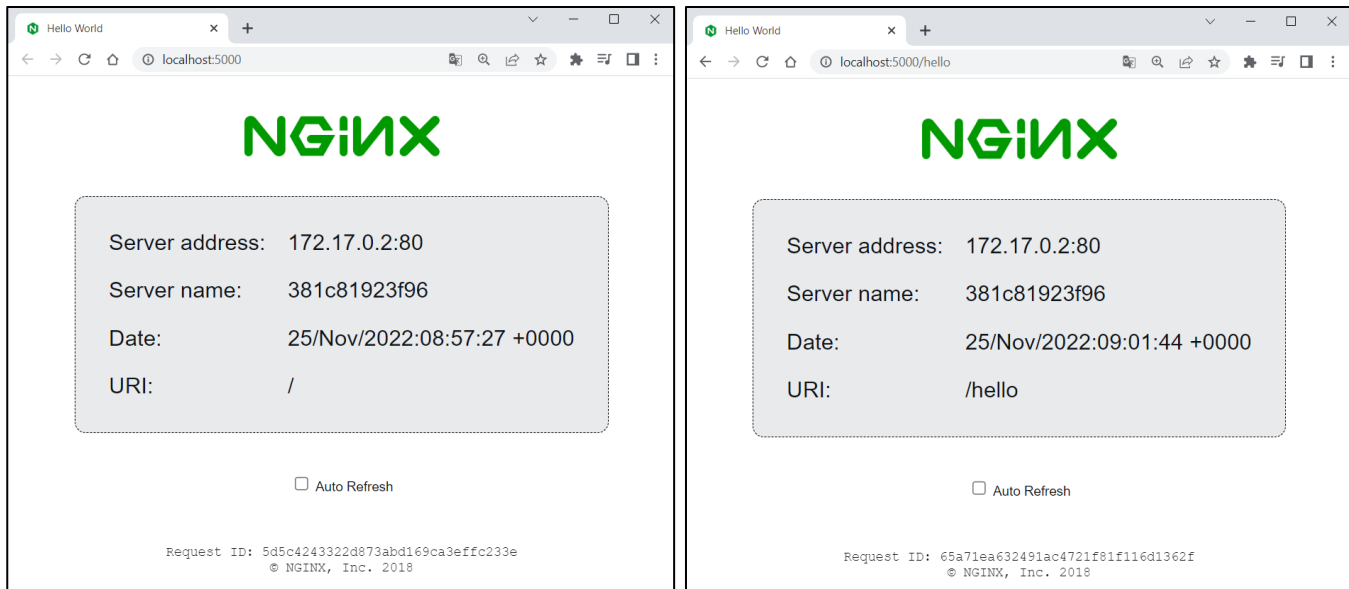


Lab: Containers and Docker

Problems for the lab for the ["Containers and Clouds" course @ SoftUni](#)

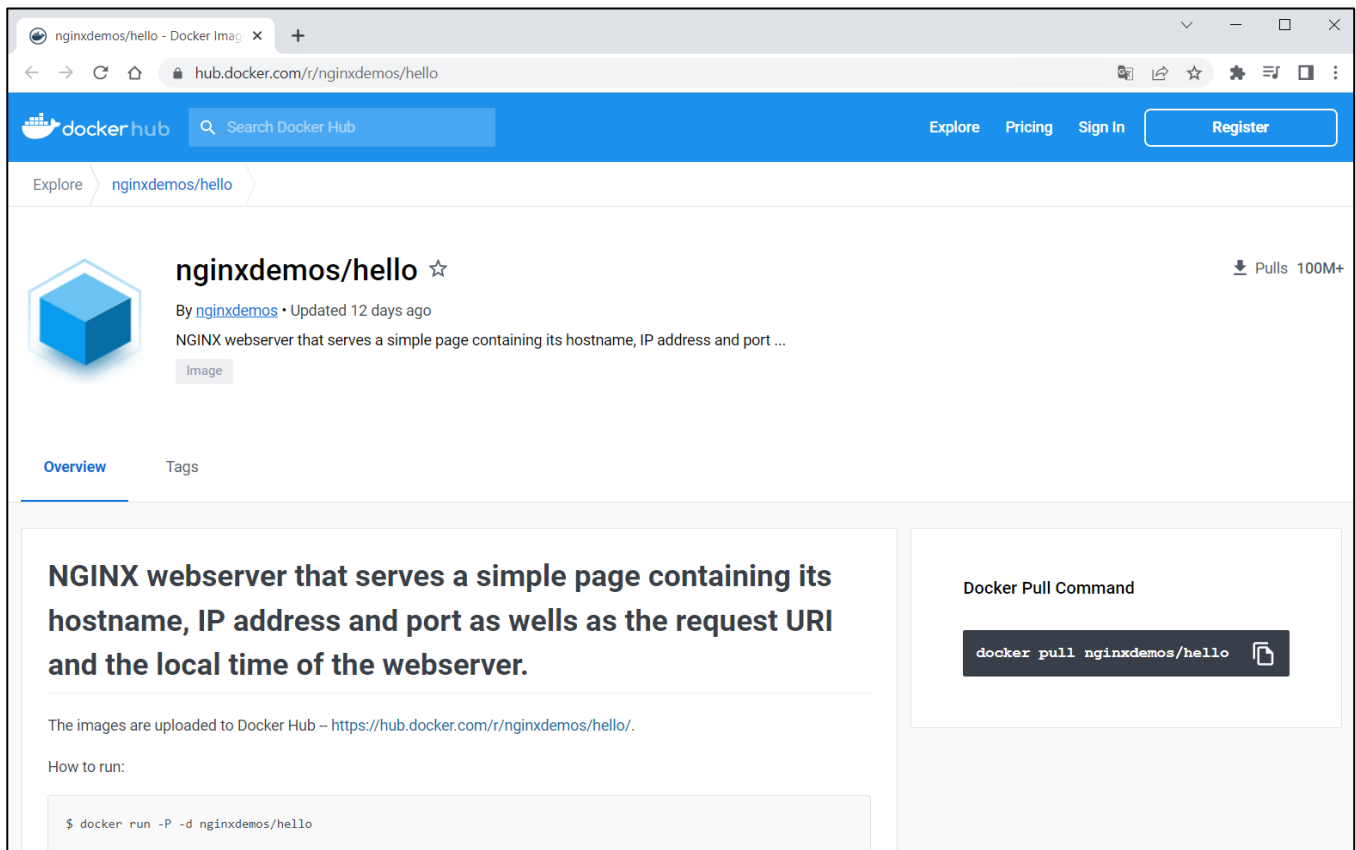
1. NGINX Server Container

In this task, we will run a **simple NGINX server** in a **Docker container**. The **server** only returns **some server information**:



Download Image

To create an **NGINX server container**, we shall first **pull the NGINX server image** from Docker Hub. You can find the image documentation on **Docker Hub** here: <https://hub.docker.com/r/nginxdemos/hello>. You can always **refer to the documentation** to get instructions on how to pull, build and run the image:



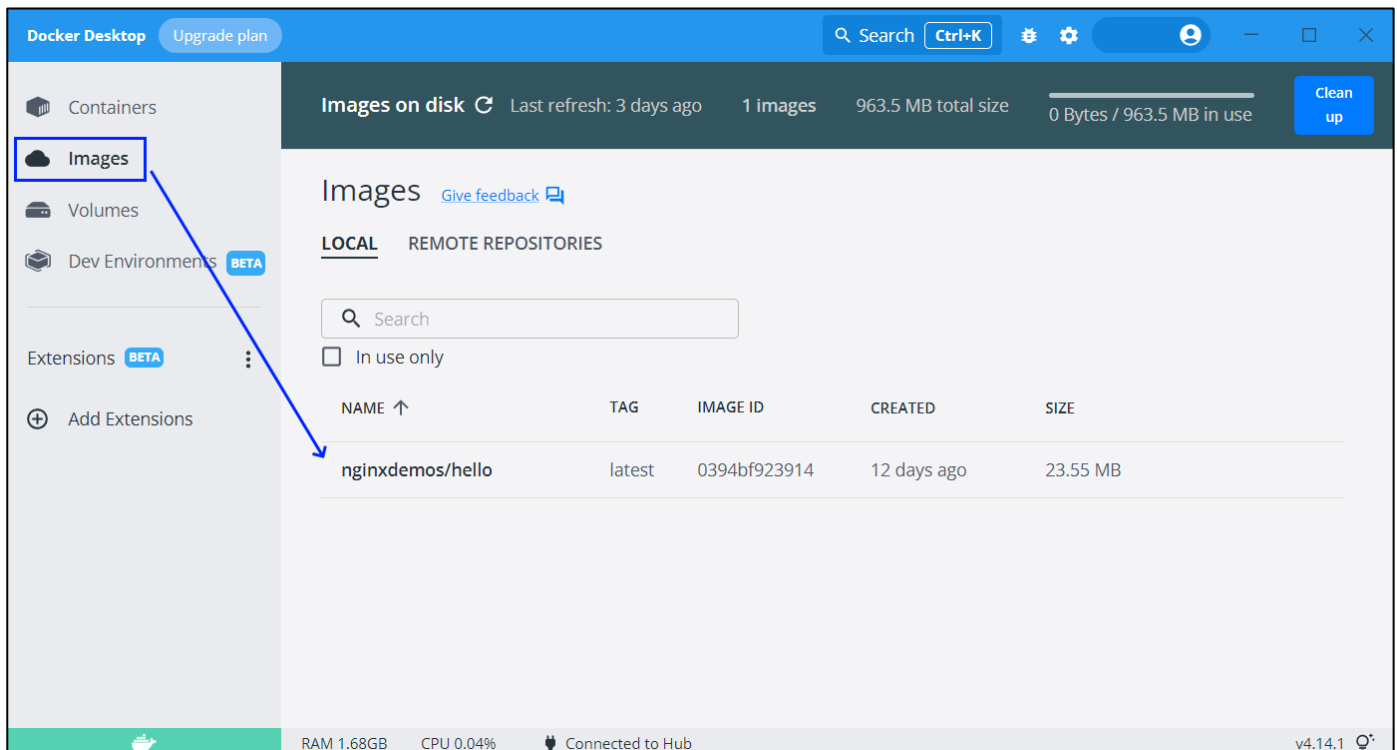
The screenshot shows the Docker Hub page for the `nginxdemos/hello` image. The page includes the Docker Hub logo, a search bar, and navigation links. The image details section shows the image icon, name, and a description: "NGINX webserver that serves a simple page containing its hostname, IP address and port ...". It also displays the Docker Pull Command: `docker pull nginxdemos/hello` and a terminal snippet showing the command to run the image: `$ docker run -P -d nginxdemos/hello`.

Open a **CLI**, for example, **PowerShell** or **Windows Terminal** or **Command Prompt** (**PowerShell** is recommended), and let's first **pull the Docker image** of the server. You should just use the **pull command** from the documentation:

```
PS C:\Users\PC> docker pull nginxdemos/hello
Using default tag: latest
latest: Pulling from nginxdemos/hello
ca7dd9ec2225: Already exists
76a48b0f5898: Already exists
2f12a0e7c01d: Already exists
1a7b9b9bbef6: Already exists
b704883c57af: Already exists
4342b1ab302e: Already exists
56b66d0c752b: Pull complete
b53a553ce476: Pull complete
04050cb56f56: Pull complete
Digest: sha256:0c9ae07ff186e92e8ee5b94b42266504f0ad4139a0f548ab29cd72ca089af49e
Status: Downloaded newer image for nginxdemos/hello:latest
docker.io/nginxdemos/hello:latest
```

You can see that the **latest image is downloaded by default**. In addition, some of the **image layers** existed from before (from other installations), so they **were not downloaded again** – this is the advantage of Docker image layers.

You can look at your **downloaded images** in **Docker Desktop**, in the **[Images] tab**:



You can also use the command below to **display a list of all images** you have:

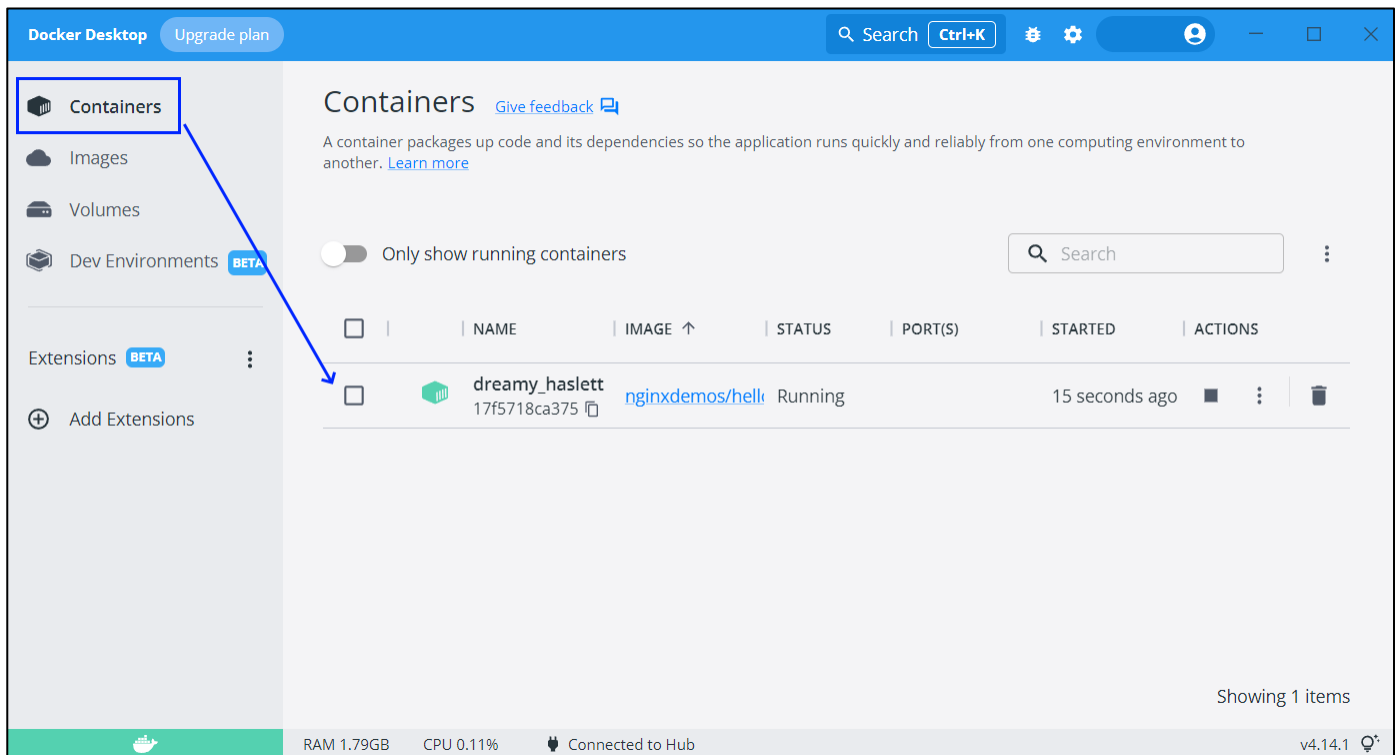
```
PS C:\Users\PC> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
nginxdemos/hello    latest         0394bf923914   12 days ago    23.6MB
```

Run a Container

Now we want to **run a container** with the **NGINX server image**, which is already downloaded from Docker Hub to our local machine. Use the **docker run** command and try this way:

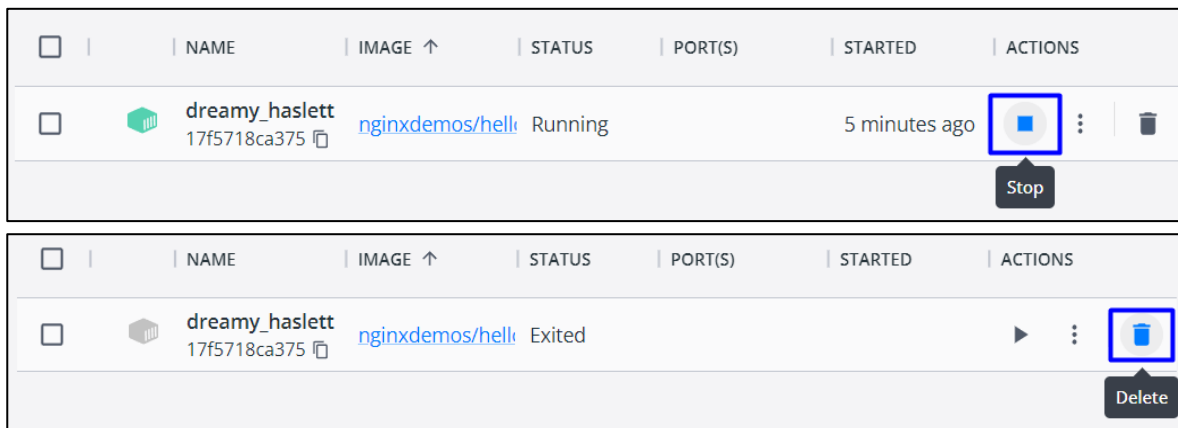
```
PS C:\Users\PC> docker run nginxdemos/hello
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf is not a file or does not exist
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
```

Now we have a **running container** with a **random name**, as we did not set it explicitly. We can see it on **Docker Desktop [Containers]**:



However, you can see that the "PORT(S)" column is empty, which means that our container cannot be accessed through the Internet, as it is isolated.

To fix this, we should **expose a port from the container**. But first, let's **stop and delete the container** we already have by **clicking on the buttons** in Docker Desktop:



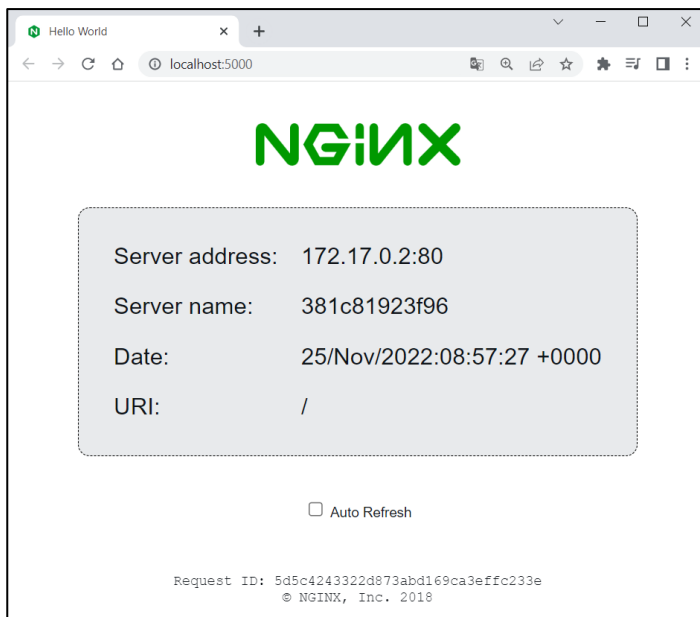
Let's **create another container** and **publish its port to the host**. This is done with the **-p option**. After it, we shall add the **port we want the server to be on our machine** (it can be any free port, but let's use **5000**) and the **internal port the server uses** – in our case **80**. Do it like this:

```
PS C:\Users\PC> docker run -p 5000:80 nginxdemos/hello
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf is not a file or does not exist
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
```

Now the **container is exposed**:

	NAME	IMAGE ↑	STATUS	PORT(S)	STARTED	ACTIONS
	dreamy_kalam 381c81923f96	nginxdemos/hello:latest	Running	5000:80	3 minutes ago	

So, you can go to **http://localhost:5000** and access the working server:



Stop and delete the container again and let's do one more thing.

Run a Named Container in Detached Mode

This time, we want to **run a container** with a **name** and in **detached mode** (the container will be running in the background). To do this, use the **--name** option with the **container name you want**, and the **-d** option for **detached mode**:

```
PS C:\Users\PC> docker run -p 5000:80 -d --name code_it_up nginxdemos/hello
e33b27fefae6a98cef1e6af689de1752609a5844c316644e17e8c33efdc94d66
```





As you can see, now we have only the **container id** returned in the terminal and the **container logs are now shown** (because of the **detached mode**). However, you can **see the logs** with **docker logs** and the **container id or name**:

```
PS C:\Users\PC> docker logs e33b27fefae6a98cef1e6af689de1752609a5844c316644e17e8c33efdc94d66
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf is not a file or does not exist
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
```

You can also use only the **first two symbols of the container's id**, not the whole one when they are unique (we have no other container with the same first symbols):





```
PS C:\Users\PC> docker logs e3
```

Or you can see them directly from **Docker Desktop** when you **click on the container's name**:

	NAME	IMAGE ↑	STATUS	PORT(S)	STARTED	ACTIONS
<input type="checkbox"/>	code_it_up e33b27fefae6	nginxdemos/hello	Running	5000:80	12 minutes ag	   

code_it_up nginxdemos/hello

RUNNING

Logs

Inspect

Terminal

Stats

```

2022-11-25 11:08:34 /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-defa
2022-11-25 11:08:34 /docker-entrypoint.sh: info: /etc/nginx/conf.d/default.conf is not a :
2022-11-25 11:08:34 /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.Lu
2022-11-25 11:08:34 /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
2022-11-25 11:08:34 /docker-entrypoint.sh: Configuration complete; ready for start up
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: using the "epoll" event method
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: nginx/1.23.2
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: built by gcc 11.2.1 20220219 (Alpine 11.2.1_git20220219)
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: OS: Linux 5.10.16.3-microsoft-standard-WSL2
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker processes
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 20
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 21
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 22
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 23
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 24
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 25
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 26
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 27
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 28
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 29
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 30
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 31
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 32
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 33
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 34
2022-11-25 11:08:34 2022/11/25 09:08:34 [notice] 1#1: start worker process 35

```

RAM 1.82GB CPU 0.04% Connected to Hub v4.14.1

Notice that the **name of our container** is how we explicitly set it to be.

Examine and Delete Container and Image

We can **see all containers** we have like this:

```

PS C:\Users\PC> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
e33b27fefae6   nginxdemos/hello "/docker-entrypoint...." 18 minutes ago Up About a minute 0.0.0.0:5000->80/tcp

```

To **see all running containers**, use:

```

PS C:\Users\PC> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
e33b27fefae6   nginxdemos/hello "/docker-entrypoint...." 18 minutes ago Up About a minute 0.0.0.0:5000->80/tcp

```

In our case, we have a **single running container** and we have the **same output** from the two commands.

Now let's use the terminal to **stop and delete our container**. Use the following commands with the **container id** or **container name**:

```

PS C:\Users\PC> docker stop e3
e3
PS C:\Users\PC> docker rm e3
e3

```

Both commands return the **id or the name of the container**.

Finally, we can also **delete the NGINX server image**:

```
PS C:\Users\PC> docker rmi nginxdemos/hello
Untagged: nginxdemos/hello:latest
Untagged: nginxdemos/hello@sha256:0c9ae07ff186e92e8ee5b94b42266504f0ad4139a0f548ab29cd72ca089af49e
Deleted: sha256:0394bf9239149150b4540a57b60195e9440de1af06dfe7081370e92d5ae57b56
Deleted: sha256:e78ff3228f05ef9f22f0ce2736fbe6eb4e14c49ef625fe4655575bd76088dfea
Deleted: sha256:1bda19da7ab3bd073a81e52d642ead1a38609953ac3193a595b21234b1ef376f
Deleted: sha256:6cf0426616d5f6f15f9f26ce0422fdfe7dfc73c58c7a3ce39cb90d92b2f2c631
```

That's how we are supposed to work with **Docker images and containers** at a basic level.

2. Vue.js App in Container

Now we will see how to run a **Vue.js app** in a **Docker container**. We will **create an app just for the demo**.

Create and Set Up a Vue.js App

Let's use Terminal to create an app called **MyWebsite**, in a folder you choose:

```
PS C:\Users\ > vue init browserify MyWebsite

? Project name my-website
? The version of the package 0.1.0
? Project description A Vue.js project
? Author Your Name <yourname@example.com>
? Vue build standalone
? Use ESLint to lint your code? Yes
? Setup unit tests with Karma + Jasmine? No

vue-cli · Generated "MyWebsite".

To get started:

  cd MyWebsite
  npm install
  npm run dev

PS C:\Users\ > cd MyWebsite
PS C:\Users\ \MyWebsite> npm install
```

Let's now **pull the image** we will need to run the app – **NodeJs**.

(https://hub.docker.com/_/node). Note that the **image version** should be **the same** as the **app's Node version**:

```
PS C:\Users\ \MyWebsite> docker pull node:16
16: Pulling from library/node
a94073ab46f8: Already exists
c0f81a21db43: Already exists
431d5332f45f: Already exists
817c83dfe9f0: Already exists
48824897ccbe: Already exists
ab2e44d4ae4a: Already exists
efc390dd9bed: Already exists
be2c80304069: Already exists
Digest: sha256:550f484fc5f314b575f5e397c9e2c71d7f218e59729fcd9ffa7ea1fc825dce7
Status: Downloaded newer image for node:16
docker.io/library/node:16
```

Now let's **run the application locally** in the standard way to check if **everything works as expected**:

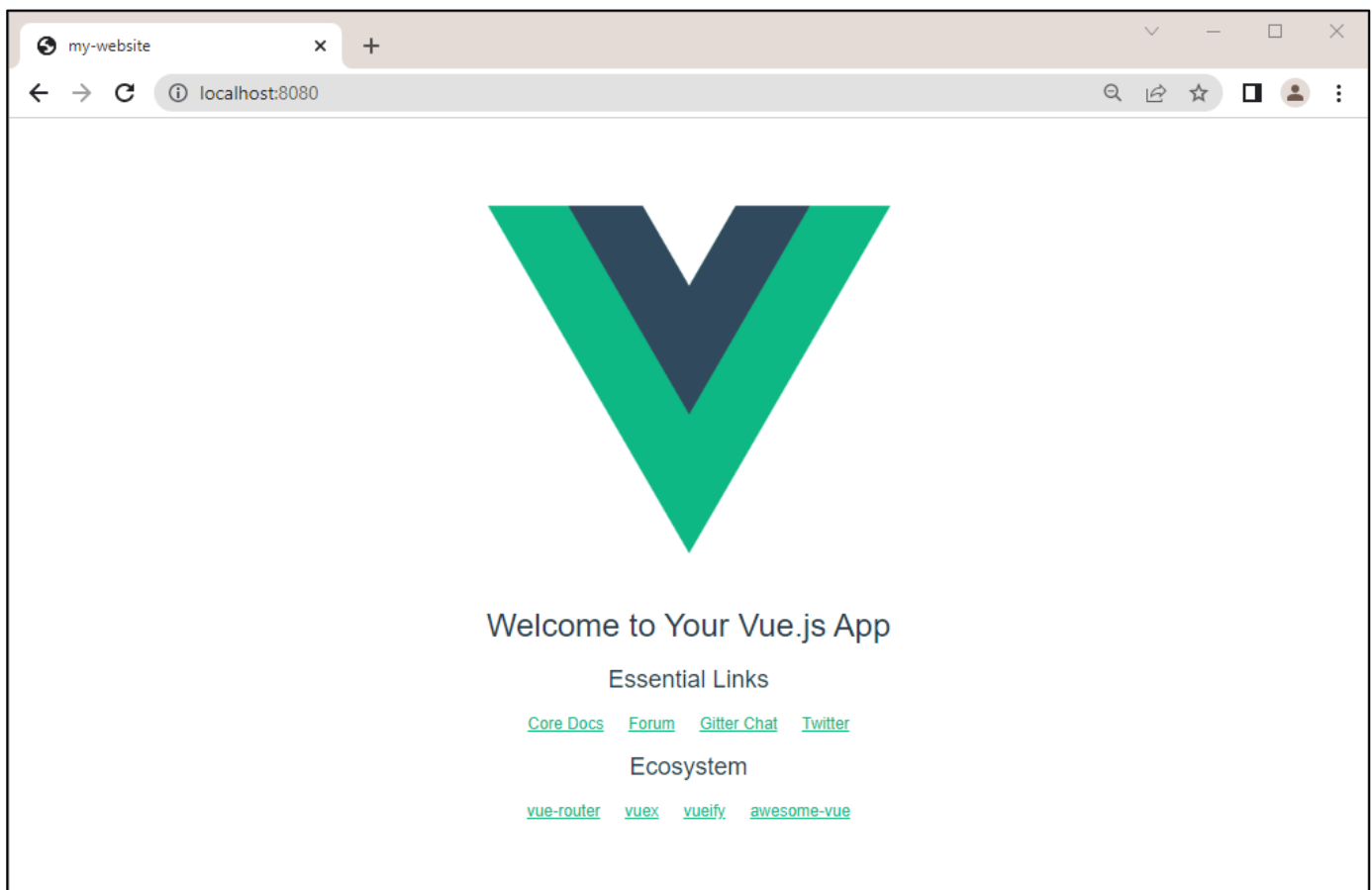

```
PS C:\Users\ \MyWebsite> npm run dev

> my-website@0.1.0 dev
> npm-run-all --parallel watchify serve

> my-website@0.1.0 watchify
> watchify -vd -p browserify-hmr -e src/main.js -o dist/build.js

> my-website@0.1.0 serve
> http-server -o -c 1 -a localhost

Starting up http-server, serving ./
Available on:
  http://localhost:8080
Hit CTRL-C to stop the server
```



Now we should **modify the app's package .json file**, so that the **app runs on the IP we want**. Open the file in any editor and **change the scripts section settings** like this:

```
"scripts": {
  "watchify": "watchify -vd -p browserify-hmr -e src/main.js -o dist/build.js",
  "serve": "http-server -o -c 1 -a 0.0.0.0",
  "dev": "npm-run-all --parallel watchify serve",
  "lint": "eslint --ext .js,.vue src",
  "build": "cross-env NODE_ENV=production browserify -g envify -p [ vueify/plugins/extract-css -o dist/build.css ]
},
```

Run the App in a Container

To run the Vue.js **app in a container**, we will need to **create a container** with an **exposed port**, a **volume** and an **interactive shell**, so that we can **run the app inside the container** with the **docker run** command.

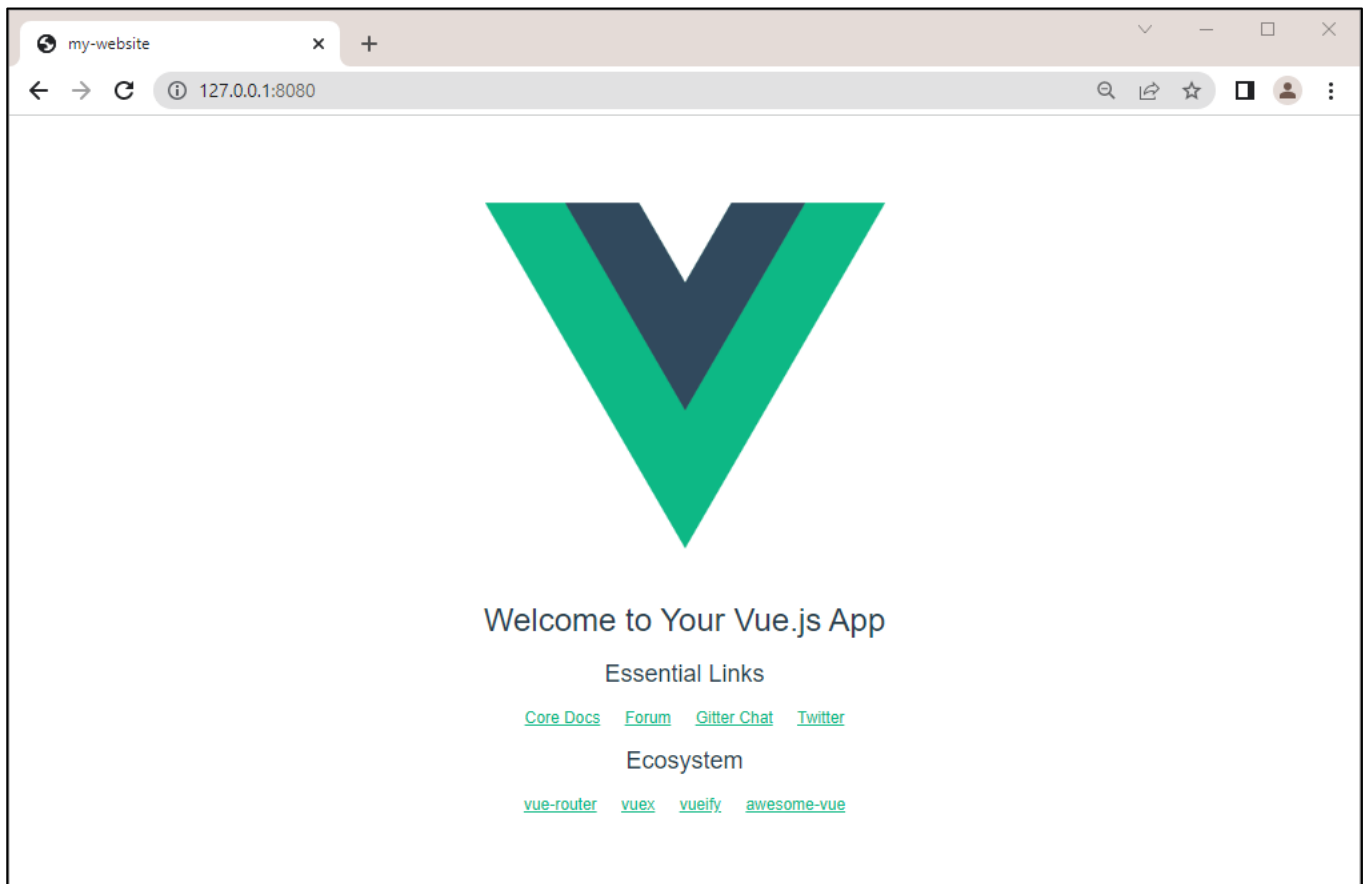
To do so, execute the following command:

```
PS C:\Users\ \MyWebsite> docker run -it -p 8080:8080 `
>> -v ${PWD}:/app -w /app node:16 npm run serve

> my-website@0.1.0 serve
> http-server -o -c 1 -a 0.0.0.0

Starting up http-server, serving ./
Available on:
  http://127.0.0.1:8080
  http://172.17.0.2:8080
Hit CTRL-C to stop the server
```

Access the app at <http://127.0.0.1:8080> to validate that the app is running:



3. MongoDB in Container

Our next task is to **run a container** with a **Mongo database** in it. To do this, we will need the **following image** from **Docker Hub**: https://hub.docker.com/_/mongo

You can look at the "How to use this Image" section to learn how to **run the database container**. However, we will also **show and explain** the process step by step.

Create the Container

First, pull the MongoDB image with the **docker pull mongo** command:

```
PS C:\Users\ \MyWebsite> docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
1bc677758ad7: Pull complete
7eb83bb7be98: Pull complete
e95121721c4c: Pull complete
799041b403ca: Pull complete
1828e70ef29a: Pull complete
8e3781beae9e: Pull complete
5d5753162333: Pull complete
44dd404b40f4: Pull complete
44599c9d5d1b: Pull complete
Digest: sha256:928347070dc089a596f869a22a4204c0feace3eb03470a6a2de6814f11fb7309
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
```

Our next step is to **run** the container, using the following command:

```
PS C:\Users\ \MyWebsite> docker run `
>> -p 27017:27017 `
>> -v ${PWD}/data:/etc/mongo `
>> -d `
>> -e MONGO_INITDB_ROOT_USERNAME=mongoadmin `
>> -e MONGO_INITDB_ROOT_PASSWORD=mongoadminpass `
>> mongo
4216ee7578dd9a558e8ca68578dd55bcf57d274de79331eaecc5b174b0d0985a
```

- **docker run** → starts a new Docker container;
- **-p 27017:27017** → sets the external and internal ports to **27017**, so that we can access the MongoDB from outside the container;
- **-v \${PWD}/data:/etc/mongo**
 - **\${PWD}** → the host directory;
 - **/etc/mongo** → the container directory;
- **-d** → runs the container in detached mode (it will run in the background);
- **-e MONGO_INITDB_ROOT_USERNAME=mongoadmin** → sets the admin username;
- **-e MONGO_INITDB_ROOT_PASSWORD=mongoadminpass** → sets the admin password;
- **mongo** → specifies the image.

Connect to the Database Container through MongoDB Compass

You can connect to the container database in **MongoDB Compass**, using the **username** and password that we created in the **previous** step:

New Connection

Connect to a MongoDB deployment

FAVORITE

URI ⓘ Edit Connection String ☒

mongodb://mongoadmin:mongoadminpass@localhost:27017/?authMechanism=DEFAULT

▼ Advanced Connection Options

General
 Authentication
 TLS/SSL
 Proxy/SSH
 In-Use Encryption
 Advanced

Authentication Method

None
 Username/Password
 X.509
 Kerberos
 LDAP
 AWS IAM

Username

mongoadmin

Password

.....

Save

Save & Connect

Connect

MongoDB Compass - localhost:27017/local.startup_log

Connect
 Edit
 View
 Collection
 Help

localhost:27017
...

Documents

local.startup_log

+

My Queries

Databases

Search

▶ admin
 ▶ config
 ▼ local
 ▶ replset.election
 ▶ replset.minvalid
 ▶ **startup_log** ...

local.startup_log

2 DOCUMENTS 1 INDEXES

Documents
 Aggregations
 Schema
 Explain Plan
 Indexes
 Validation

Filter ⓘ ⓘ
 Type a query: { field: 'value' }
 Reset Find </> More Options ▶

+ ADD DATA
 EXPORT COLLECTION
 1 - 2 of 2
 < >
 ≡ {} |

```

_id: "e2b56c281f76-1683847605492"
hostname: "e2b56c281f76"
startTime: 2023-05-11T23:26:45.000+00:00
startTimeLocal: "Thu May 11 23:26:45.492"
  ▶ cmdLine: Object
  ▶ pid: 28
  ▶ buildinfo: Object

_id: "e2b56c281f76-1683847615240"
hostname: "e2b56c281f76"
startTime: 2023-05-11T23:26:55.000+00:00
startTimeLocal: "Thu May 11 23:26:55.240"
  ▶ cmdLine: Object
  ▶ pid: 1
  ▶ buildinfo: Object
          
```

>_MONGOSH