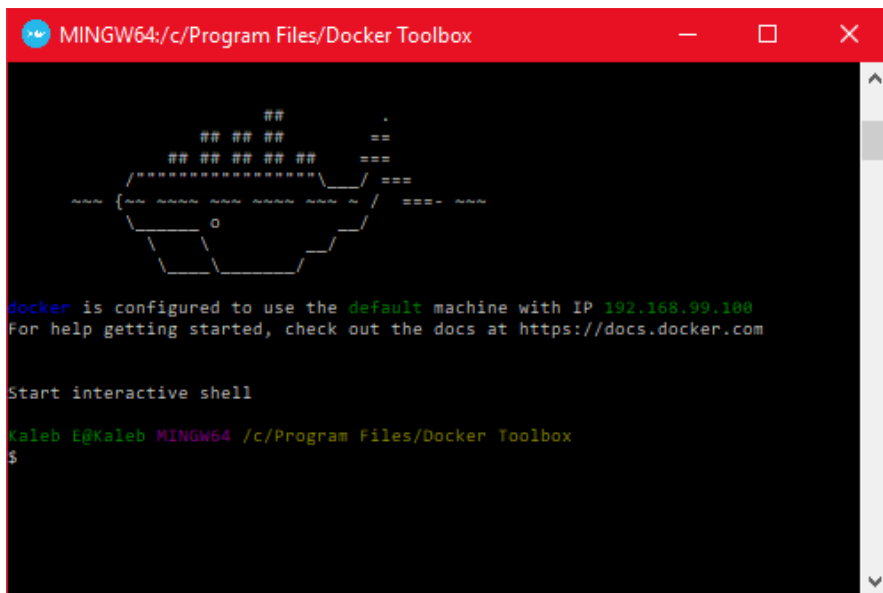
Step 3: Get things running

Once the installation is complete, you should have the following three programs on your computer:

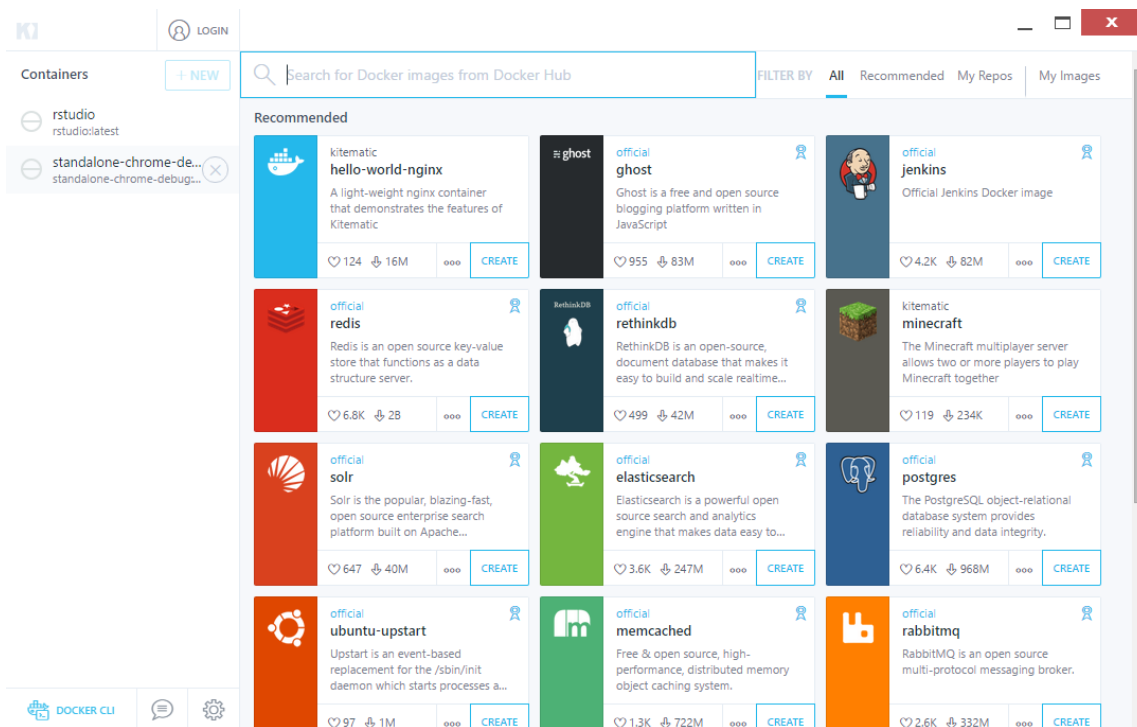


Docker Quickstart and Kitematic are going to be our main focus here. The VirtualBox is necessary to make all of this work, but you don't need to worry about it.

Open up Docker Quickstart and it should run a bunch of stuff automatically to get things set up. In the future, when you open it up, you should be greeted by a screen like this:



Now open up Kitematic. This program is a GUI for Docker and is very helpful in managing your containers and getting them to work properly. It should greet you with something that looks like this:

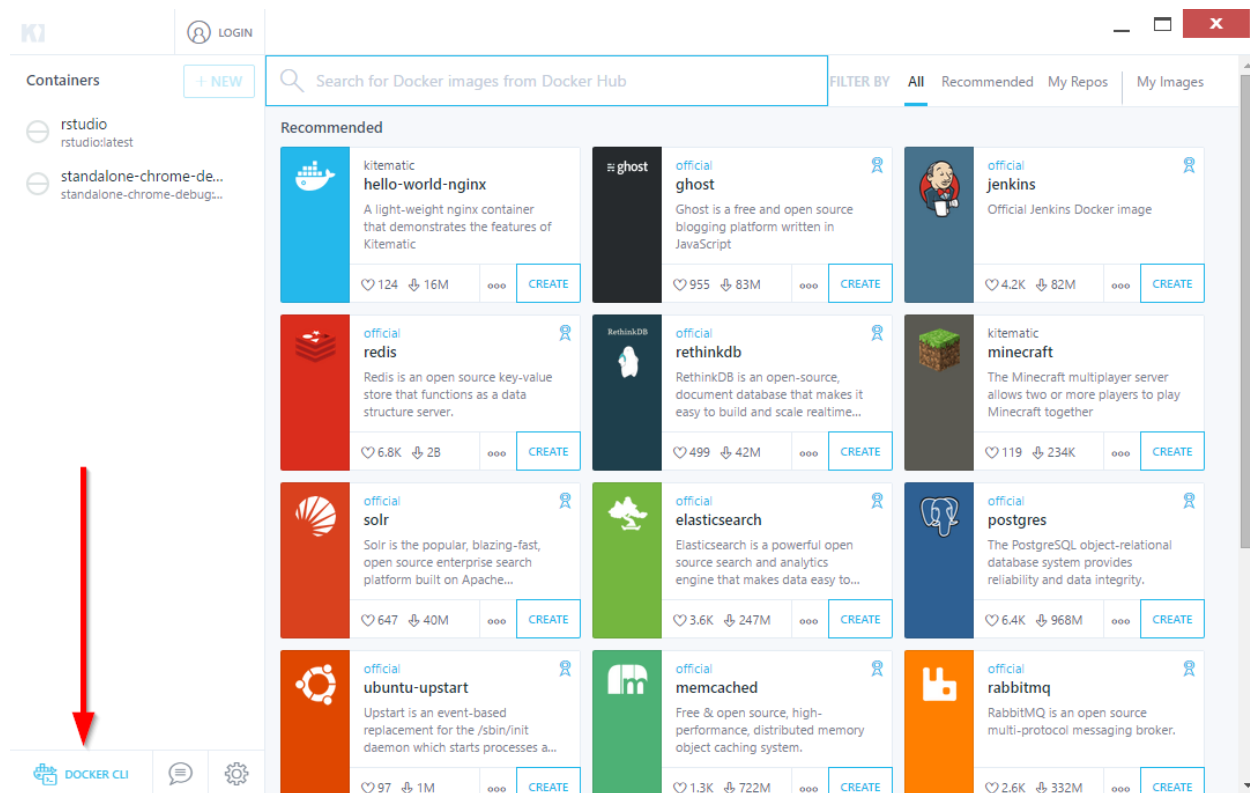


These are all options for Docker containers that you can set up. You can also search for others in the search bar at the top of the application.

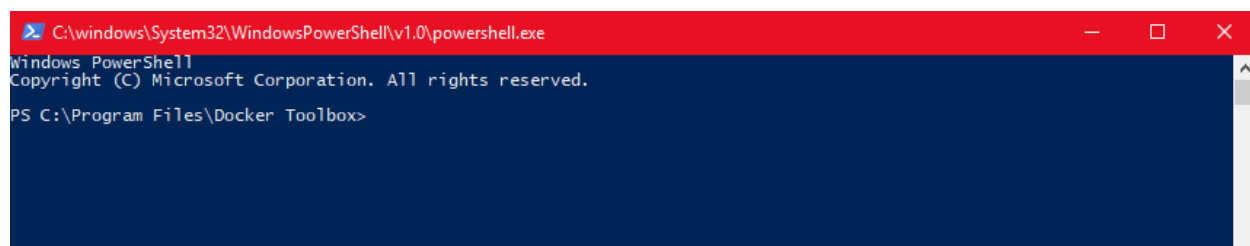
Any containers that are currently running will be listed on the left of the application. You can see that I currently have two: one called rstudio and the other standalone-chrome-de...

Step 4: Create Our Own Container

There are lots of options for where we could write these Docker commands. For this, we are going to start in Kitematic. Click on the button labeled DOCKER CLI in the bottom left corner:



This should open up an instance of Powershell. It will look something like this:



We can run Docker commands here! The following code will set up a container that runs Rstudio and also designates a local file path to look for files:

```
docker run -d --rm --name rstud -e PASSWORD=class -p 8787:8787 -v /Users:/rstudio rocker/rstudio
```

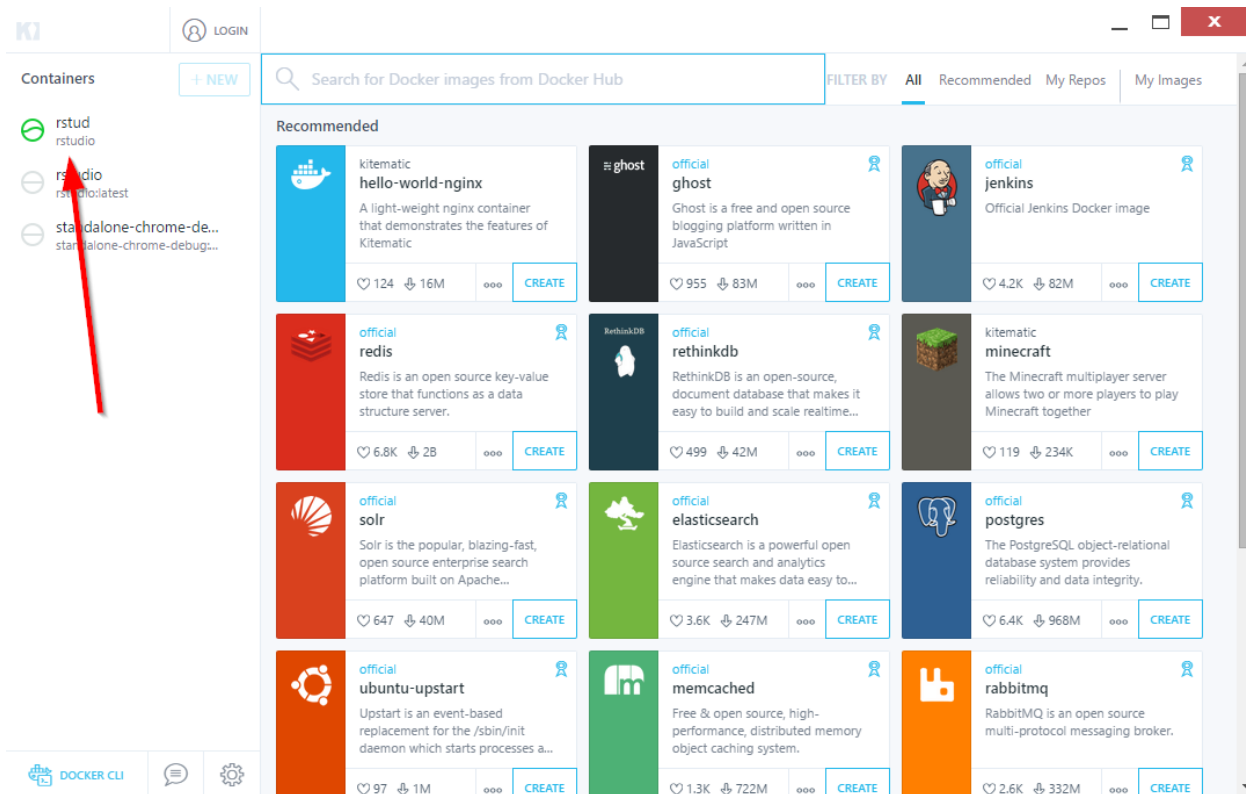
When you run it, you should see a string of numbers and letters below. This means that it worked properly:

```
C:\windows\System32\WindowsPowerShell\v1.0\powershell.exe
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

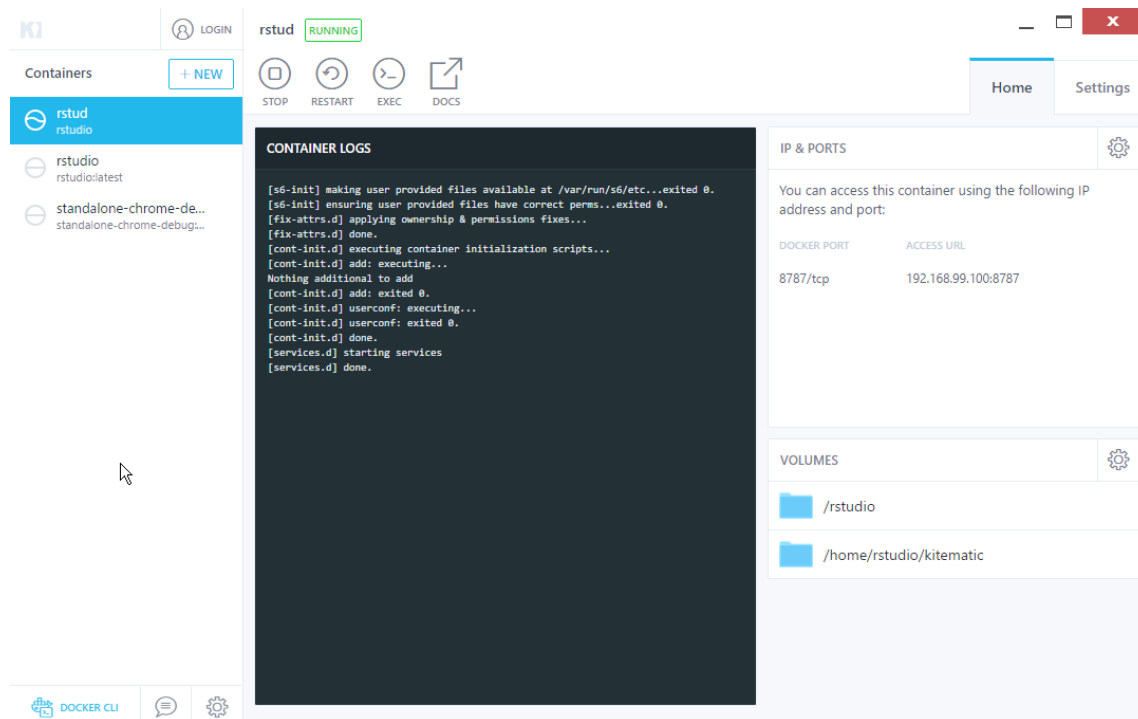
PS C:\Program Files\Docker Toolbox> docker run -d --rm --name rstud -e PASSWORD=class -p 8787:8787 -v /Users:/rstudio rocker/rstudio
9ceaf1919f03b50e5419724173c3210b4c1e17dc734a4c5069f854de509cc3d4
PS C:\Program Files\Docker Toolbox>
```

Step 5: Finishing up with Kitematic

Now, we need to go back to Kitematic. Since we started a new container, it should appear on the left bar with a green circle next to it. If it doesn't appear, you might need to close Kitematic and start it back up again.



Note that it's called "rstud", which is what we named the container in that Docker command. Click on this container to view some details about it. You should see something like this:



The screenshot shows the Kitematic application interface. On the left, a sidebar lists containers: 'rstud' (selected), 'rstudio', 'rstudiolatest', and 'standalone-chrome-de...'. The main area displays details for the 'rstud' container, which is in a 'RUNNING' state. The 'CONTAINER LOGS' section shows the output of the container's initialization scripts. The 'IP & PORTS' section indicates that the container can be accessed via Docker port 8787/tcp at the access URL 192.168.99.100:8787. The 'VOLUMES' section lists two volumes: '/rstudio' and '/home/rstudio/kitematic'.

Containers: + NEW

STOP RESTART EXEC DOCS

Home Settings

CONTAINER LOGS

```
[s6-init] making user provided files available at /var/run/s6/etc...exited 0.
[s6-init] ensuring user provided files have correct perms...exited 0.
[fix-attrs.d] applying ownership & permissions fixes...
[fix-attrs.d] done.
[cont-init.d] executing container initialization scripts...
[cont-init.d] add: executing...
Nothing additional to add
[cont-init.d] add: exited 0.
[cont-init.d] userconf: executing...
[cont-init.d] userconf: exited 0.
[cont-init.d] done.
[services.d] starting services
[services.d] done.
```

IP & PORTS

You can access this container using the following IP address and port:

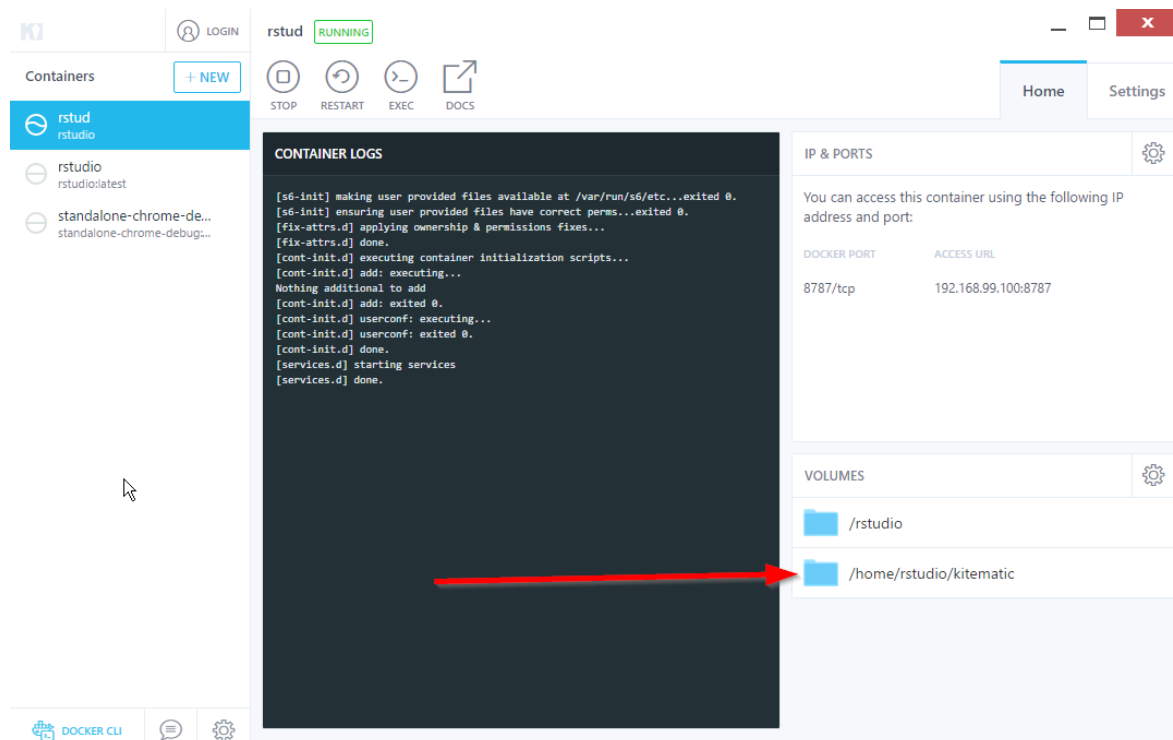
DOCKER PORT	ACCESS URL
8787/tcp	192.168.99.100:8787

VOLUMES

/rstudio
/home/rstudio/kitematic

There are two important things here: 1. The IP & Ports section and 2. The Volumes section.

First, you'll want to click on the file with /home/rstudio/kitematic, as pointed out below:



This screenshot is identical to the previous one, but with a red arrow pointing to the '/home/rstudio/kitematic' volume entry in the 'VOLUMES' section.

Containers: + NEW

STOP RESTART EXEC DOCS

Home Settings

CONTAINER LOGS

```
[s6-init] making user provided files available at /var/run/s6/etc...exited 0.
[s6-init] ensuring user provided files have correct perms...exited 0.
[fix-attrs.d] applying ownership & permissions fixes...
[fix-attrs.d] done.
[cont-init.d] executing container initialization scripts...
[cont-init.d] add: executing...
Nothing additional to add
[cont-init.d] add: exited 0.
[cont-init.d] userconf: executing...
[cont-init.d] userconf: exited 0.
[cont-init.d] done.
[services.d] starting services
[services.d] done.
```

IP & PORTS

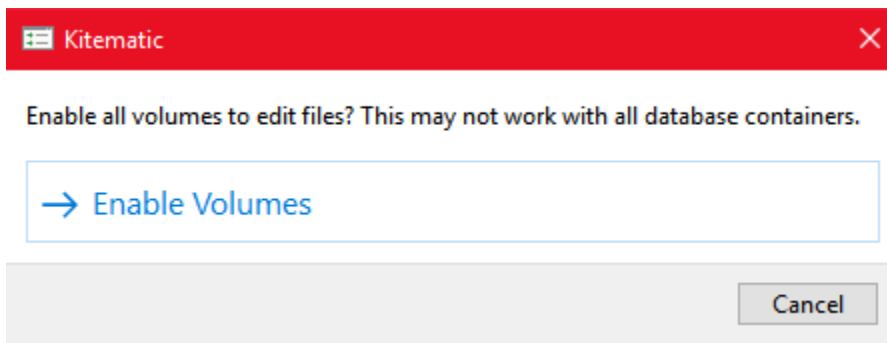
You can access this container using the following IP address and port:

DOCKER PORT	ACCESS URL
8787/tcp	192.168.99.100:8787

VOLUMES

/rstudio
/home/rstudio/kitematic

A popup will ask you if you want to enable volumes to edit files. Click to enable volumes. As far as I can tell, this is what allows Docker to communicate with your local files:

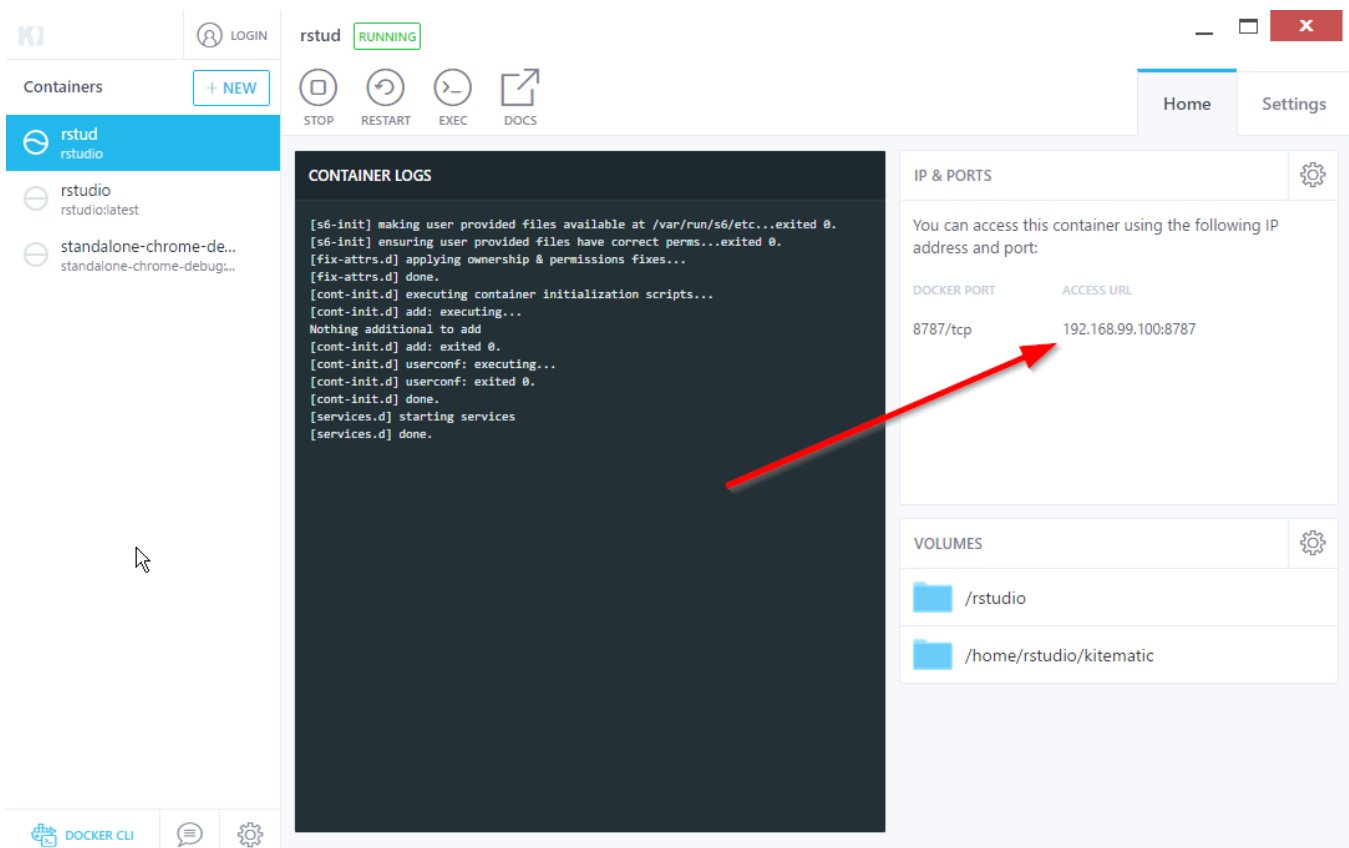


This will open up a window showing you where the local files can be found. Minimize this for now, but we'll want to refer to it later.

Step 6: Going into RStudio

Now that everything is set up properly, we can open up RStudio in a browser and ensure that the local files are working.

Back to Kitematic, you can find the address to put in your browser here:



Just copy that and put it into the address bar and it will take you to the login page.

Username: rstudio
Password: class

Once you're in RStudio, you need to run the following code to make sure that it is pointing at the correct directory:

```
setwd("~/kitematic")
```

And we should be up and running! To double check, run the following code in RStudio:

```
f <- c(1,2,3)  
write.csv(f, 'trial.csv')
```

Once you run it, pull up the local windows file that we minimized earlier. Navigate into the kitematic folder and you should see your 'trial.csv' file waiting for you.