
Docker as a Virtualization Platform for Cloud and Edge

Difficulty Level: Medium

Objectives

Gain general understanding of container-based virtualization concepts and obtain practical experience with using Docker to create and manage containers for easy installation and sharing of applications on different platforms and operating systems.

Achievements

- ☐ Learning the basics of virtualization and container technologies
- ☐ Getting familiar with Docker as a container-based virtualization tool
- ☐ Installing and running Docker on a host PC
- ☐ Fetching an application as a bundled Docker image and running it as a virtual machine (container) in Docker
- ☐ Monitoring and managing the available images and containers in Docker

Getting started with Docker

Docker¹ is a highly popular container-based virtualization tool, which allows you to run the same application or virtual machine on any operating system without having to deal with the libraries and other dependencies required by the application. This is achieved by bundling the application with a lightweight runtime and all required dependencies in a so-called “container” image. The container image can be easily shared across different systems. Single or multiple instances of this image can be started as a new “virtual machine” (container instance) running in the Docker environment.

Step 1: Install and Run Docker Desktop on your PC

Simply download and install the desktop application for your operating system (OS) from the Docker website: <https://www.docker.com/products/docker-desktop>. In this module we will focus on a Windows PC, but Linux and Mac versions of Docker are also available and the documentation here applies to those as well after successfully installing the Docker Desktop application on the target OS.

Once the installation is completed, you can start the Docker application by clicking on the Windows Start Menu and start typing Docker. Then the “Docker Desktop” application will appear, which you should click on (see Figure 1).

¹ <https://www.docker.com/>

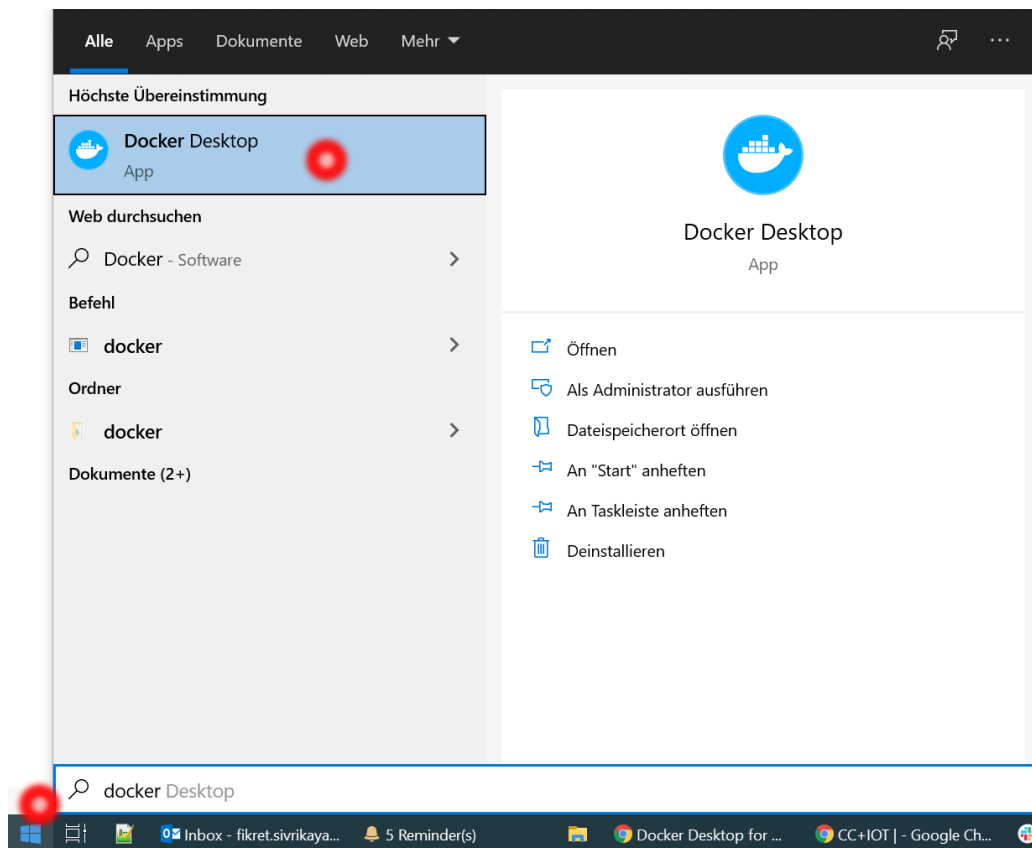


Figure 1. Starting Docker Desktop application on Windows after the installation

Step 2: Familiarizing with the User Interface

When the Docker Desktop application launches, it may start as a minimized tray icon on Windows. If this is the case, you can open up its main application window by double clicking on its icon in the Windows tray area (usually in the right bottom corner of your screen, next to the time on the task bar).

The main application window has a simple interface, which looks something like in Figure 2. Note that depending on the current version of Docker, this may change.

On the left panel of Docker window, we see three tabs / pages:

- Containers / Apps
- Images
- Dev Environments

We will look into the first two in more detail next, while the new third option remains outside the scope of this lab session.

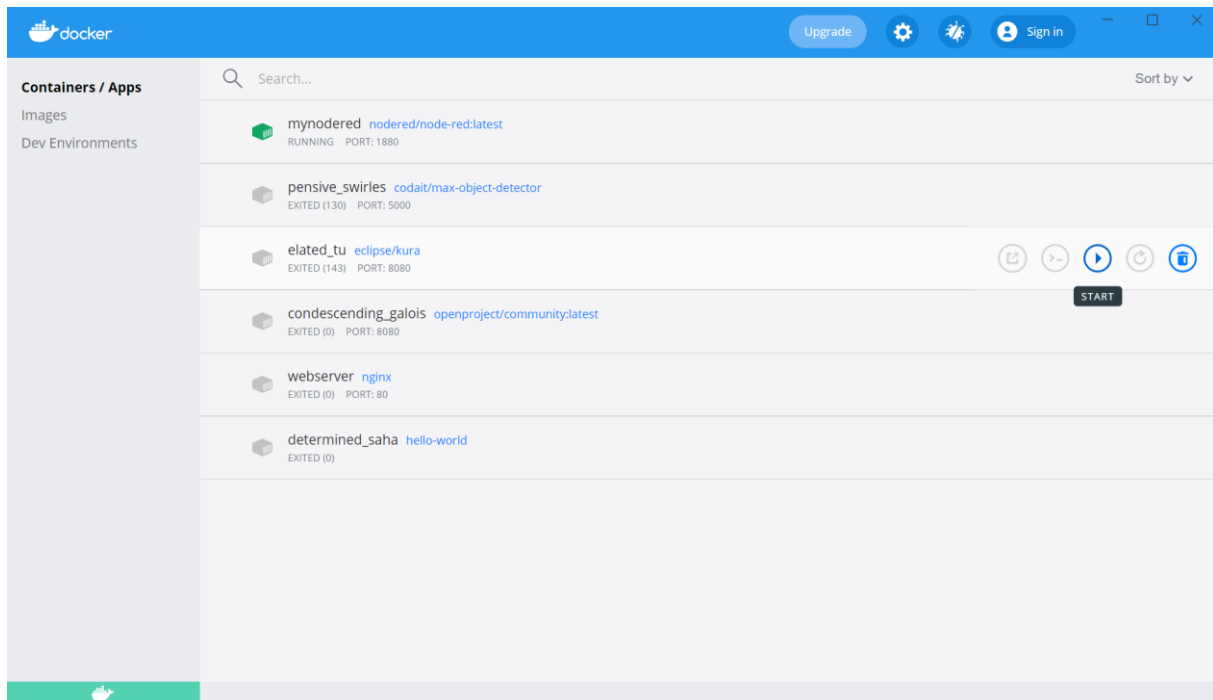


Figure 2. Main application window of Docker Desktop, showing the “Containers / Apps” page

The Containers / Apps page shows the available application instances that are available for execution in your Docker. In the example screenshot shown in Figure 2:

- The first line shows a Node-RED application (container) that is currently running, which is indicated by its green icon and the text “RUNNING” under the container name “mynodered”. This is just a user-given name for the application instance.
- In the third line, we see an Eclipse Kura application, which is an IoT gateway software. This container is currently idle, shown by the gray icon and the status text “EXITED”. You can think of this as a (virtual) machine that is currently turned off. We can easily turn on this machine (the container/app) by clicking on the Start button, as shown in the figure.

Where do those containers come from? That’s what we will see next in the “Images” page in Docker Desktop, as depicted in Figure 3.

- Here we see a list of locally available “image” files that were previously fetched / downloaded from the web. You can think of each image as the *template* of an application, which you can then instantiate to create actual running application(s).
- When we hover our mouse over an image item in this list, a “RUN” icon/button appears, allowing us to create a new instance (a new container/app) of this image, as we will later see in the 3rd lab session of this module.

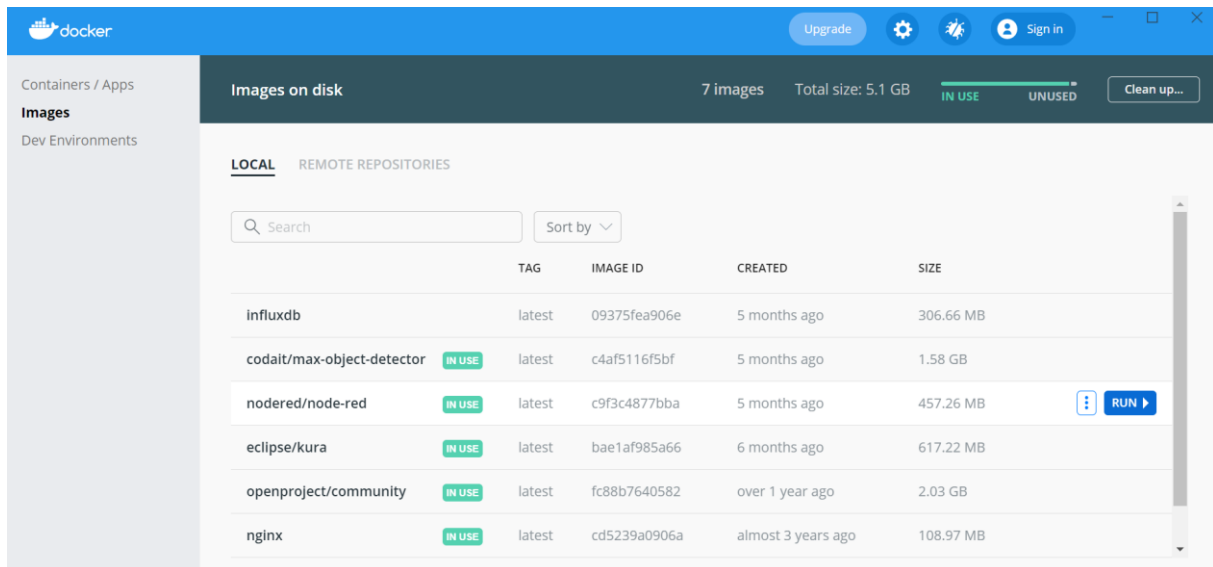


Figure 3. “Images” in Docker Desktop, where we can start (“run”) a new container / app

Installing an Application as a Docker Container / Pulling a new Image

In the previous section, we have seen a list of already existing images and containers in Docker. In this section, we will now look at downloading / pulling a new Docker image directly from the web.

We will use **Node-RED** as the example **application that we want to install as a Docker container**.

Before we start with installing the application, we will first learn about another way of interacting with the Docker environment: via the command line.

- We first need to start the Windows “command prompt” on our PC.
- For this, you can search for “cmd” in the Start menu to start the standard interface
- As an alternative, it is recommended to use the Windows “PowerShell”, which provides a Linux-style interface and all Linux/Unix system command. This could be more useful for you in the future.
- Let’s run the PowerShell interface and type “docker -help” to test if we can reach our docker setup through the command line. You should see something similar to this:

```
Windows PowerShell
PS C:\Users\fikret> docker --help

Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default
                        "C:\Users\fikret\.docker")
  -c, --context string  Name of the context to use to connect to the
                        daemon (overrides DOCKER_HOST env var and
                        default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "C:\Users\fikret\.docker\ca.pem")
  --tlscert string      Path to TLS certificate file (default
                        "C:\Users\fikret\.docker\cert.pem")
  --tlskey string       Path to TLS key file (default
                        "C:\Users\fikret\.docker\key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit

Management Commands:
  app*      Docker App (Docker Inc., v0.9.1-beta3)
  builder   Manage builds
  buildx*   Build with BuildKit (Docker Inc., v0.5.1-docker)
  config    Manage Docker configs
  container Manage containers
```

- There are many Docker commands that you can use to manage images and containers, which may learn more on its website², but we will cover the most useful commands for our purposes in the following section, which are “**Docker run**” and “**Docker pull**”.

We are now ready to go ahead and install Node-RED as our first Docker application. For this, we can follow the instructions on this page: <https://nodered.org/docs/getting-started/docker>.

Even though there are lots of details on this page, we will simply execute the following “**docker run**” command in our Windows PowerShell prompt.

docker run -it -p 1880:1880 -v node_red_data:/data --name mynodered nodered/node-red

When you do this for the first time on your PC, Docker will say that it is unable to find the node-red image locally. Then it will go ahead and “pull” (download) the required image over the Internet, as illustrated in the screenshot below:

² <https://docs.docker.com/engine/reference/commandline/cli/>

```
Windows PowerShell
PS C:\Users\fikret> docker run -it -p 1880:1880 -v node_red_data:/data --name mynodered nodered/node-red
Unable to find image 'nodered/node-red:latest' locally
latest: Pulling from nodered/node-red
ddad3d7c1e96: Pull complete
de915e575d22: Downloading [=====>] 6.224MB/22.21MB
7150aa69525b: Download complete
d7aa47be044e: Download complete
c380f47829c1: Downloading [=====>] 1.125MB/1.722MB
1f69ce71d6f2: Waiting
9e53ad41cd22: Waiting
4eea28ecf4c5: Waiting
5a79c0ddb9da: Waiting
3d5b829902c7: Waiting
ff6369088c85: Waiting
0471fe9913aa: Waiting
```

As an alternative, you can first use the “**docker pull nodered/node-red**” command to download the image and then use the same “**docker run**” command above to create a container from this image.

Let’s now breakup the “**docker run**” command above and try to understand the parameters and options in more detail:

docker run | -it | -p 1880:1880 | -v node_red_data:/data | --name mynodered | nodered/node-red

| | |
|-------------------------------|---|
| docker run | Main command to run a container, first pulling it if necessary |
| -it | Allows us to see the progress in a terminal session |
| -p 1880:1880 | connect local port 1880 to the exposed internal port 1880 |
| -v node_red_data:/data | Maps the local “node_red_data” folder on your PC to the container’s virtual “/data” folder, so that any changes made to flows are persisted even when you turn off or restart the container |
| --name mynodered | Assigns this container the name “mynodered” |
| nodered/node-red | The image from which the container is to be created |

When docker run command finishes its job and if everything goes well, your Node-RED application should be running as a container on your PC.

Your Windows PowerShell interface should look like in the figure below, which also shows (on the last line) the web address (URL) where your Node-RED application can be reached. Note that the local IP address 127.0.0.1 shown here is equivalent to “localhost”. So the URL shown in the figure is the same as <http://localhost:1880/>.

```

Welcome to Node-RED
=====

4 May 16:34:26 - [info] Node-RED version: v1.3.4
4 May 16:34:26 - [info] Node.js version: v10.24.1
4 May 16:34:26 - [info] Linux 5.10.25-linuxkit x64 LE
4 May 16:34:26 - [info] Loading palette nodes
4 May 16:34:26 - [info] Settings file : /data/settings.js
4 May 16:34:26 - [info] Context store : 'default' [module=memory]
4 May 16:34:26 - [info] User directory : /data
4 May 16:34:26 - [warn] Projects disabled : editorTheme.projects.enabled=false
4 May 16:34:26 - [info] Flows file : /data/flows.json
4 May 16:34:26 - [warn]

-----

Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.

-----

4 May 16:34:26 - [info] Starting flows
4 May 16:34:26 - [info] Started flows
4 May 16:34:26 - [info] Server now running at http://127.0.0.1:1880/


```

Now you can go back to your Docker Desktop application and you should be able to see the “nodered/node-red” image (as was shown in Figure 3 in the first lab) and “mynodered” container in the running state (as was shown in Figure 2).

Running & Configuring a Container/App from an Existing Local Image

In this lab section, we will see how we can create a new container/app as an instance of an already existing image in Docker. We will then learn how we can manage our running containers.

- Following from the first lab session, in the “Images” page of Docker (Figure 3), let’s click on the “RUN” icon for the node-red image:

| | | | | | | |
|------------------|--------|--------|--------------|--------------|-----------|---|
| nodered/node-red | IN USE | latest | c9f3c4877bba | 5 months ago | 457.26 MB |  RUN ▶ |
|------------------|--------|--------|--------------|--------------|-----------|---|

- This will create a pop-up window, where we can give our new container/app a name of our choice and specify the port on which we can access this app through the web browser.

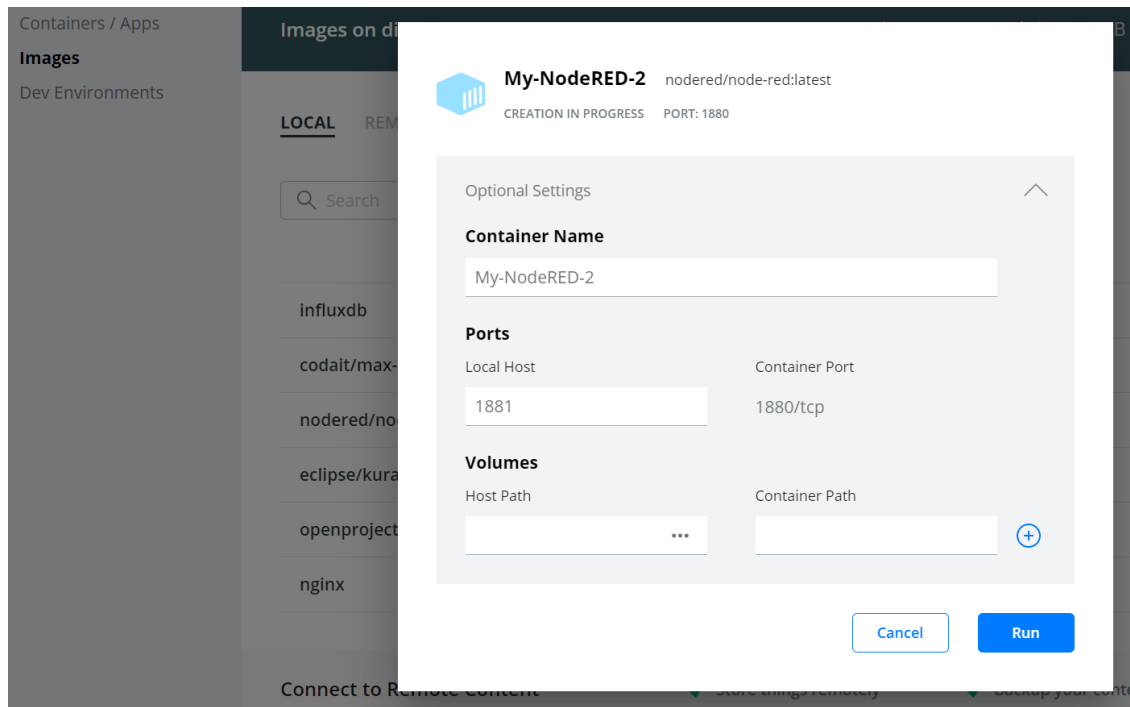


Figure 4. Pop-up window for configuring and running a new container/app

- In this example, we are instantiating a Node-RED application and making it available on port 1881 (since the default port of 1880 is already used by another Node-RED instance in our case). You can also choose another port.
- The Volumes section allows you to “map” a local folder (physical disk space) on your PC with a virtual folder on the container to be created. This helps you keep your application data even when you stop (“turn off”) and/or restart the container. Otherwise, the data in the container will be deleted once it is stopped.
- Once you click on the RUN button once again on the popup window, Docker will automatically switch to the “Containers / Apps” page, which should then show the new “My-NodeRED-2” application in the running state. This is shown in Figure 5.
- After the container is started, your application will be available at the address <http://localhost:1881>, depending on the selected port in the previous steps. In our example, the web address of our application will be <http://localhost:1881>.
- To open running the application, you can either
 - open your web browser and manually go this address or
 - hover your mouse over the My-Node-RED-2 container line and use the “OPEN IN BROWSER” icon, as illustrated in Figure 5.

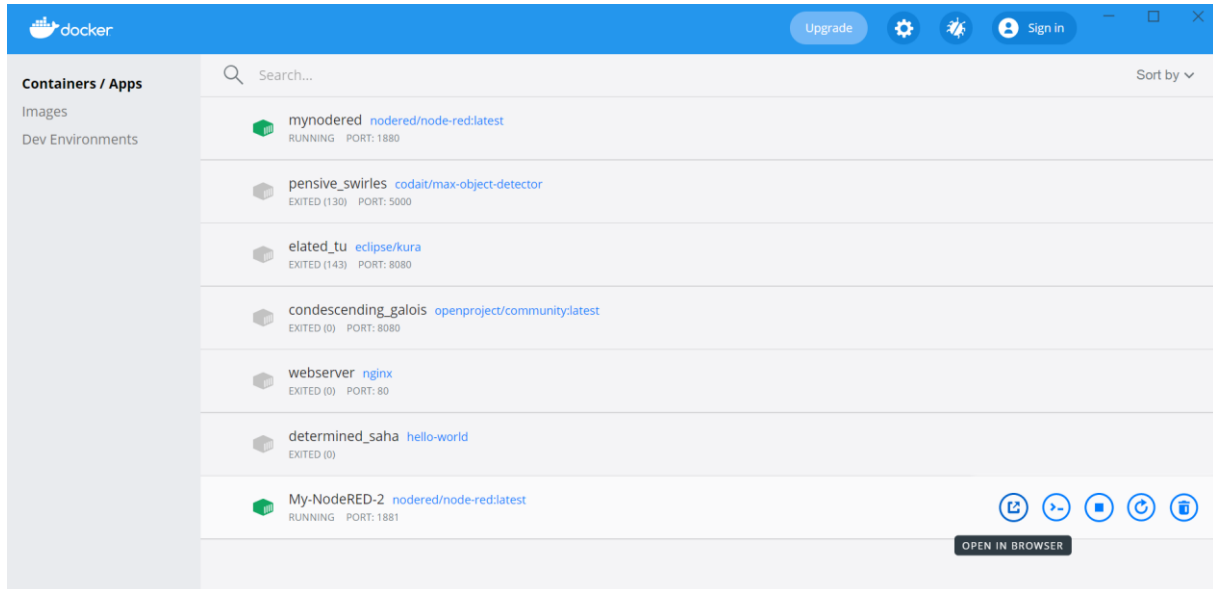
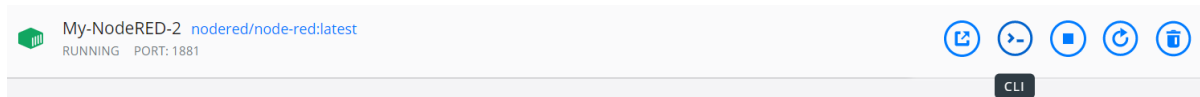


Figure 5. The newly created Node-RED container in 'running' state

We can also access our new container (virtual machine) through a command-line interface (CLI) by clicking on the "CLI" button, as shown below



- This will open up a command prompt, which more clearly shows that this Node-Red container is really like a virtual machine running Linux operating system.
- Here we can enter standard Linux commands, such as "cd" to change the current directory or "ls -al" for a detailed listing of all files and folders in the current directory, as depicted in Figure 6.
- Using the Linux commands is beyond the scope of this module, so we will not go into the details of here. You may now safely close the command line interface.




```

docker exec -it ebb6bbd2fb8f56c2b9f4e1337c0247e828dd4bff56829314180fff4bf0a79e8c /bin/sh
~ $ ls -al
total 32
drwxrwxr-x 1 node-red root      4096 May  4 15:05 .
drwxr-xr-x 1 root      root      4096 Dec  8 18:25 ..
-rw-r----- 1 node-red node-red    74 May  4 15:17 .ash_history
drwxr-xr-x 1 node-red root     12288 Dec  8 18:26 node_modules
-rw-rw-r-- 1 node-red root      1038 Dec  8 18:25 package.json
~ $ cd ..
/usr/src $ cd ..
/usr $ ls -al
total 52
drwxr-xr-x 1 root      root      4096 Dec  8 18:26 .
drwxr-xr-x 1 root      root      4096 May  4 14:41 ..
drwxr-xr-x 1 root      root      4096 Dec  8 18:26 bin
drwxr-xr-x 22 root      root      4096 Dec  8 18:26 include
drwxr-xr-x 1 root      root      4096 Dec  8 18:26 lib
drwxr-xr-x 1 root      root      4096 Dec  8 18:26 libexec
drwxr-xr-x 1 root      root      4096 Oct 28 2020 local
drwxr-xr-x 1 root      root      4096 Dec  8 18:25 sbin
drwxr-xr-x 1 root      root      4096 Dec  8 18:26 share
drwxr-xr-x 1 root      root      4096 Dec  8 18:25 src
drwxr-xr-x 4 root      root      4096 Dec  8 18:26 x86_64-alpine-linux-musl
/usr $

```

Figure 6. Command line interface of our Node-RED container (Linux virtual machine)

Finally, there are three additional icons/buttons next to CLI for managing a container, as it was illustrated in Figure 5:

- Stop: 
- Restart: 
- Delete: 

These buttons allow us to simply control the container/app, as their names imply. By using the delete button, you may also practice removing an existing container and creating a new instance through the images page, as we covered in the previous part.