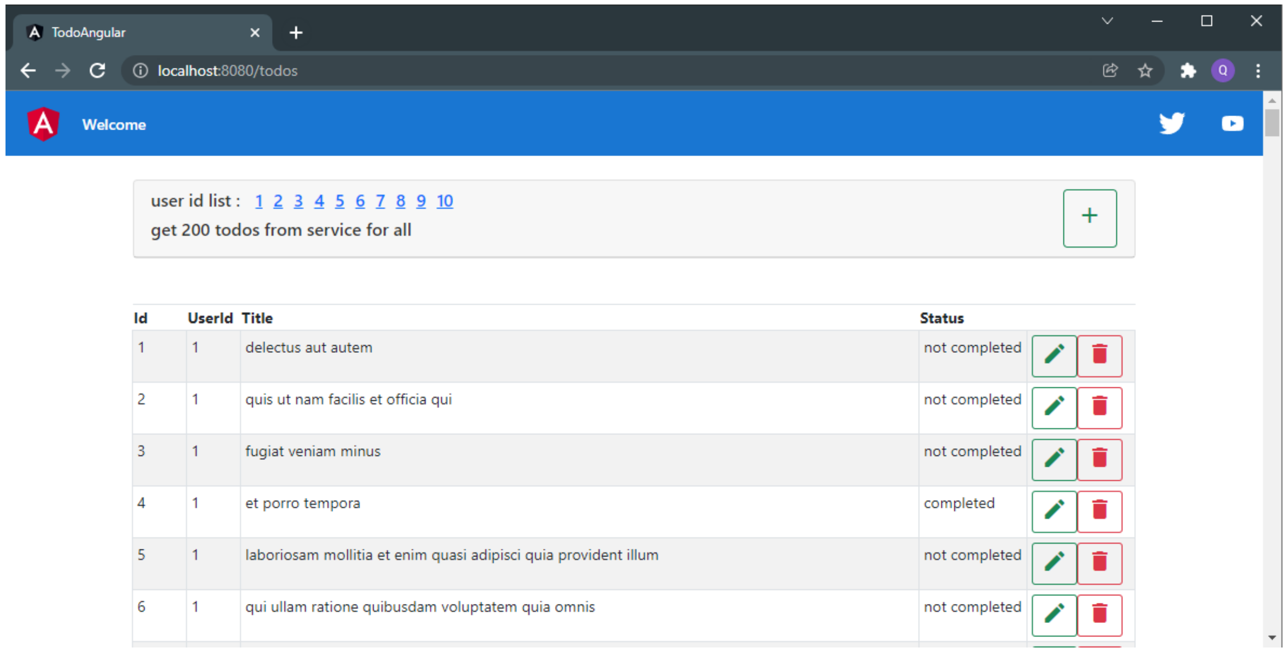
2021-12

Todo App Full Stack Development With Docker

Franke Chen

Workshop Agenda

- Docker Overview
- TODO App Overview
- TODO Angular (html/js/css)
- TODO Api (.Net core)
- TODO Console (.Net core)
- Deploy
- Q & A


TodoAngular

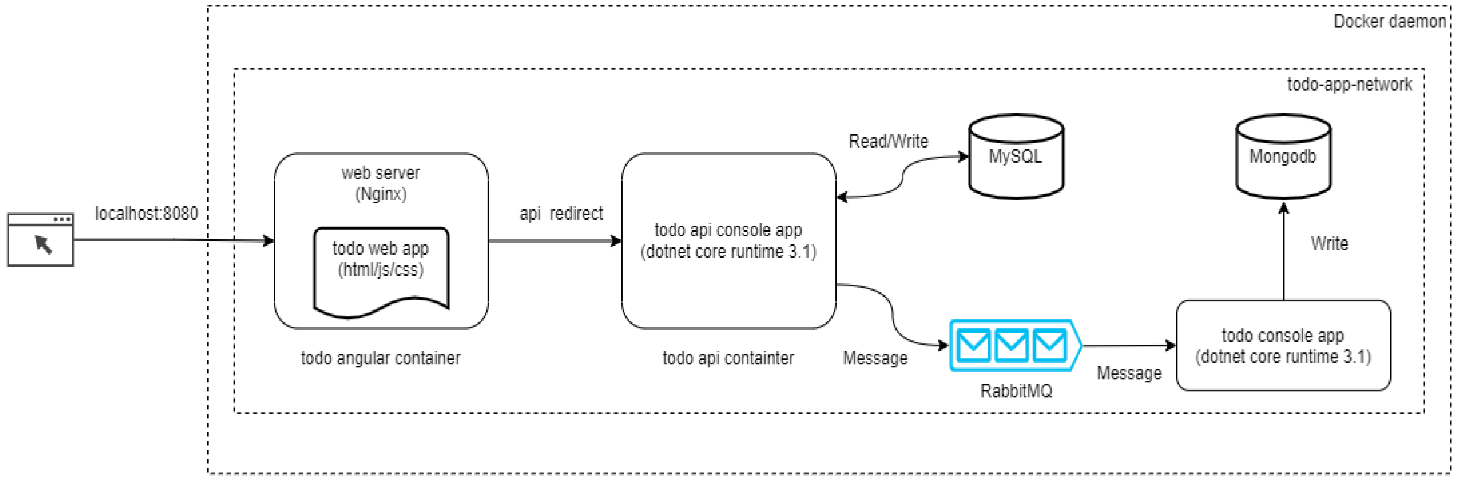
localhost:8080/todos

Welcome

user id list : 1 2 3 4 5 6 7 8 9 10

get 200 todos from service for all

Id	UserId	Title	Status
1	1	delectus aut autem	not completed
2	1	quis ut nam facilis et officia qui	not completed
3	1	fugiat veniam minus	not completed
4	1	et porro tempora	completed
5	1	laboriosam mollitia et enim quasi adipisci quia provident illum	not completed
6	1	qui ullam ratione quibusdam voluptatem quia omnis	not completed



Docker Overview

1.Background

2.What is Docker

3.Docker vs VM

4.Why Docker

5.Docker architecture

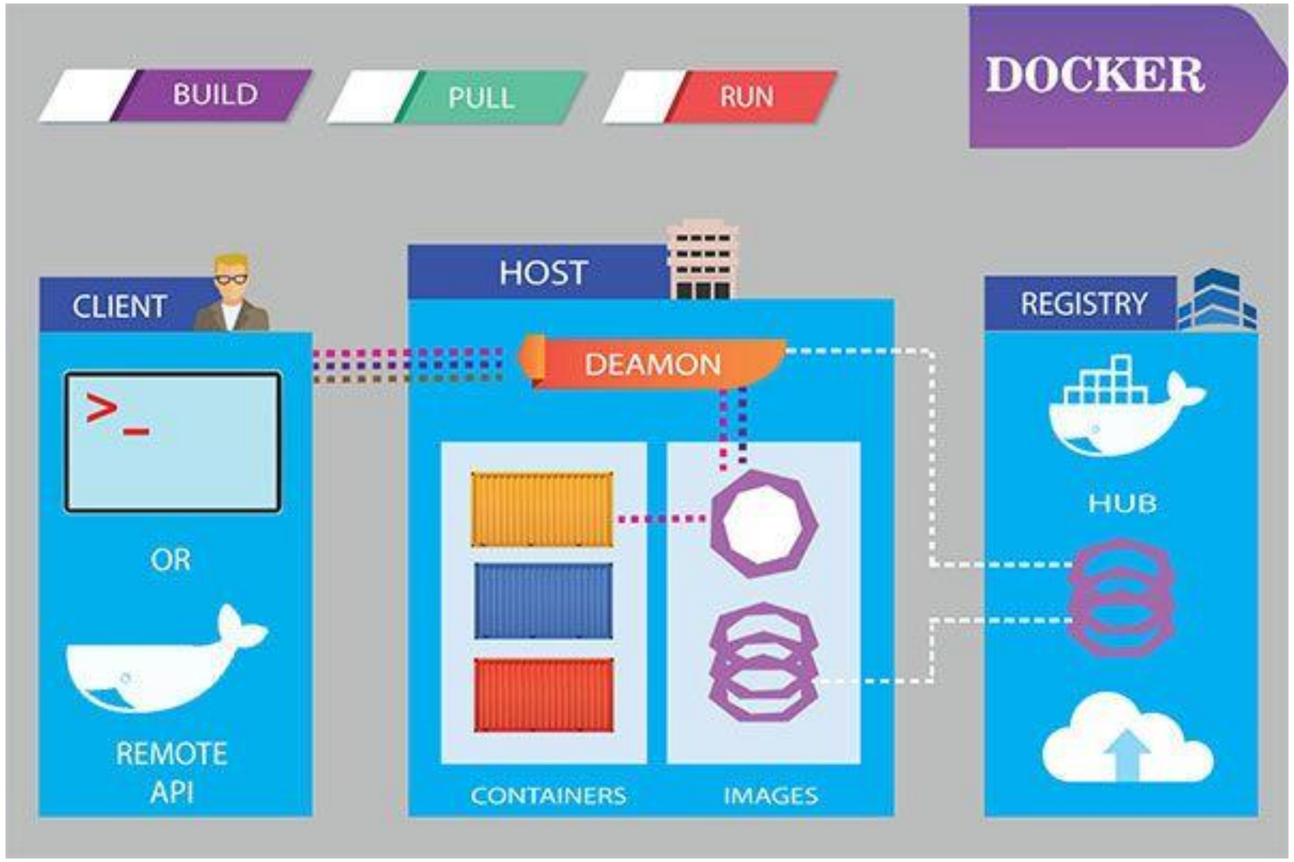
6.Samples

- Hello world
- Docker/getting-started
- Nginx

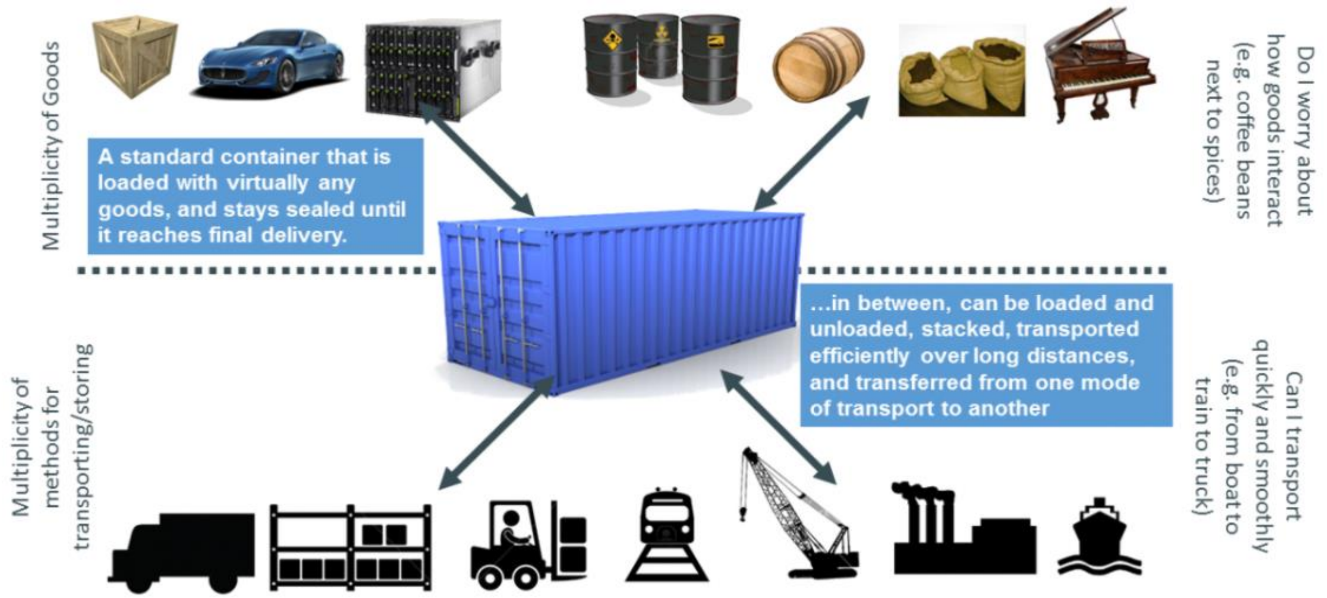
7.Docker-compose

- sample: MySQL, Mongodb, Redis, RabbitMQ

8.Dockerfile



Docker Overview - Shipment and Container

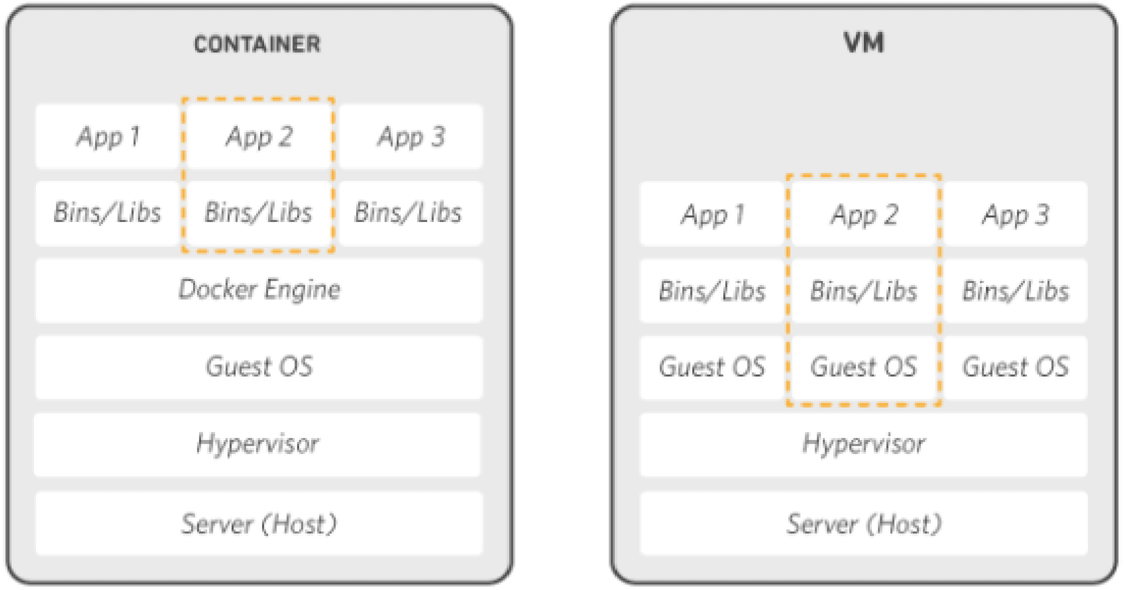

Multiplicity of Goods
A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.
Do I worry about how goods interact (e.g. coffee beans next to spices)
Multiplicity of methods for transporting/storing
...in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another
Can I transport quickly and smoothly (e.g. from boat to train to truck)

Docker Overview – What is docker

How Docker works

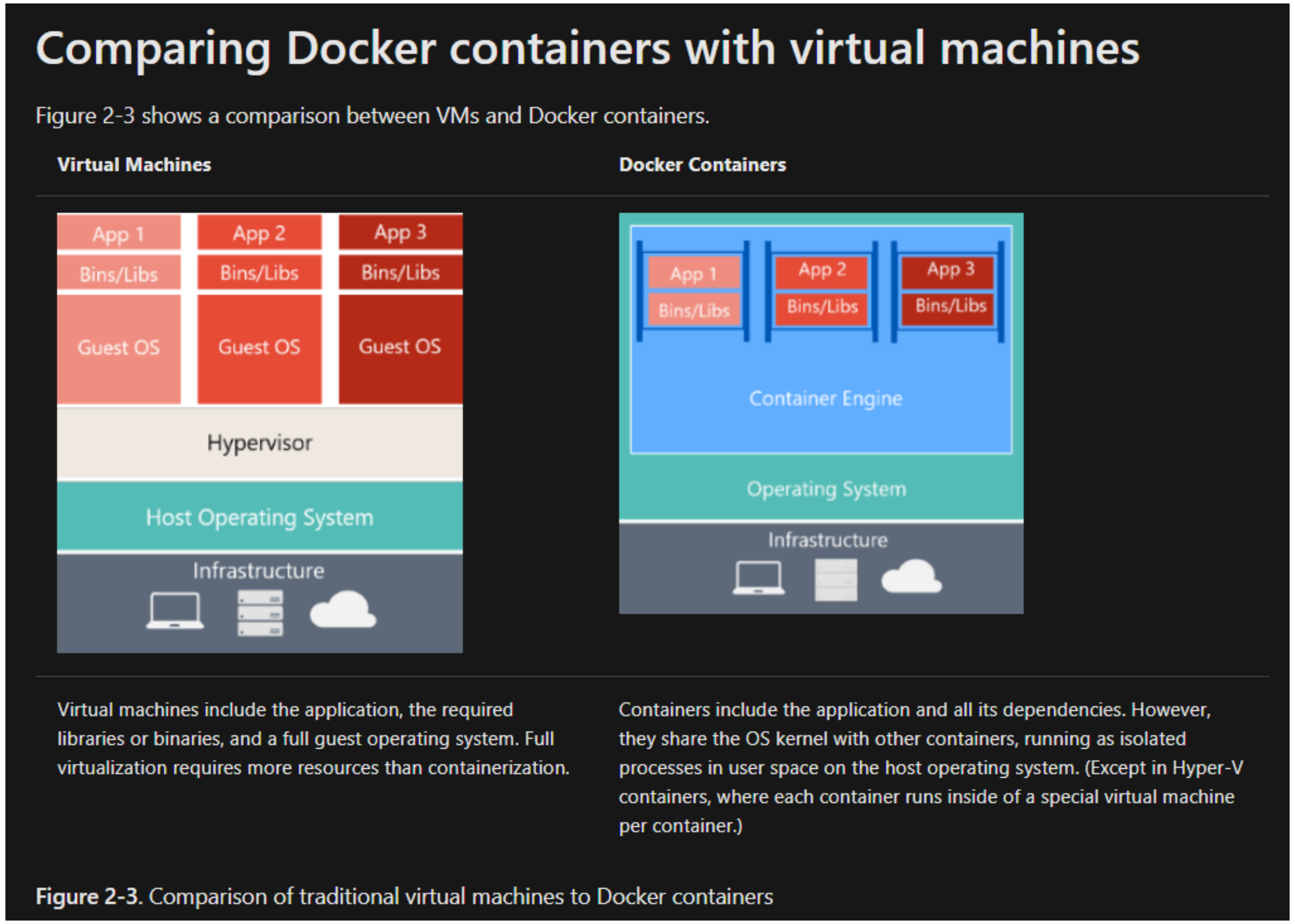
Docker works by providing a standard way to run your code. Docker is an operating system for containers. Similar to how a virtual machine virtualizes (removes the need to directly manage) server hardware, containers virtualize the operating system of a server. Docker is installed on each server and provides simple commands you can use to build, start, or stop containers.

AWS services such as AWS Fargate, Amazon ECS, Amazon EKS, and AWS Batch make it easy to run and manage Docker containers at scale.



A Beginner-Friendly Introduction to Containers, VMs and Docker (freecodecamp.org) What is Docker? | IBM

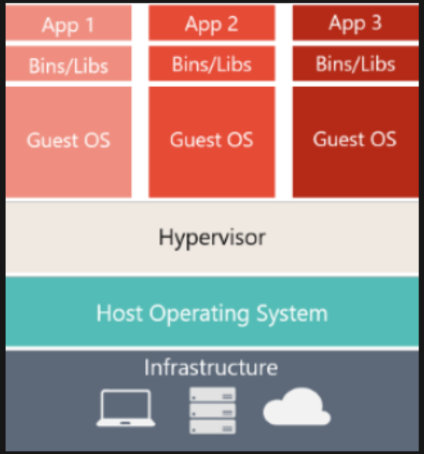
Docker Overview - Docker vs VM

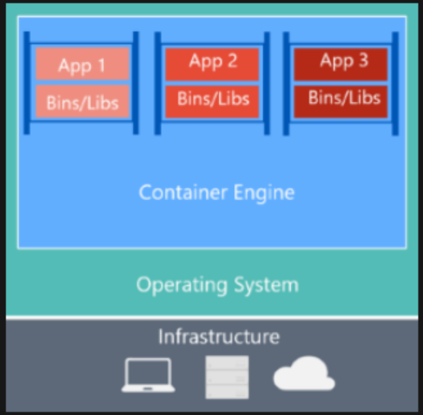


Comparing Docker containers with virtual machines

Figure 2-3 shows a comparison between VMs and Docker containers.

Virtual Machines | **Docker Containers**



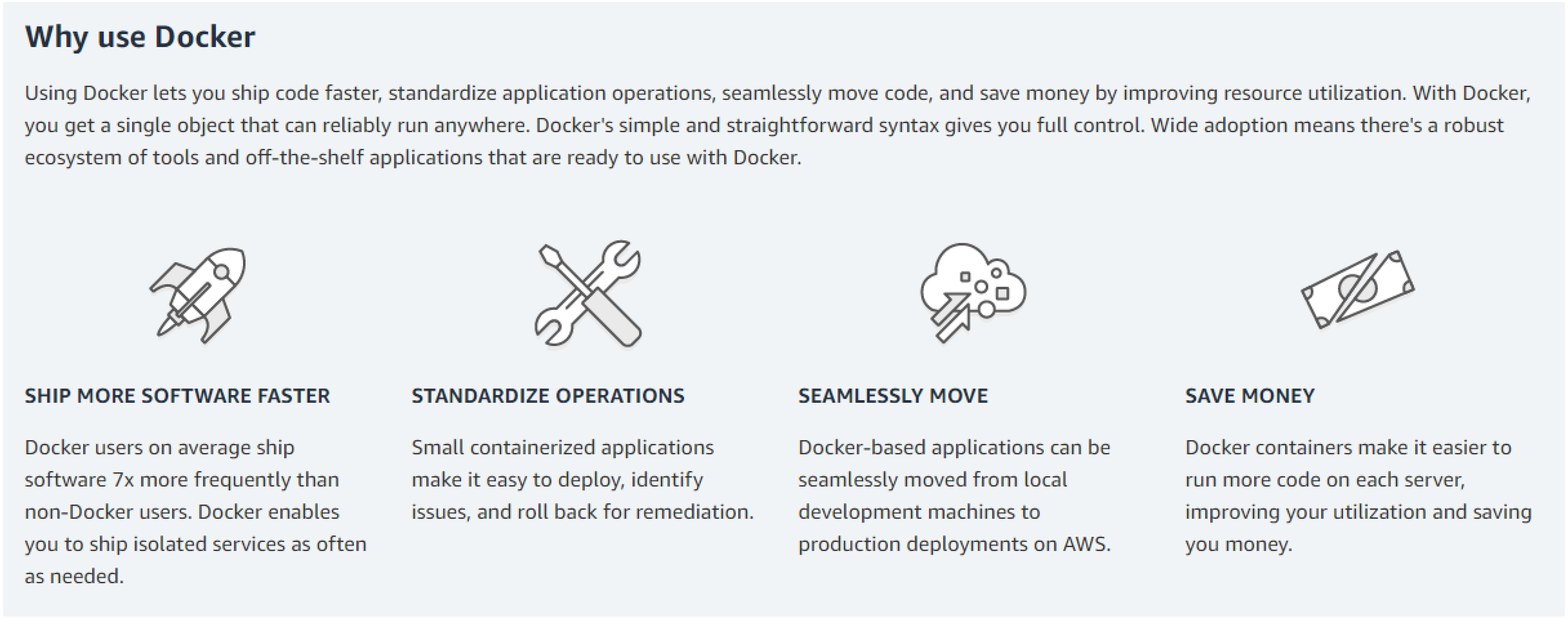


Virtual machines include the application, the required libraries or binaries, and a full guest operating system. Full virtualization requires more resources than containerization.

Containers include the application and all its dependencies. However, they share the OS kernel with other containers, running as isolated processes in user space on the host operating system. (Except in Hyper-V containers, where each container runs inside of a special virtual machine per container.)

Figure 2-3. Comparison of traditional virtual machines to Docker containers

Docker Overview - Why docker



Why use Docker

Using Docker lets you ship code faster, standardize application operations, seamlessly move code, and save money by improving resource utilization. With Docker, you get a single object that can reliably run anywhere. Docker's simple and straightforward syntax gives you full control. Wide adoption means there's a robust ecosystem of tools and off-the-shelf applications that are ready to use with Docker.

SHIP MORE SOFTWARE FASTER

Docker users on average ship software 7x more frequently than non-Docker users. Docker enables you to ship isolated services as often as needed.

STANDARDIZE OPERATIONS

Small containerized applications make it easy to deploy, identify issues, and roll back for remediation.

SEAMLESSLY MOVE

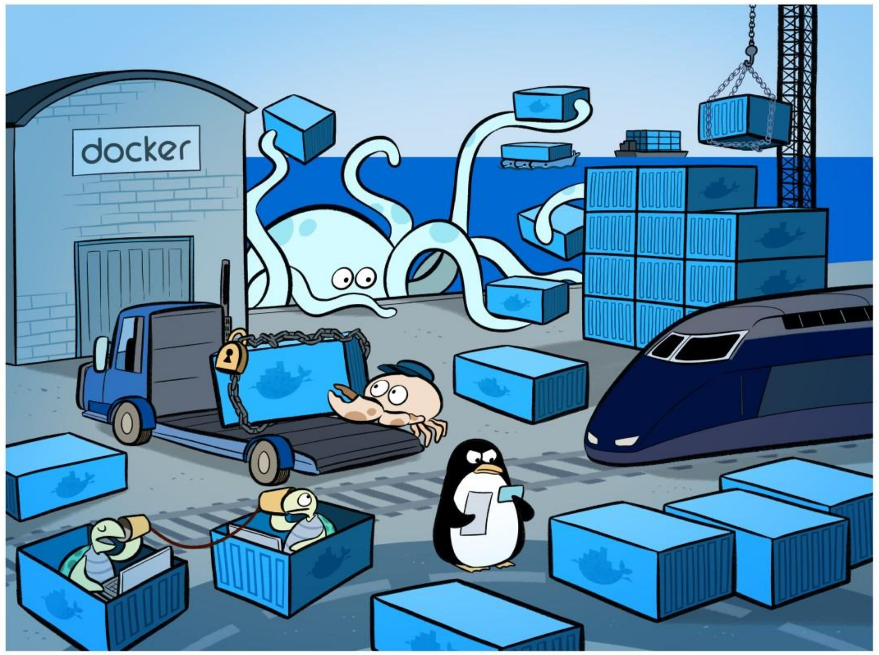
Docker-based applications can be seamlessly moved from local development machines to production deployments on AWS.

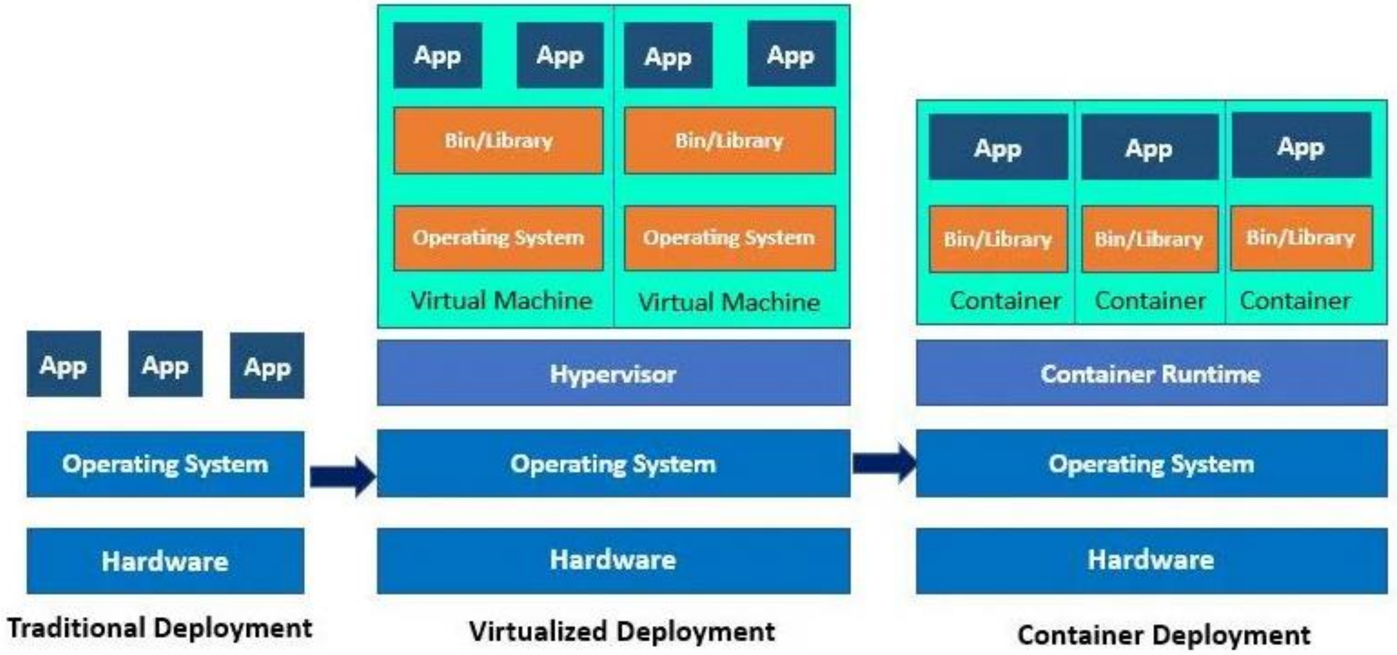
SAVE MONEY

Docker containers make it easier to run more code on each server, improving your utilization and saving you money.

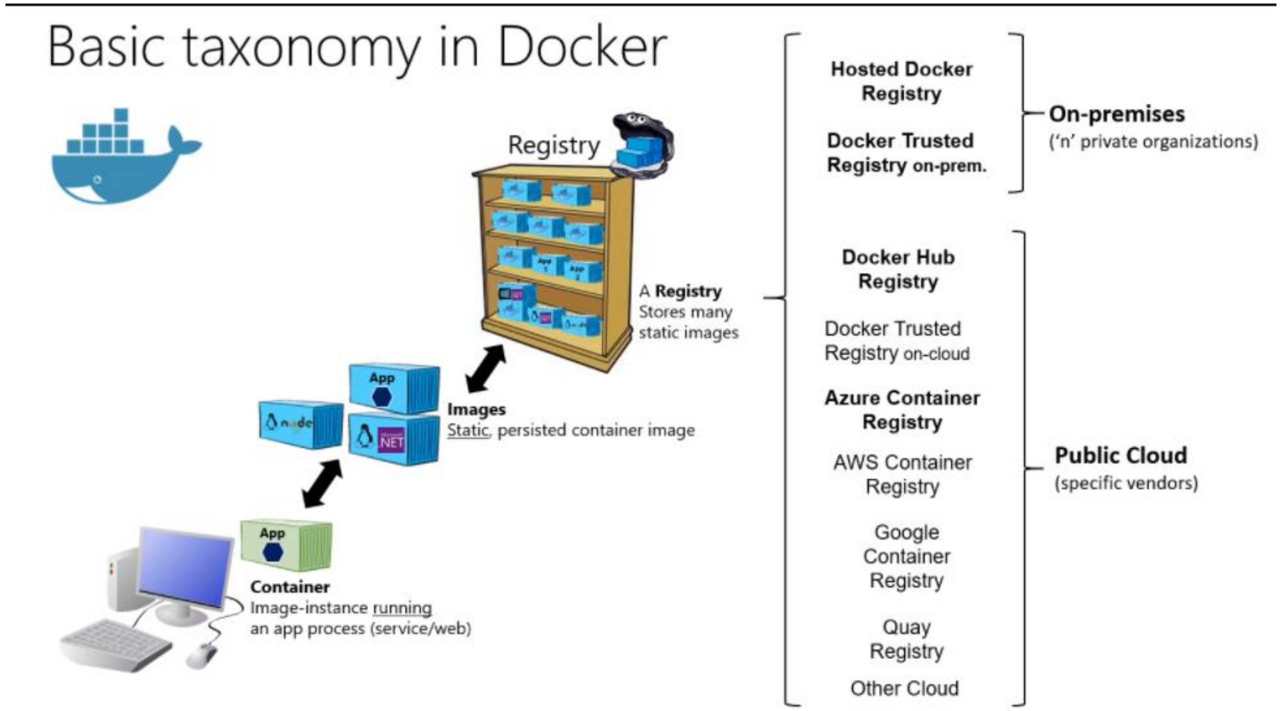
- build, ship, run anywhere

Docker Overview - Deployment

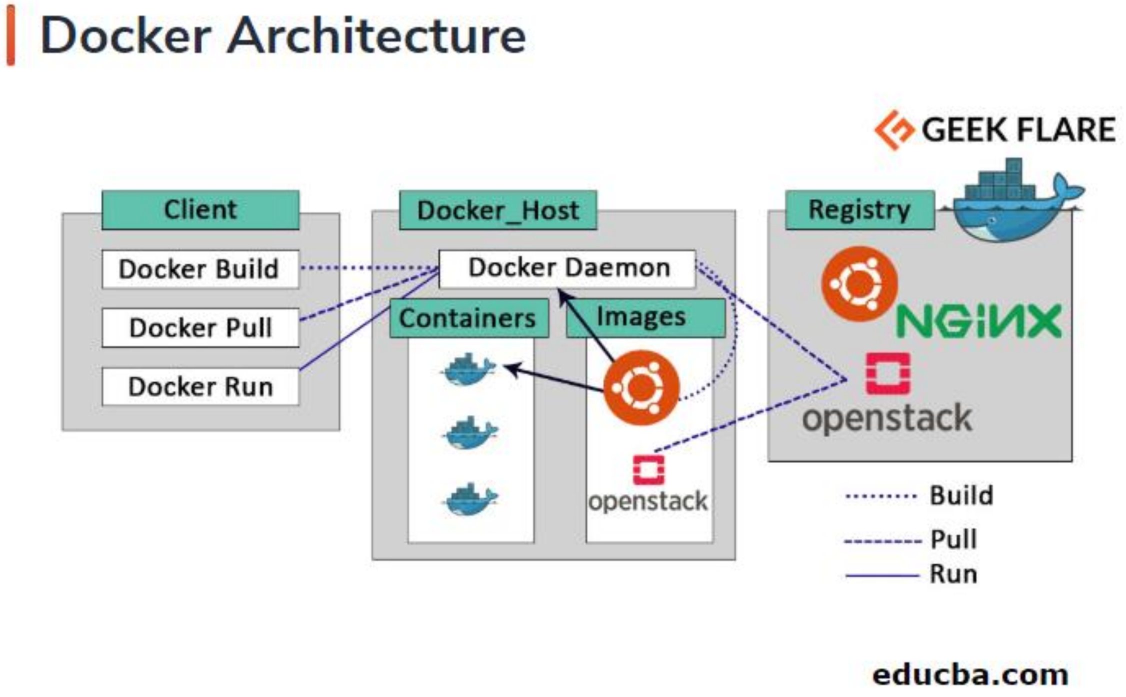




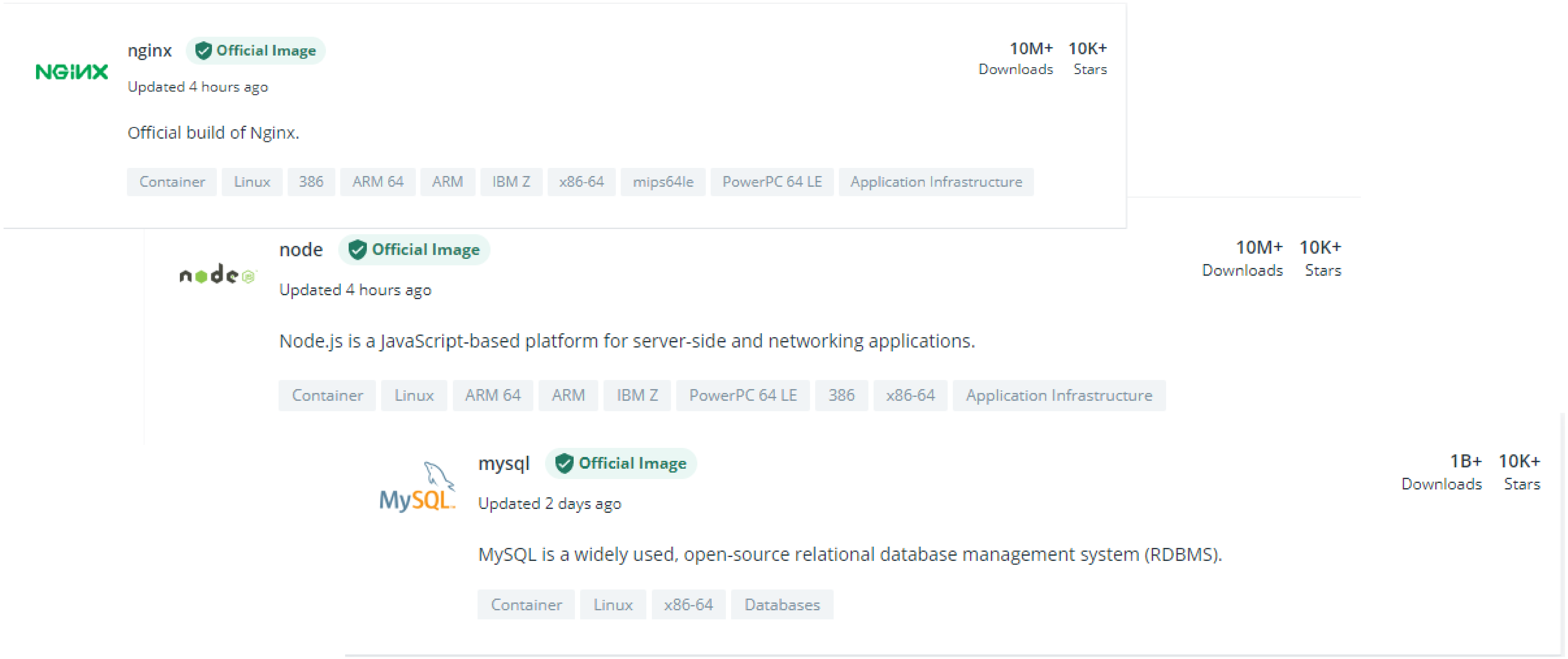
Docker Overview - Docker Architecture



Basic taxonomy in Docker
Registry
A Registry
Stores many static images
Images
Static, persisted container image
App
node
NET
Container
Image-instance running an app process (service/web)
Hosted Docker Registry
Docker Trusted Registry on-prem.
On-premises
('n' private organizations)
Docker Hub Registry
Docker Trusted Registry on-cloud
Azure Container Registry
AWS Container Registry
Google Container Registry
Quay Registry
Other Cloud
Public Cloud
(specific vendors)



Docker Overview - Registry: Docker hub



nginx Official Image

Updated 4 hours ago

Official build of Nginx.

10M+ Downloads 10K+ Stars

Container | Linux | 386 | ARM 64 | ARM | IBM Z | x86-64 | mips64le | PowerPC 64 LE | Application Infrastructure

node Official Image

Updated 4 hours ago

Node.js is a JavaScript-based platform for server-side and networking applications.

10M+ Downloads 10K+ Stars

Container | Linux | ARM 64 | ARM | IBM Z | PowerPC 64 LE | 386 | x86-64 | Application Infrastructure

mysql Official Image

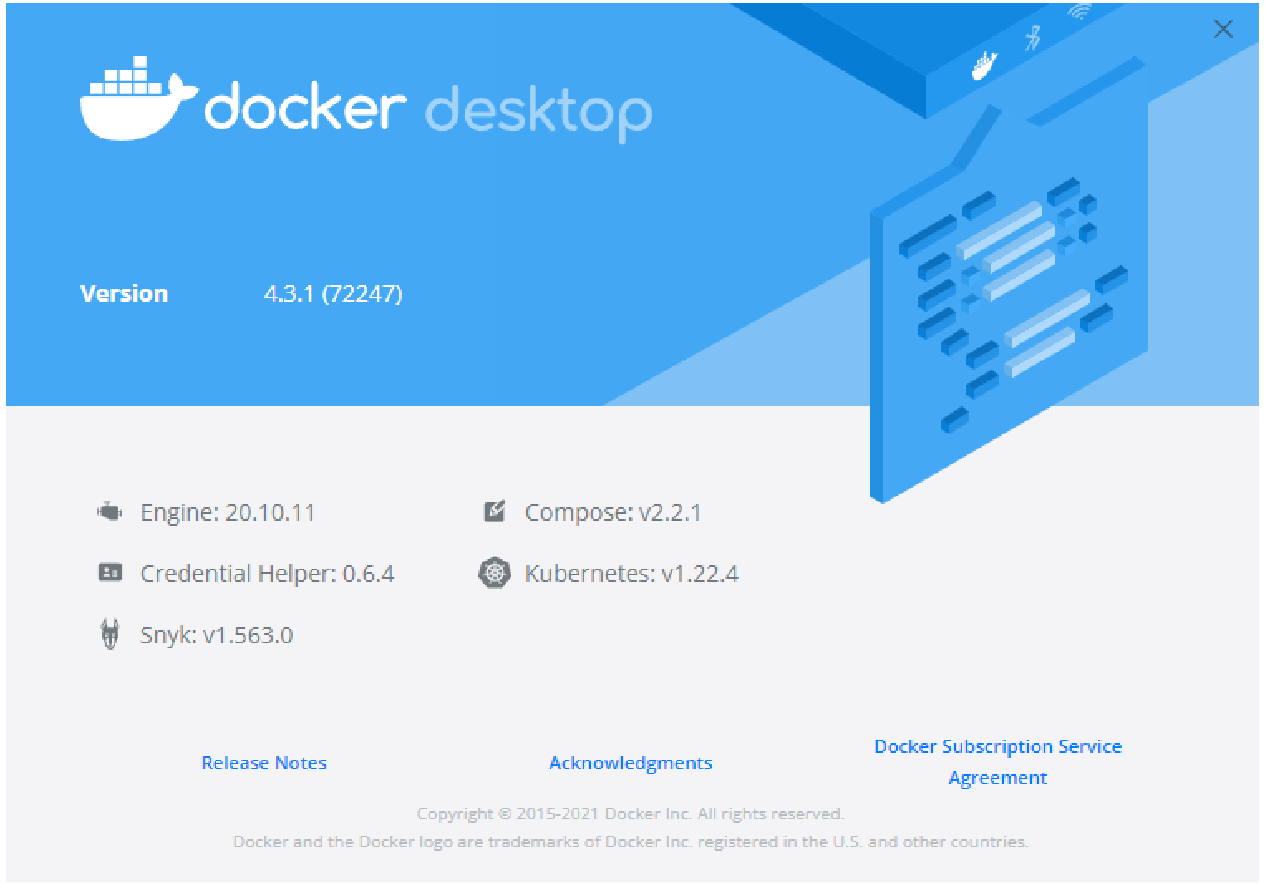
Updated 2 days ago

MySQL is a widely used, open-source relational database management system (RDBMS).

1B+ Downloads 10K+ Stars

Container | Linux | x86-64 | Databases

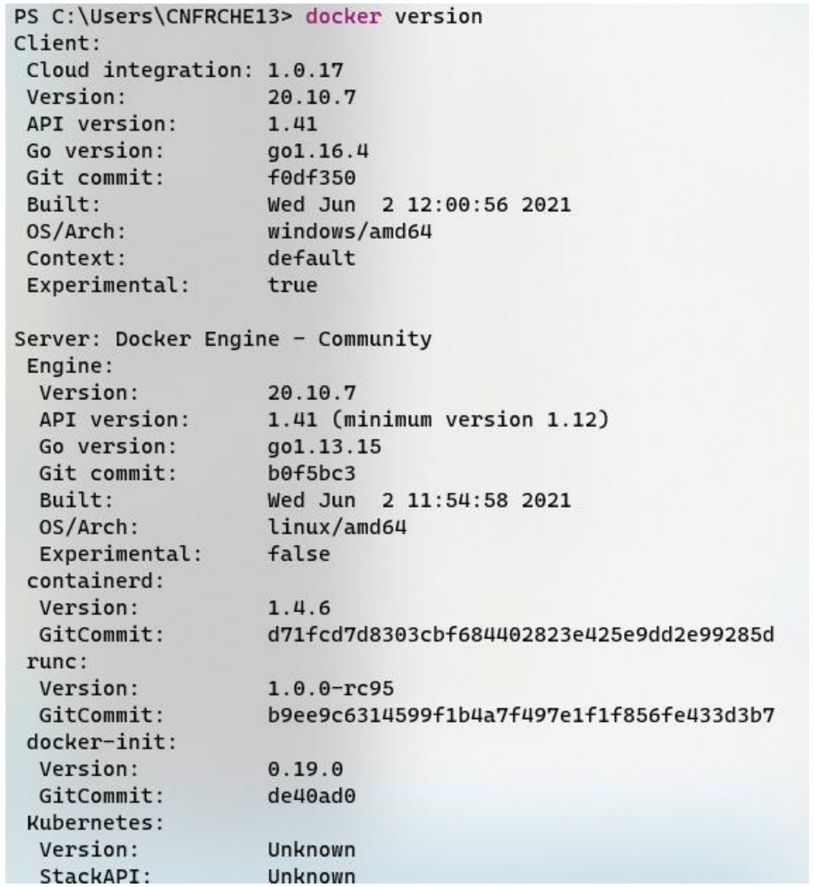
Docker Overview - Docker desktop



Version 4.3.1 (72247)

Engine: 20.10.11
Compose: v2.2.1
Credential Helper: 0.6.4
Kubernetes: v1.22.4
Snyk: v1.563.0

Release Notes
Acknowledgments
Docker Subscription Service Agreement

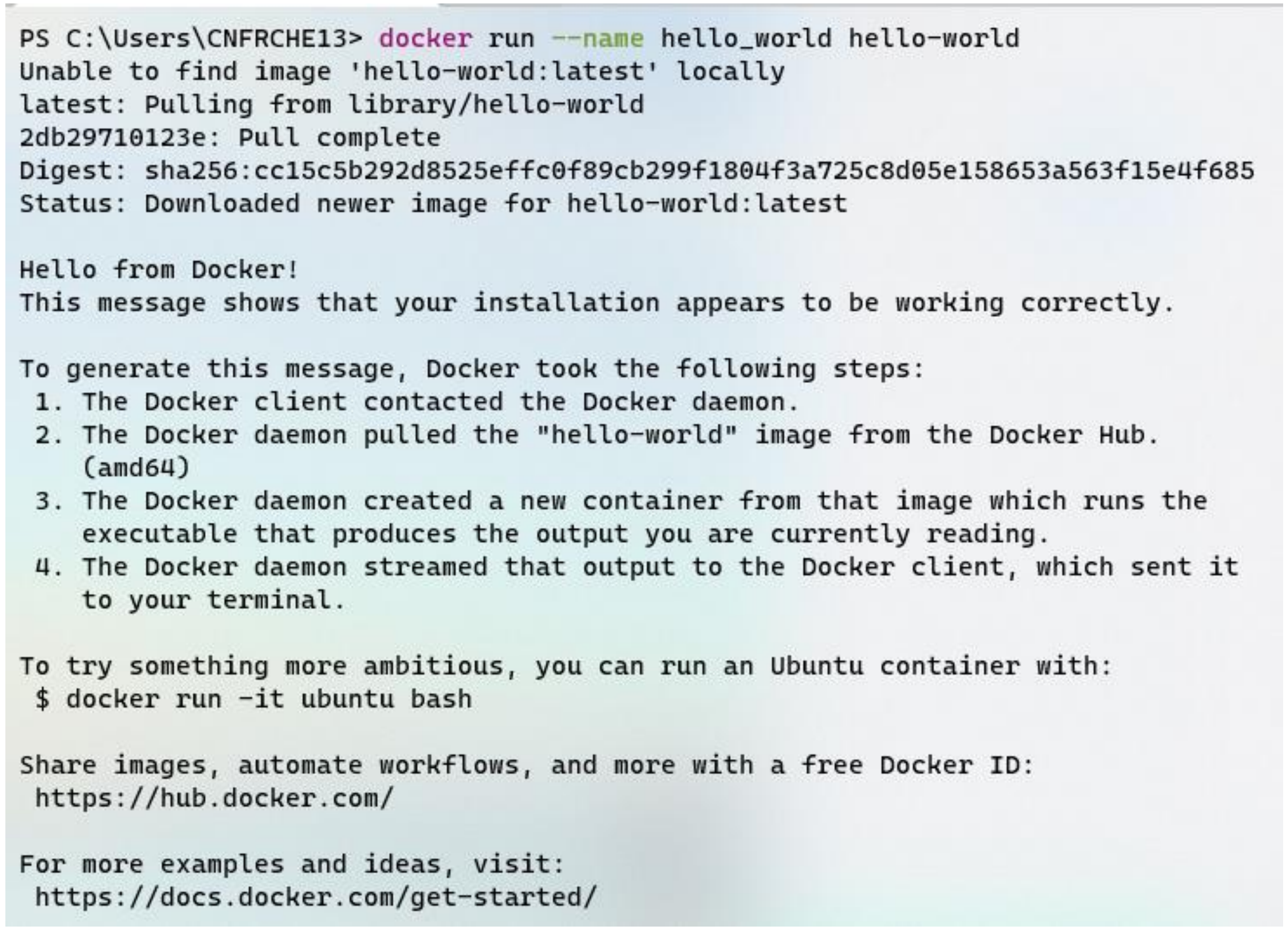


```
PS C:\Users\CNFRCHE13> docker version
Client:
 Cloud integration: 1.0.17
 Version:           20.10.7
 API version:       1.41
 Go version:        go1.16.4
 Git commit:        f0df350
 Built:             Wed Jun  2 12:00:56 2021
 OS/Arch:           windows/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.7
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.13.15
  Git commit:       b0f5bc3
  Built:            Wed Jun  2 11:54:58 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.6
  GitCommit:        d71fcd7d8303cbf684402823e425e9dd2e99285d
 runc:
  Version:          1.0.0-rc95
  GitCommit:        b9ee9c6314599f1b4a7f497e1f1f856fe433d3b7
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
 Kubernetes:
  Version:          Unknown
  StackAPI:         Unknown
```

Docker Overview – Sample: Hello world

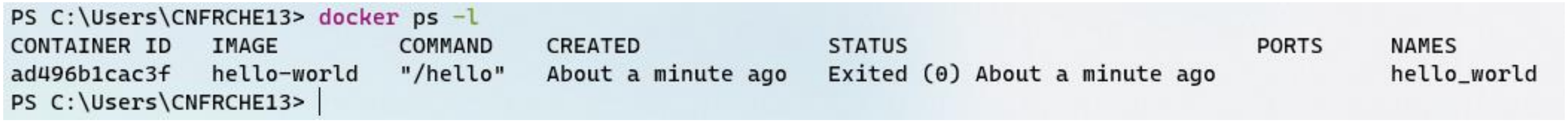
Used commands

- Docker run
- Docker restart
- Docker inspect
- Docker ls
- Docker rm

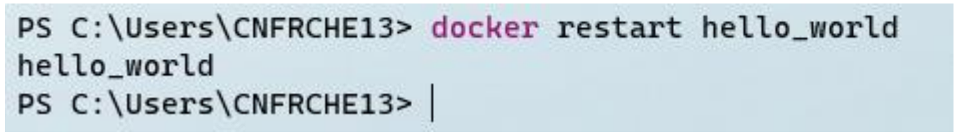


```
PS C:\Users\CNFRCHE13> docker run --name hello_world hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:cc15c5b292d8525effc0f89cb299f1804f3a725c8d05e158653a563f15e4f685
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Docker Overview – Sample: Hello World



```
PS C:\Users\CNFRCHE13> docker ps -l
CONTAINER ID   IMAGE         COMMAND    CREATED              STATUS                          PORTS     NAMES
ad496b1cac3f   hello-world   "/hello"   About a minute ago   Exited (0) About a minute ago             hello_world
PS C:\Users\CNFRCHE13>
```

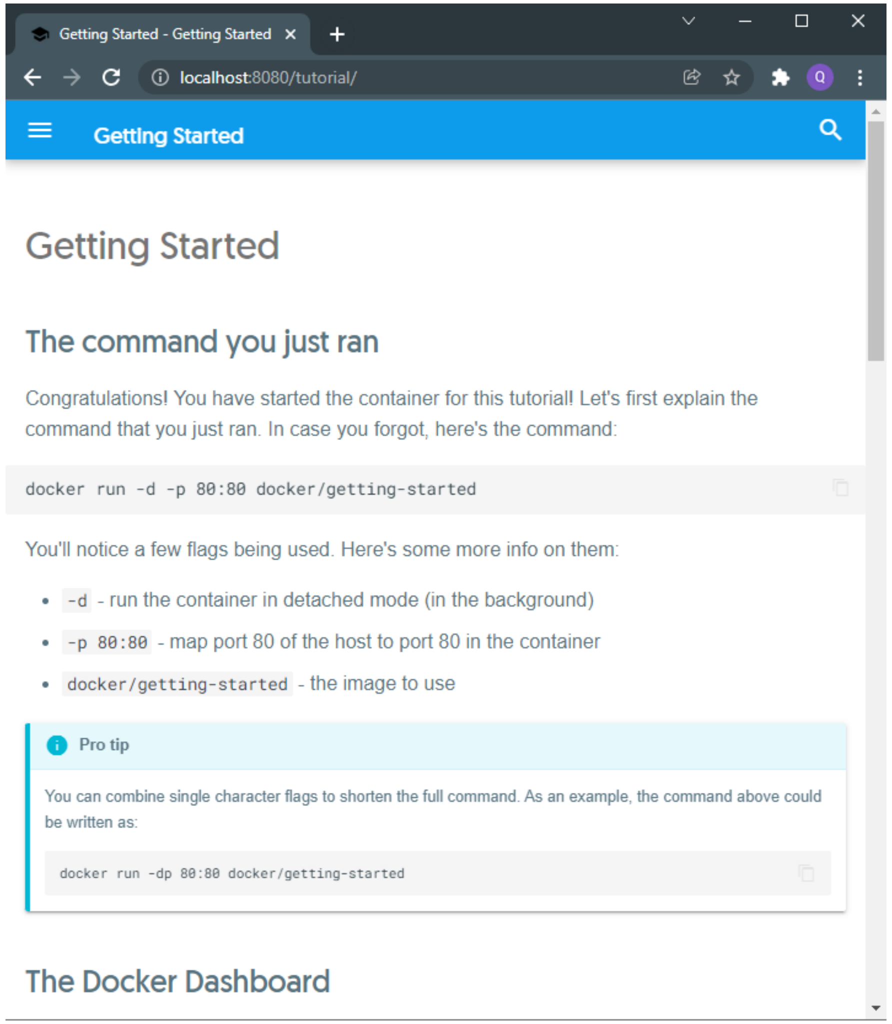


```
PS C:\Users\CNFRCHE13> docker restart hello_world
hello_world
PS C:\Users\CNFRCHE13>
```

Docker Overview – Sample: Docker/getting-started

Used commands

- Docker run
- Docker ps
- Docker exec



Getting Started

The command you just ran

Congratulations! You have started the container for this tutorial! Let's first explain the command that you just ran. In case you forgot, here's the command:

```
docker run -d -p 80:80 docker/getting-started
```

You'll notice a few flags being used. Here's some more info on them:

- `-d` - run the container in detached mode (in the background)
- `-p 80:80` - map port 80 of the host to port 80 in the container
- `docker/getting-started` - the image to use

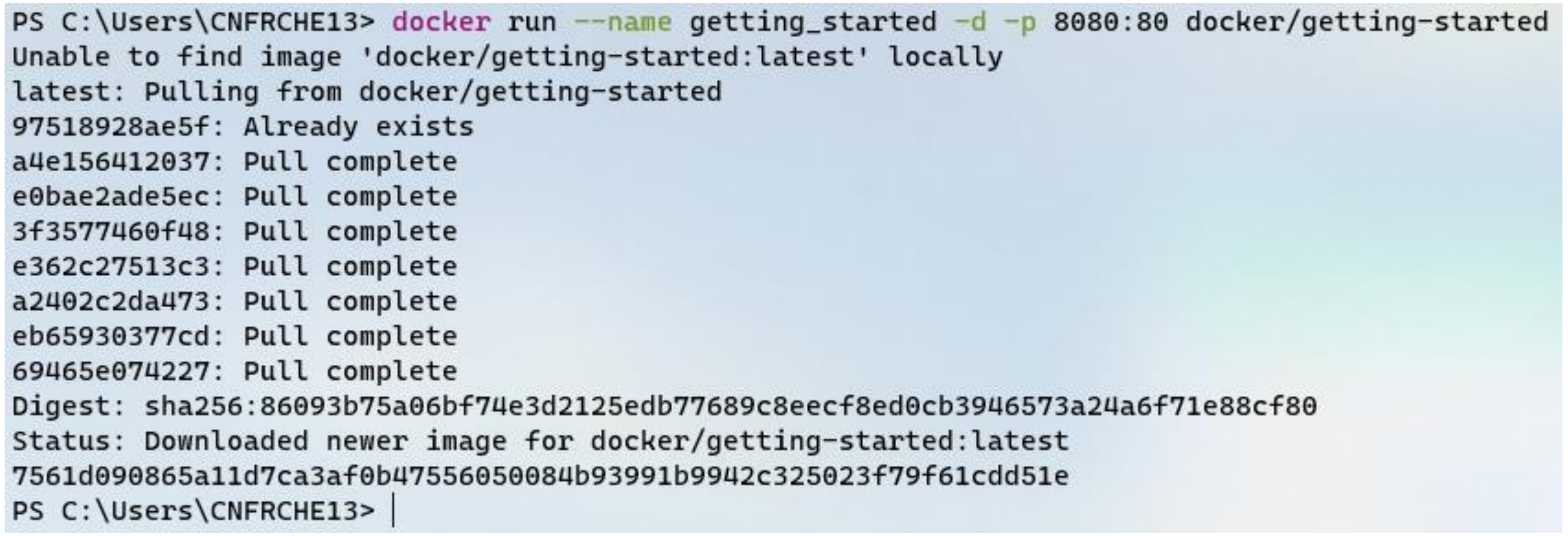
Pro tip

You can combine single character flags to shorten the full command. As an example, the command above could be written as:

```
docker run -dp 80:80 docker/getting-started
```

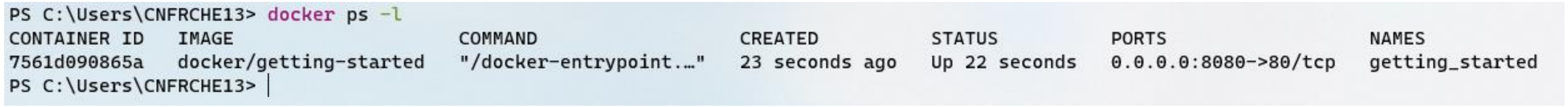
The Docker Dashboard

Docker Overview – Sample: Docker/getting-started

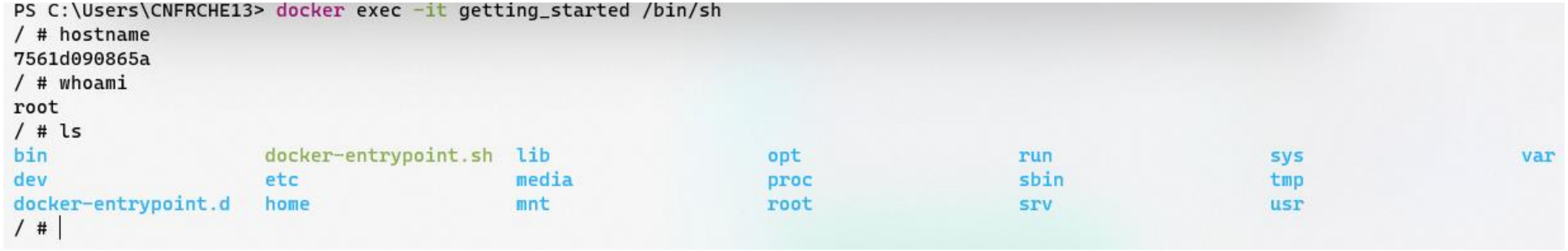


```
PS C:\Users\CNFRCHE13> docker run --name getting_started -d -p 8080:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
97518928ae5f: Already exists
a4e156412037: Pull complete
e0bae2ade5ec: Pull complete
3f3577460f48: Pull complete
e362c27513c3: Pull complete
a2402c2da473: Pull complete
eb65930377cd: Pull complete
69465e074227: Pull complete
Digest: sha256:86093b75a06bf74e3d2125edb77689c8eecf8ed0cb3946573a24a6f71e88cf80
Status: Downloaded newer image for docker/getting-started:latest
7561d090865a11d7ca3af0b47556050084b93991b9942c325023f79f61cdd51e
PS C:\Users\CNFRCHE13> |
```

Docker Overview – Sample: docker/getting-started



```
PS C:\Users\CNFRCHE13> docker ps -l
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS                  NAMES
7561d090865a   docker/getting-started   "/docker-entrypoint.…"   23 seconds ago   Up 22 seconds   0.0.0.0:8080->80/tcp   getting_started
PS C:\Users\CNFRCHE13>
```

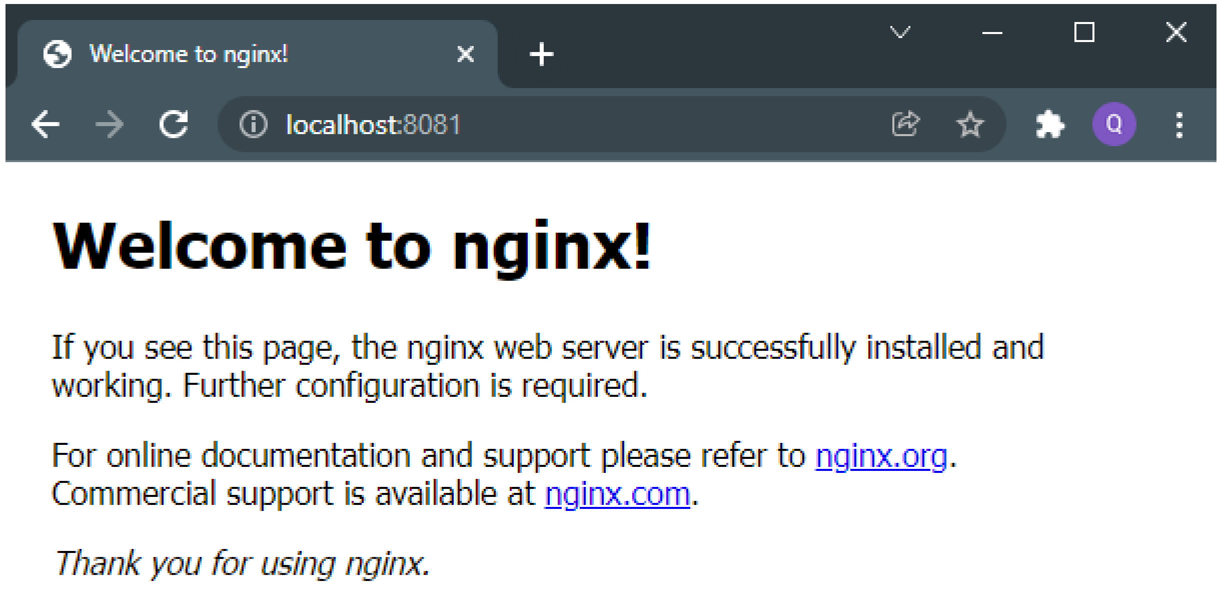


```
PS C:\Users\CNFRCHE13> docker exec -it getting_started /bin/sh
/ # hostname
7561d090865a
/ # whoami
root
/ # ls
bin                   docker-entrypoint.sh  lib                   opt                   run                   sys                   var
dev                   etc                   media                 proc                  sbin                  tmp
docker-entrypoint.d   home                  mnt                   root                  srv                   usr
/ #
```

Docker Overview – Sample: Nginx

Used commands

- Docker run
- Docker ps
- Docker logs
- Docker exec
- Docker cp

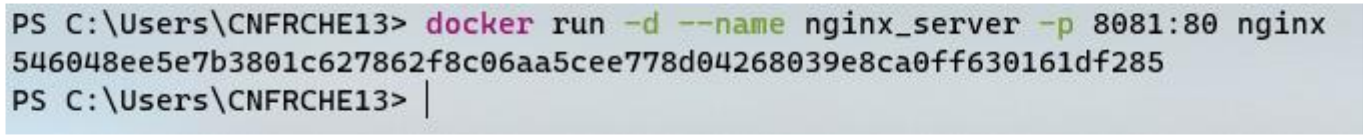


Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

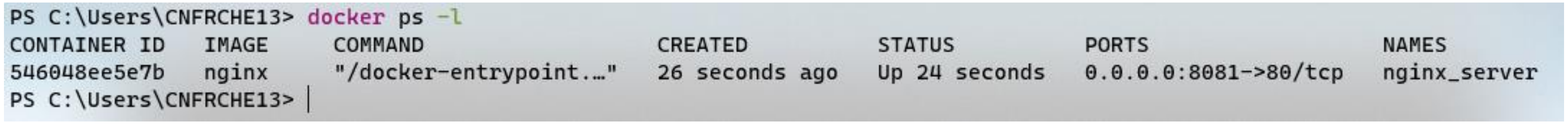
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

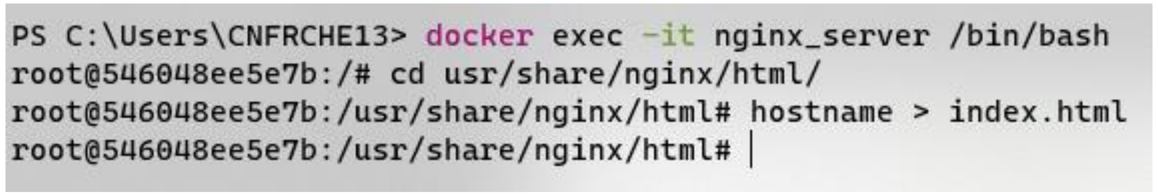


```
PS C:\Users\CNFRCHE13> docker run -d --name nginx_server -p 8081:80 nginx
546048ee5e7b3801c627862f8c06aa5cee778d04268039e8ca0ff630161df285
PS C:\Users\CNFRCHE13>
```

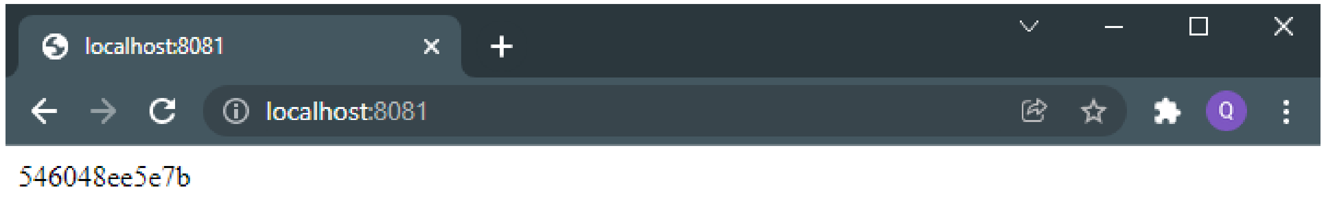
Docker Overview – Sample: Nginx



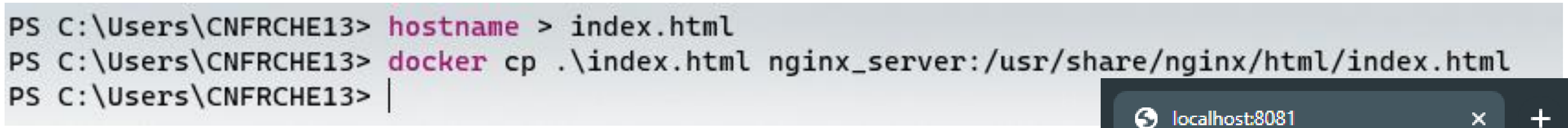
```
PS C:\Users\CNFRCHE13> docker ps -l
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                  NAMES
546048ee5e7b   nginx     "/docker-entrypoint.…"   26 seconds ago   Up 24 seconds   0.0.0.0:8081->80/tcp   nginx_server
PS C:\Users\CNFRCHE13>
```



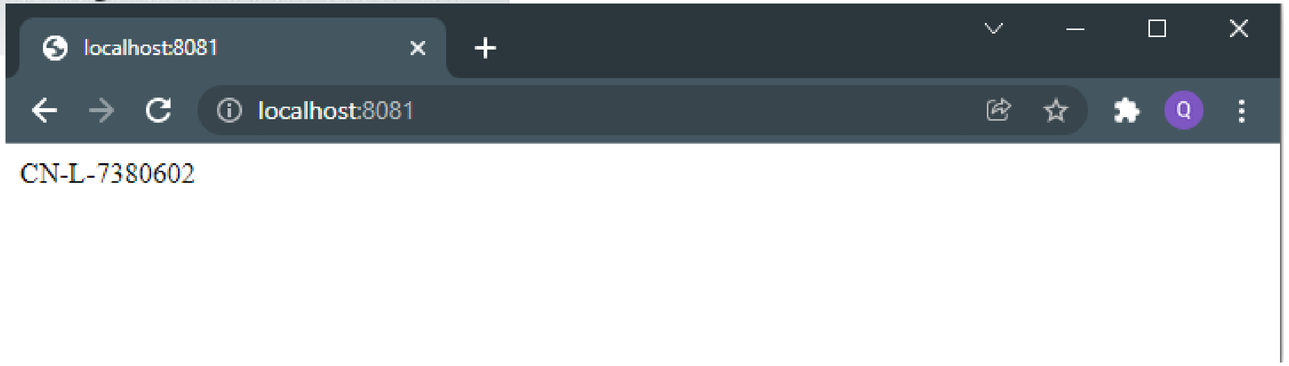
```
PS C:\Users\CNFRCHE13> docker exec -it nginx_server /bin/bash
root@546048ee5e7b:/# cd usr/share/nginx/html/
root@546048ee5e7b:/usr/share/nginx/html# hostname > index.html
root@546048ee5e7b:/usr/share/nginx/html#
```



546048ee5e7b

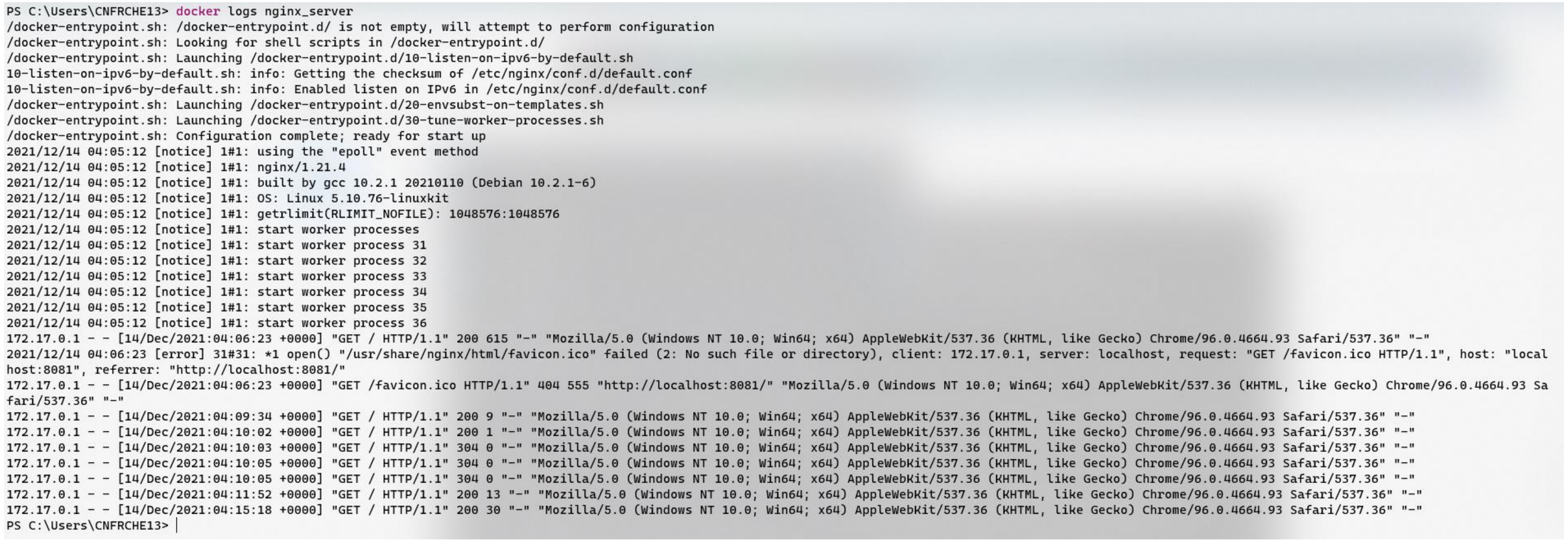


```
PS C:\Users\CNFRCHE13> hostname > index.html
PS C:\Users\CNFRCHE13> docker cp .\index.html nginx_server:/usr/share/nginx/html/index.html
PS C:\Users\CNFRCHE13>
```



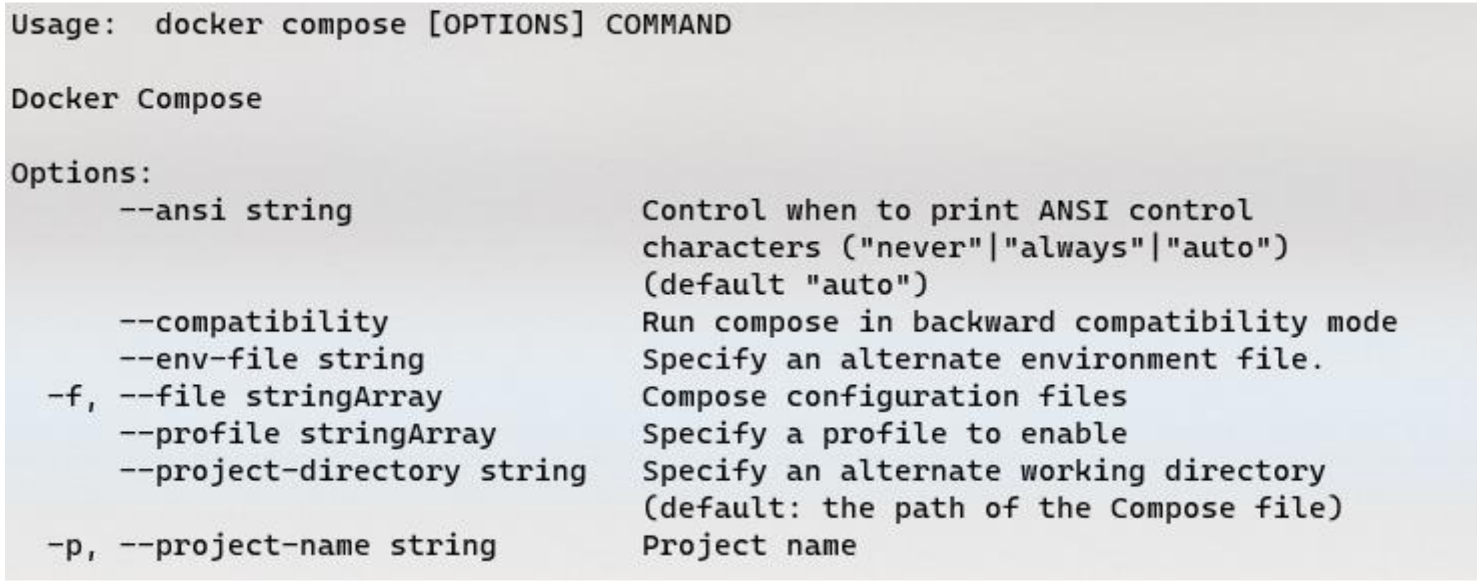
CN-L-7380602

Docker Overview – Sample: Nginx

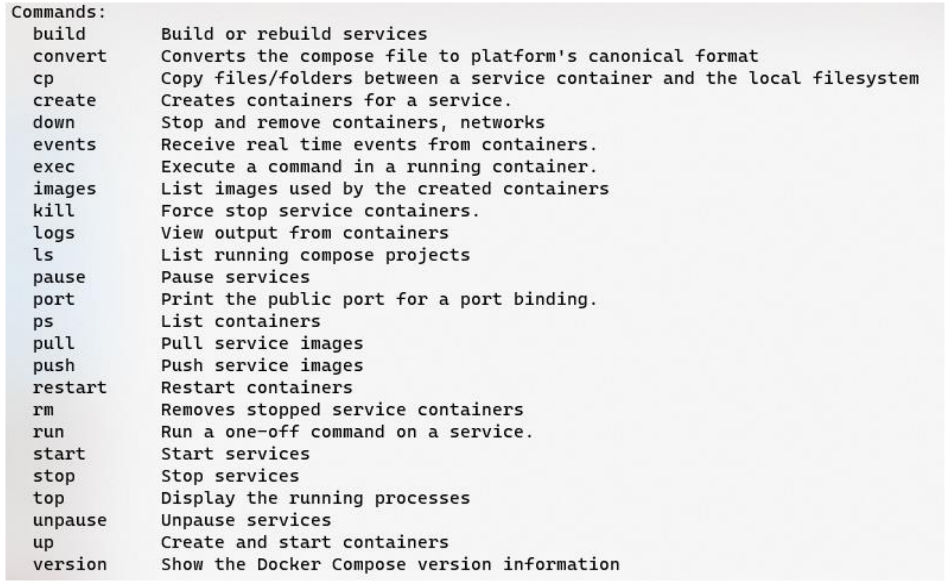


```
PS C:\Users\CNFRCHE13> docker logs nginx_server
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/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
2021/12/14 04:05:12 [notice] 1#1: using the "epoll" event method
2021/12/14 04:05:12 [notice] 1#1: nginx/1.21.4
2021/12/14 04:05:12 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2021/12/14 04:05:12 [notice] 1#1: OS: Linux 5.10.76-linuxkit
2021/12/14 04:05:12 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2021/12/14 04:05:12 [notice] 1#1: start worker processes
2021/12/14 04:05:12 [notice] 1#1: start worker process 31
2021/12/14 04:05:12 [notice] 1#1: start worker process 32
2021/12/14 04:05:12 [notice] 1#1: start worker process 33
2021/12/14 04:05:12 [notice] 1#1: start worker process 34
2021/12/14 04:05:12 [notice] 1#1: start worker process 35
2021/12/14 04:05:12 [notice] 1#1: start worker process 36
172.17.0.1 - - [14/Dec/2021:04:06:23 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36" "-"
2021/12/14 04:06:23 [error] 31#31: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 172.17.0.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "local
host:8081", referrer: "http://localhost:8081/"
172.17.0.1 - - [14/Dec/2021:04:06:23 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://localhost:8081/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Sa
fari/537.36" "-"
172.17.0.1 - - [14/Dec/2021:04:09:34 +0000] "GET / HTTP/1.1" 200 9 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36" "-"
172.17.0.1 - - [14/Dec/2021:04:10:02 +0000] "GET / HTTP/1.1" 200 1 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36" "-"
172.17.0.1 - - [14/Dec/2021:04:10:03 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36" "-"
172.17.0.1 - - [14/Dec/2021:04:10:05 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36" "-"
172.17.0.1 - - [14/Dec/2021:04:10:05 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36" "-"
172.17.0.1 - - [14/Dec/2021:04:11:52 +0000] "GET / HTTP/1.1" 200 13 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36" "-"
172.17.0.1 - - [14/Dec/2021:04:15:18 +0000] "GET / HTTP/1.1" 200 30 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36" "-"
PS C:\Users\CNFRCHE13> |
```

Docker Overview – Docker compose



```
Usage:  docker compose [OPTIONS] COMMAND

Docker Compose

Options:
      --ansi string                Control when to print ANSI control
                                   characters ("never"|"always"|"auto")
                                   (default "auto")
      --compatibility              Run compose in backward compatibility mode
      --env-file string            Specify an alternate environment file.
  -f, --file stringArray           Compose configuration files
      --profile stringArray        Specify a profile to enable
      --project-directory string   Specify an alternate working directory
                                   (default: the path of the Compose file)
  -p, --project-name string        Project name
```



```
Commands:
  build       Build or rebuild services
  convert     Converts the compose file to platform's canonical format
  cp          Copy files/folders between a service container and the local filesystem
  create      Creates containers for a service.
  down        Stop and remove containers, networks
  events      Receive real time events from containers.
  exec        Execute a command in a running container.
  images      List images used by the created containers
  kill        Force stop service containers.
  logs        View output from containers
  ls          List running compose projects
  pause       Pause services
  port        Print the public port for a port binding.
  ps          List containers
  pull        Pull service images
  push        Push service images
  restart     Restart containers
  rm          Removes stopped service containers
  run         Run a one-off command on a service.
  start       Start services
  stop        Stop services
  top         Display the running processes
  unpause     Unpause services
  up          Create and start containers
  version     Show the Docker Compose version information
```

Docker Overview – Docker compose(MySQL, Mongodb, RabbitMQ)

```yaml
version: '3.3'
services:

    todo_mysql:
        image: mysql
        volumes:
            - ./db/:/docker-entrypoint-initdb.d/
        environment:
            - "MYSQL_ROOT_PASSWORD=123456"
        ports:
            - "3306:3306"

    todo_mongodb:
        image:  mongo
        ports:
            - "27017:27017"

    todo_rabbitmq:
        hostname: todo_rabbitmq_host
        image: rabbitmq:management
        environment:
            - RABBITMQ_DEFAULT_USER=admin
            - RABBITMQ_DEFAULT_PASS=123456
        ports:
            - "5672:5672"
            - "15672:15672"

# networks:
#   default:
#       external: true
#       name: todo-app-test
```

- Yml
- Service name
- Image name
- Container name
- Volumes
- Environment variables
- Ports
- Expose
- Network (bridge, overlay, etc..)

Docker Overview – Dockerfile

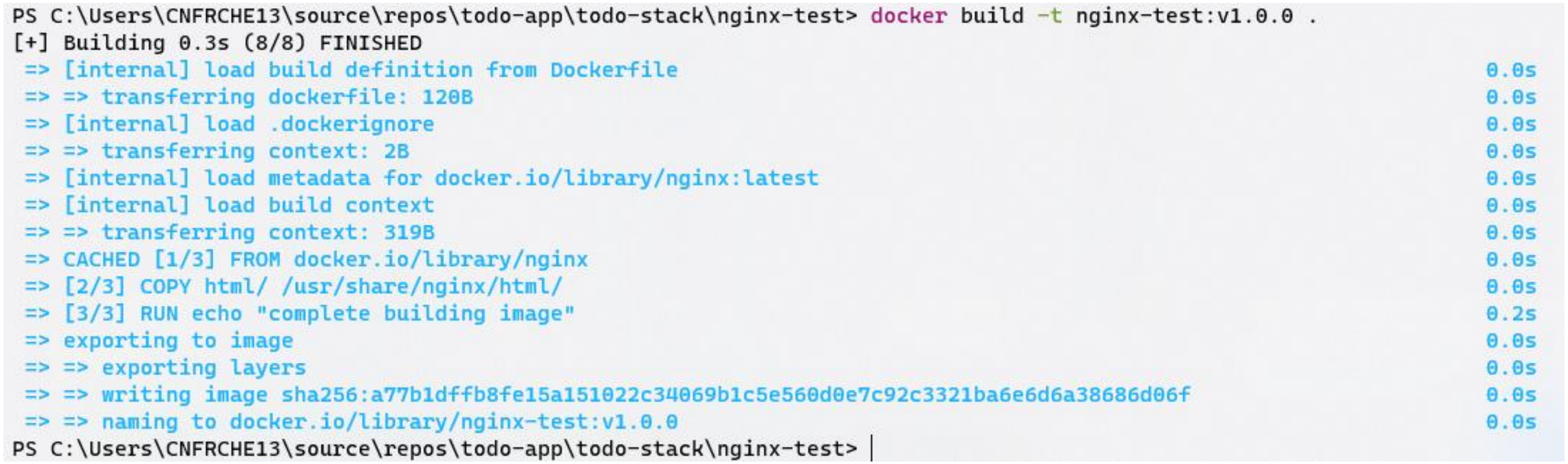
```
FROM nginx

COPY dist/todo-angular/ /usr/share/nginx/html/

COPY nginx.conf /etc/nginx/nginx.conf

RUN echo "complete building image"
```

```
FROM mcr.microsoft.com/dotnet/aspnet:3.1 AS base
WORKDIR /app
EXPOSE 80

FROM mcr.microsoft.com/dotnet/sdk:3.1 AS build
WORKDIR /src
COPY ["todo-api/todo-api.csproj", "todo-api/"]
RUN dotnet restore "todo-api/todo-api.csproj"
COPY . .
WORKDIR "/src/todo-api"
RUN dotnet build "todo-api.csproj" -c Release -r linux-x64 -f
netcoreapp3.1 -o /app/build

FROM build AS publish
RUN dotnet publish "todo-api.csproj" -c Release --self-contained false
-r linux-x64 -f netcoreapp3.1 -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "todo-api.dll"]
```

Docker Overview – Build image

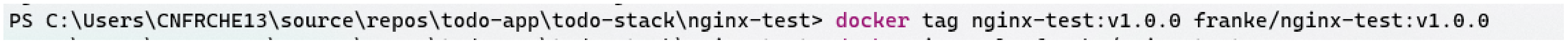
```
FROM nginx
COPY html/ /usr/share/nginx/html/
RUN echo "complete building image"
```

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Nginx Test</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
  <h1>nginx test</h1>
</body>
</html>
```

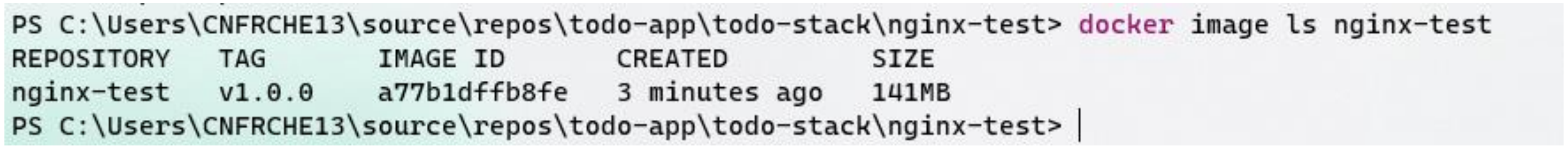


```
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-stack\nginx-test> docker build -t nginx-test:v1.0.0 .
[+] Building 0.3s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 120B                                                                          0.0s
 => [internal] load .dockerignore                                                                             0.0s
 => => transferring context: 2B                                                                               0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                                               0.0s
 => [internal] load build context                                                                             0.0s
 => => transferring context: 319B                                                                             0.0s
 => CACHED [1/3] FROM docker.io/library/nginx                                                                 0.0s
 => [2/3] COPY html/ /usr/share/nginx/html/                                                                   0.0s
 => [3/3] RUN echo "complete building image"                                                                  0.2s
 => exporting to image                                                                                        0.0s
 => => exporting layers                                                                                       0.0s
 => => writing image sha256:a77b1dffb8fe15a151022c34069b1c5e560d0e7c92c3321ba6e6d6a38686d06f                  0.0s
 => => naming to docker.io/library/nginx-test:v1.0.0                                                          0.0s
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-stack\nginx-test>
```

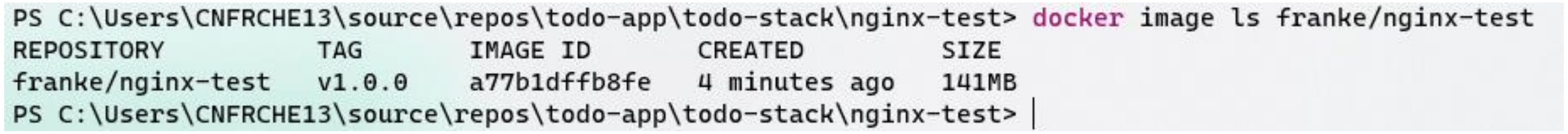
Docker Overview – Build image



```
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-stack\nginx-test> docker tag nginx-test:v1.0.0 franke/nginx-test:v1.0.0
```

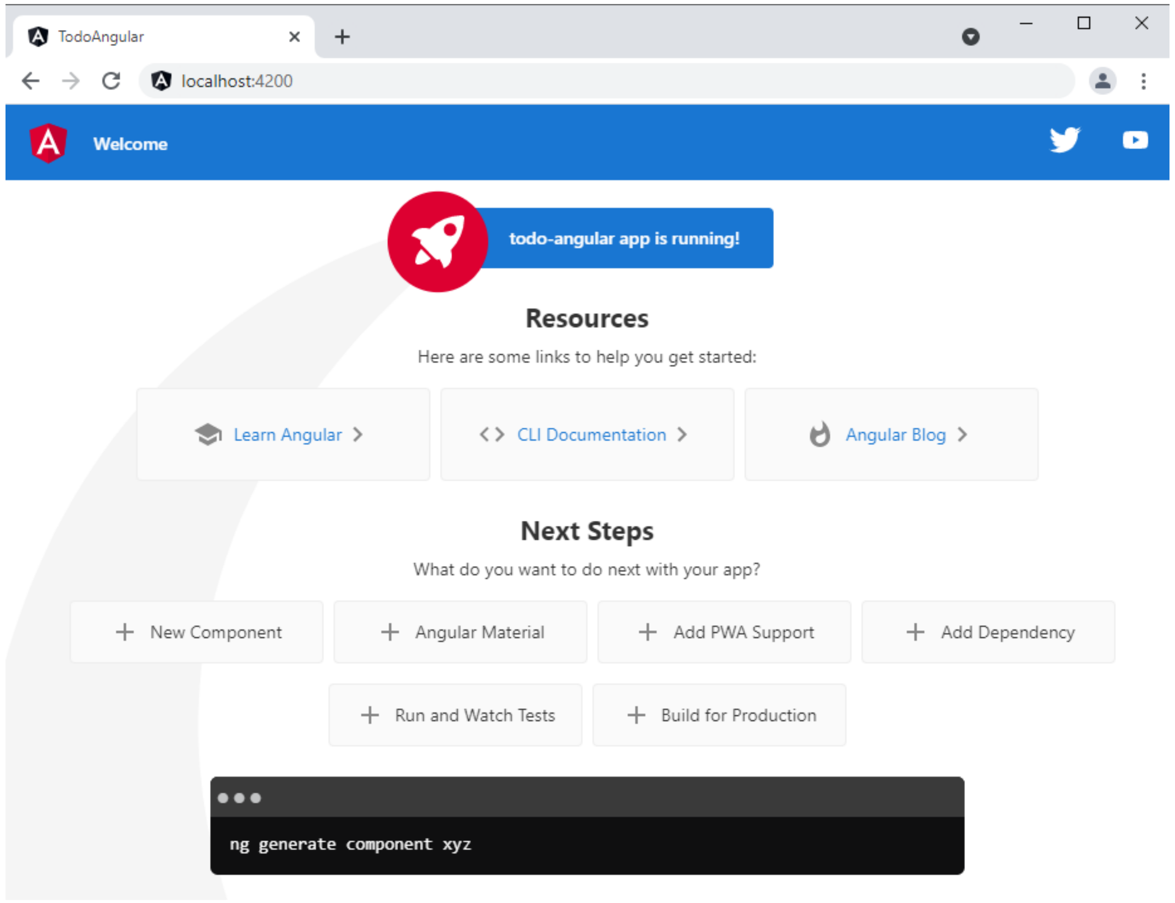


```
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-stack\nginx-test> docker image ls nginx-test
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
nginx-test   v1.0.0    a77b1dffb8fe   3 minutes ago   141MB
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-stack\nginx-test> |
```



```
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-stack\nginx-test> docker image ls franke/nginx-test
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
franke/nginx-test   v1.0.0    a77b1dffb8fe   4 minutes ago   141MB
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-stack\nginx-test> |
```

TODO App Overview



TodoAngular

localhost:4200

Welcome

todo-angular app is running!

Resources

Here are some links to help you get started:

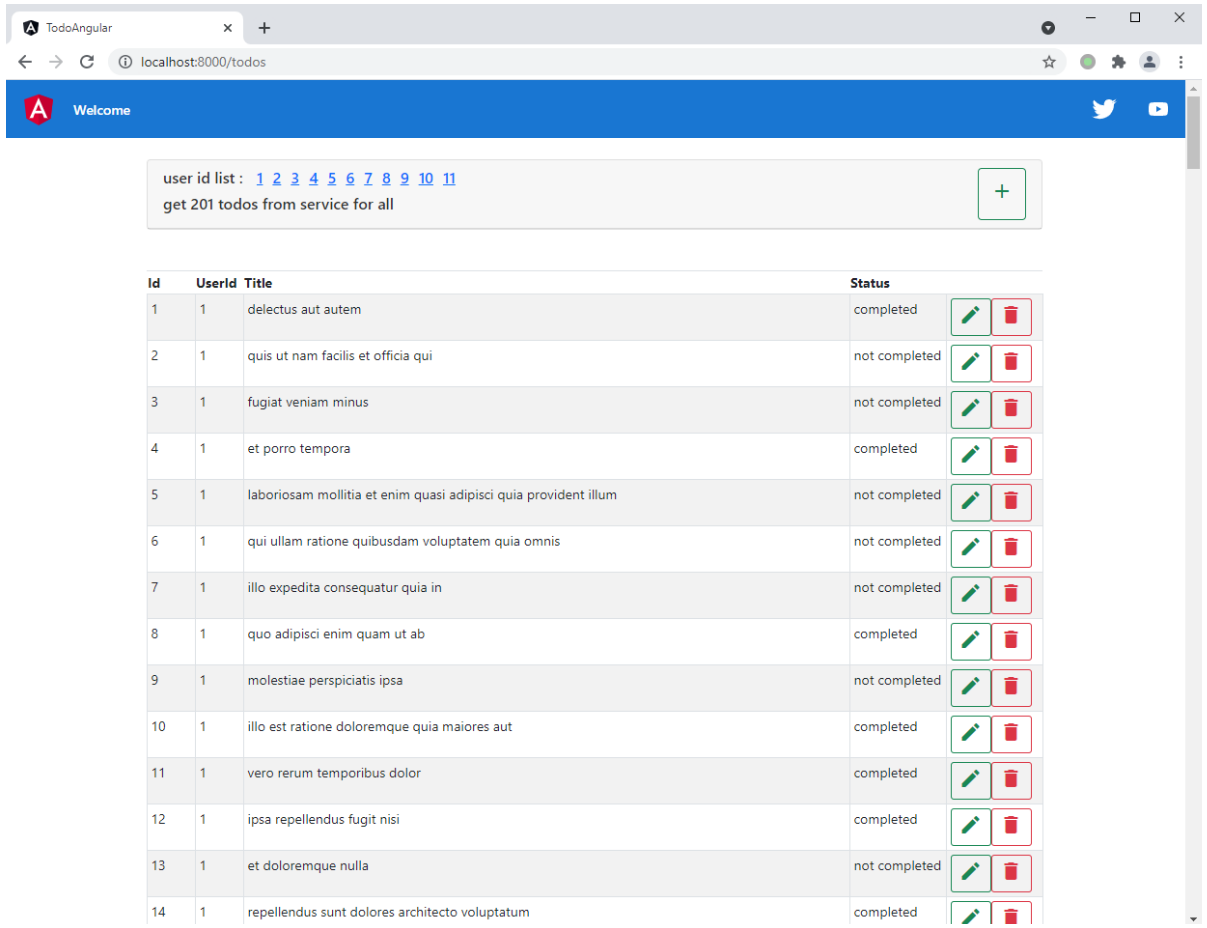
Learn Angular > | CLI Documentation > | Angular Blog >

Next Steps

What do you want to do next with your app?

+ New Component | + Angular Material | + Add PWA Support | + Add Dependency | + Run and Watch Tests | + Build for Production

```
ng generate component xyz
```



TodoAngular

localhost:8000/todos

Welcome

user id list : 1 2 3 4 5 6 7 8 9 10 11

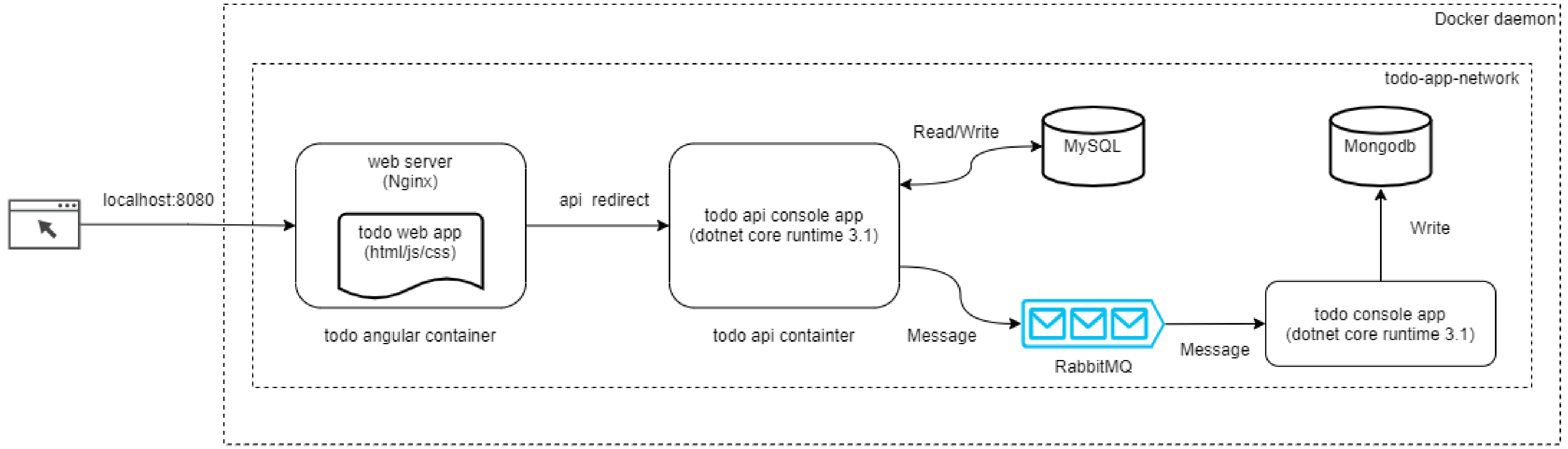
get 201 todos from service for all

Id	UserId	Title	Status
1	1	delectus aut autem	completed
2	1	quis ut nam facilis et officia qui	not completed
3	1	fugiat veniam minus	not completed
4	1	et porro tempora	completed
5	1	laboriosam mollitia et enim quasi adipisci quia provident illum	not completed
6	1	qui ullam ratione quibusdam voluptatem quia omnis	not completed
7	1	illo expedita consequatur quia in	not completed
8	1	quo adipisci enim quam ut ab	completed
9	1	molestiae perspiciatis ipsa	not completed
10	1	illo est ratione doloremque quia maiores aut	completed
11	1	vero rerum temporibus dolor	completed
12	1	ipsa repellendus fugit nisi	completed
13	1	et doloremque nulla	not completed
14	1	repellendus sunt dolores architecto voluptatum	completed

TODO App Overview - Architecture

Main stacks

- MySQL
- Mongodb
- RabbitMQ
- Nginx
- Angular
- Asp Net Core Web API
- Docker
- Docker Compose

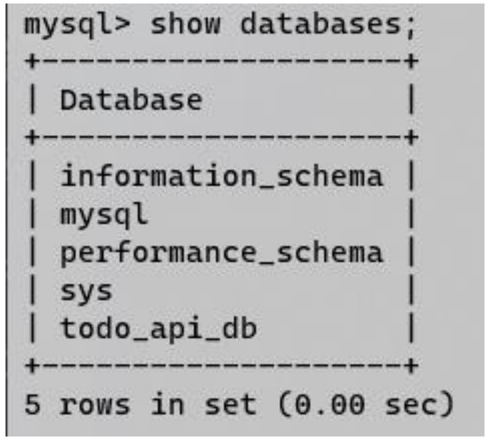


TODO App Overview - Data source

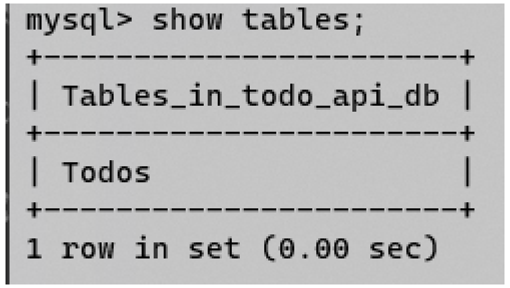
Data source: https://jsonplaceholder.typicode.com/todos

```
CREATE TABLE `Todos` (
  `Id` int unsigned NOT NULL AUTO_INCREMENT,
  `UserId` int unsigned DEFAULT NULL,
  `Title` varchar(100) DEFAULT NULL,
  `Completed` tinyint DEFAULT NULL,
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8mb3;
```

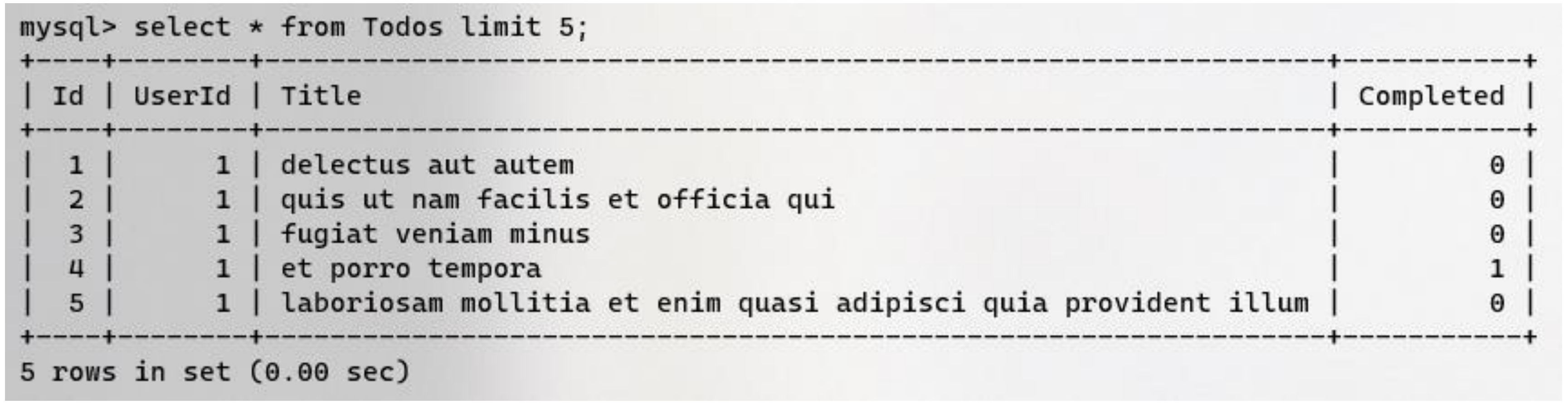
```
Sample: [
{
   "userId": 1,
   "id": 1,
   "title": "delectus aut autem",
   "completed": false
 },
{
   "userId": 1,
   "id": 2,
   "title": "quis ut nam facilis et officia qui",
   "completed": false
  }
]
```



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| todo_api_db        |
+--------------------+
5 rows in set (0.00 sec)
```



```
mysql> show tables;
+-----------------------+
| Tables_in_todo_api_db |
+-----------------------+
| Todos                 |
+-----------------------+
1 row in set (0.00 sec)
```



```
mysql> select * from Todos limit 5;
+----+--------+-----------------------------------------------------+-----------+
| Id | UserId | Title                                               | Completed |
+----+--------+-----------------------------------------------------+-----------+
|  1 |      1 | delectus aut autem                                  |         0 |
|  2 |      1 | quis ut nam facilis et officia qui                  |         0 |
|  3 |      1 | fugiat veniam minus                                 |         0 |
|  4 |      1 | et porro tempora                                    |         1 |
|  5 |      1 | laboriosam mollitia et enim quasi adipisci quia provident illum |         0 |
+----+--------+-----------------------------------------------------+-----------+
5 rows in set (0.00 sec)
```

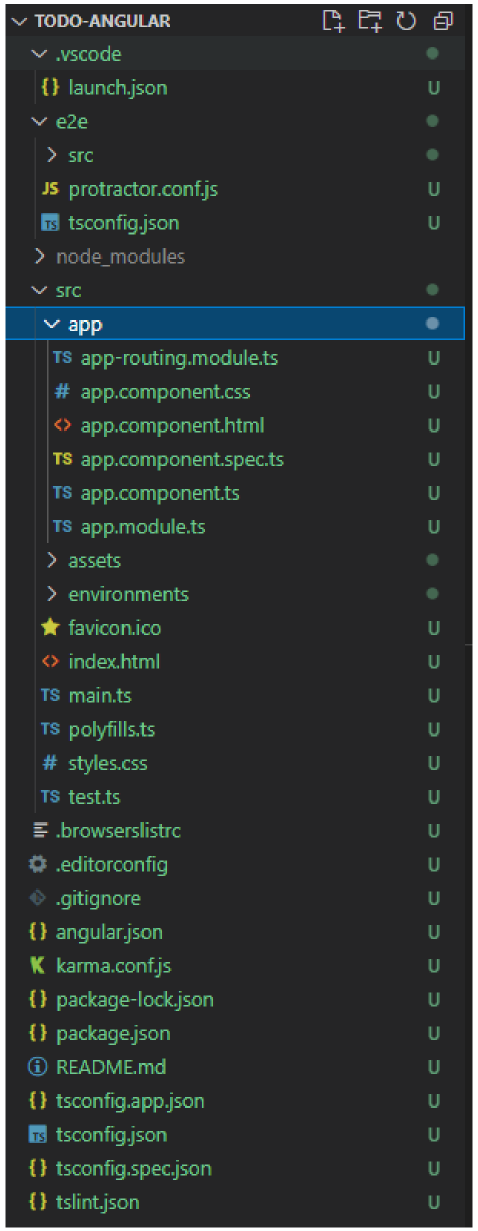
TODO Angular

- Ng new
- Ng serve
- Ng test
- Ng lint
- Ng build
- Proxy
- Dockerfile
- Nginx.conf
- Docker build

```
{
  "/api": {
    "target": "http://localhost:5000",
    "secure": true
  }
}
```

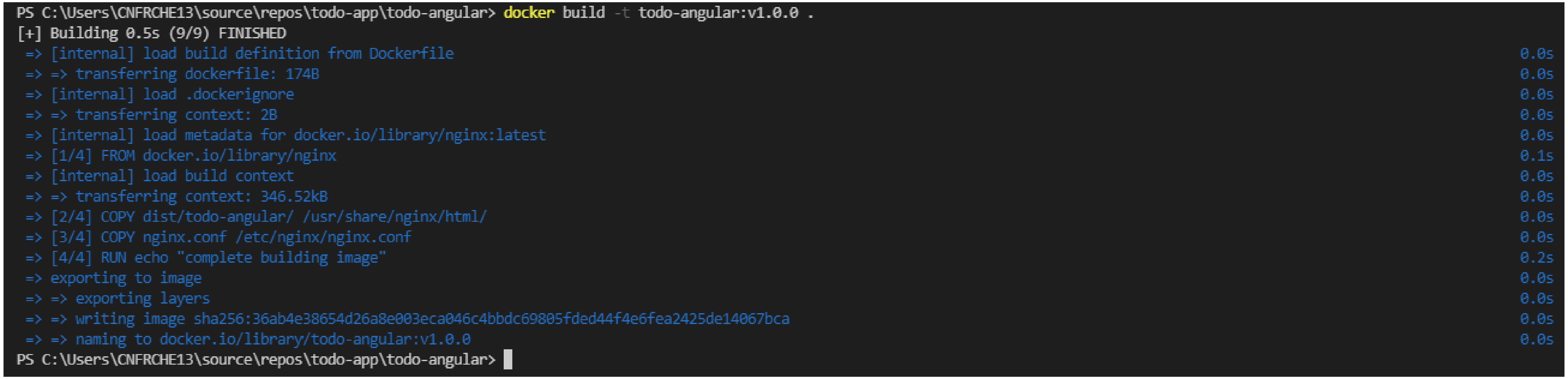
```
FROM nginx
COPY dist/todo-angular/ /usr/share/nginx/html/
COPY nginx.conf /etc/nginx/nginx.conf
RUN echo "complete building image"
```

```
  server {

    listen 80;
    server_name localhost; # dmmain name

    location  / {
      root /usr/share/nginx/html;
      index index.html;
      try_files $uri $uri/ /index.html;
    }

    location /api {
      proxy_pass  http://todo_api:80; # gateway port
    }
  }
```



```
TODO-ANGULAR
  .vscode
    launch.json                U
  e2e
    src
    protractor.conf.js         U
    tsconfig.json              U
  node_modules
  src
    app
      app-routing.module.ts    U
      app.component.css        U
      app.component.html       U
      app.component.spec.ts    U
      app.component.ts         U
      app.module.ts            U
    assets
    environments
    favicon.ico                U
    index.html                 U
    main.ts                    U
    polyfills.ts               U
    styles.css                 U
    test.ts                    U
  .browserslistrc              U
  .editorconfig                U
  .gitignore                   U
  angular.json                 U
  karma.conf.js                U
  package-lock.json            U
  package.json                 U
  README.md                    U
  tsconfig.app.json            U
  tsconfig.json                U
  tsconfig.spec.json           U
  tslint.json                  U
```

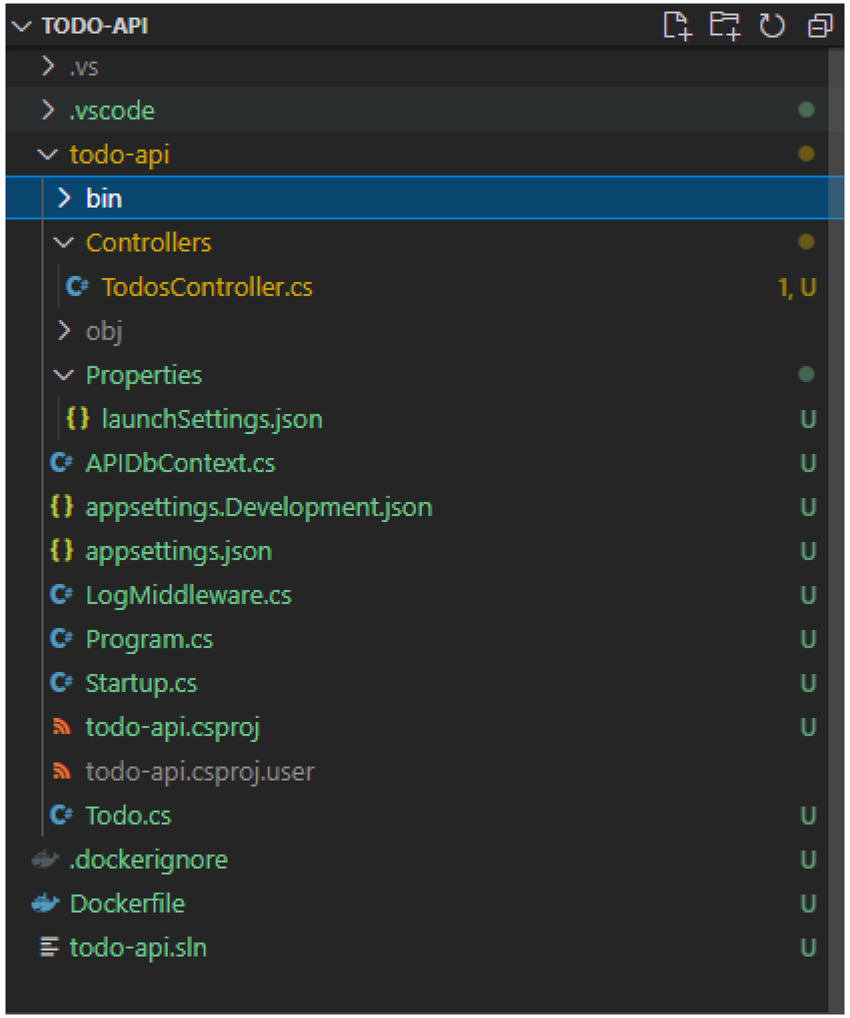
TODO Angular – Build image



```
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-angular> docker build -t todo-angular:v1.0.0 .
[+] Building 0.5s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                                                                        0.0s
 => => transferring dockerfile: 174B                                                                                                        0.0s
 => [internal] load .dockerignore                                                                                                           0.0s
 => => transferring context: 2B                                                                                                             0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                                                                             0.0s
 => [1/4] FROM docker.io/library/nginx                                                                                                      0.1s
 => [internal] load build context                                                                                                           0.0s
 => => transferring context: 346.52kB                                                                                                       0.0s
 => [2/4] COPY dist/todo-angular/ /usr/share/nginx/html/                                                                                    0.0s
 => [3/4] COPY nginx.conf /etc/nginx/nginx.conf                                                                                             0.0s
 => [4/4] RUN echo "complete building image"                                                                                                0.2s
 => exporting to image                                                                                                                      0.0s
 => => exporting layers                                                                                                                     0.0s
 => => writing image sha256:36ab4e38654d26a8e003eca046c4bbdc69805fded44f4e6fea2425de14067bca                                                0.0s
 => => naming to docker.io/library/todo-angular:v1.0.0                                                                                      0.0s
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-angular>
```

TODO API

- Dotnet new
- Appsettings.json
- EFCore, dbcontext
- Todo controller
- RabbitMQ client
- Log middleware
- Dotnet run
- Dotnet build

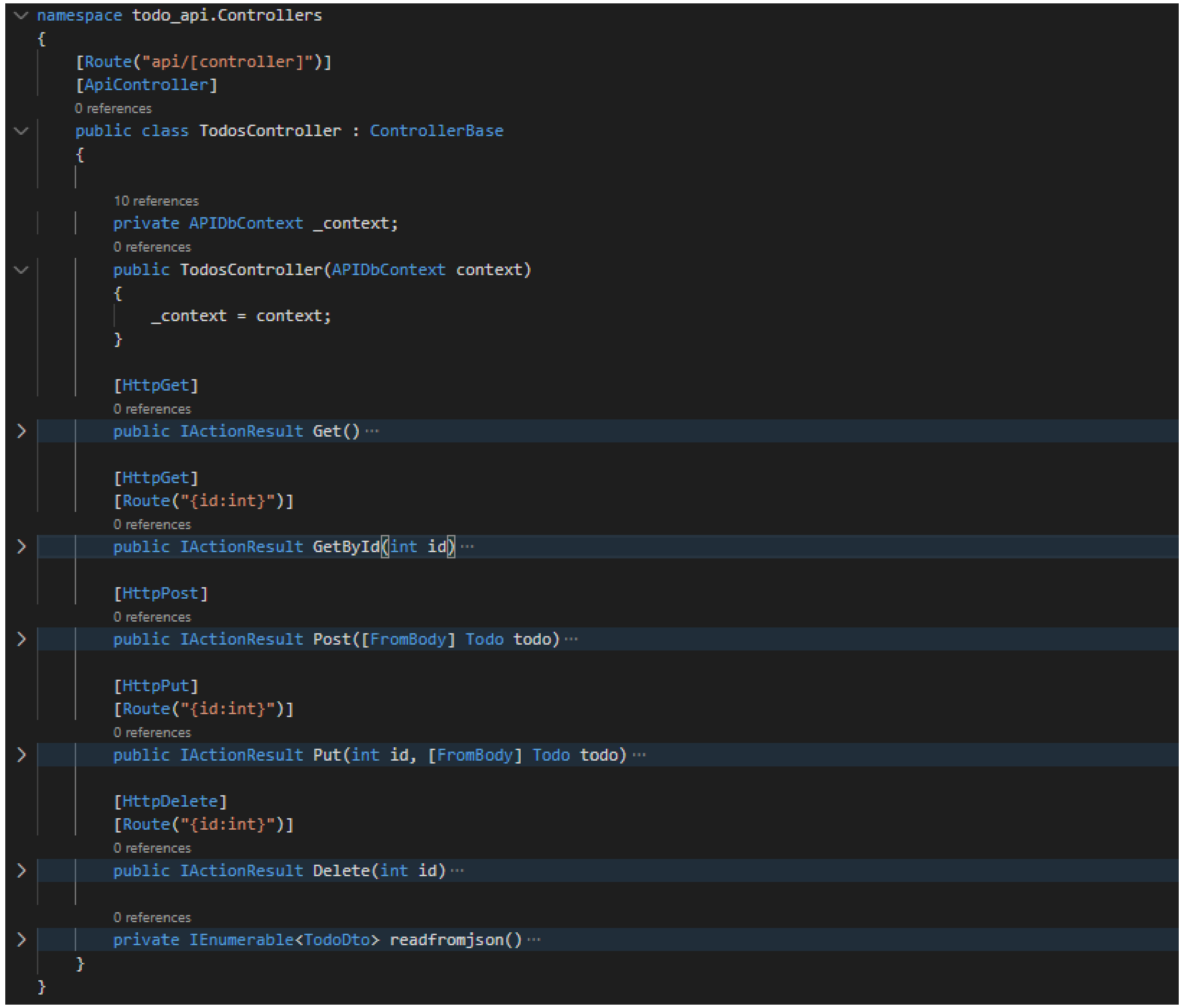


```
TODO-API
  > .vs
  > .vscode
  v todo-api
    > bin
    v Controllers
        TodosController.cs          1, U
    > obj
    v Properties
        launchSettings.json         U
      APIDbContext.cs               U
      appsettings.Development.json  U
      appsettings.json              U
      LogMiddleware.cs              U
      Program.cs                    U
      Startup.cs                    U
      todo-api.csproj               U
      todo-api.csproj.user
      Todo.cs                       U
    .dockerignore                   U
    Dockerfile                      U
    todo-api.sln                    U
```

TODO API – App settings.json

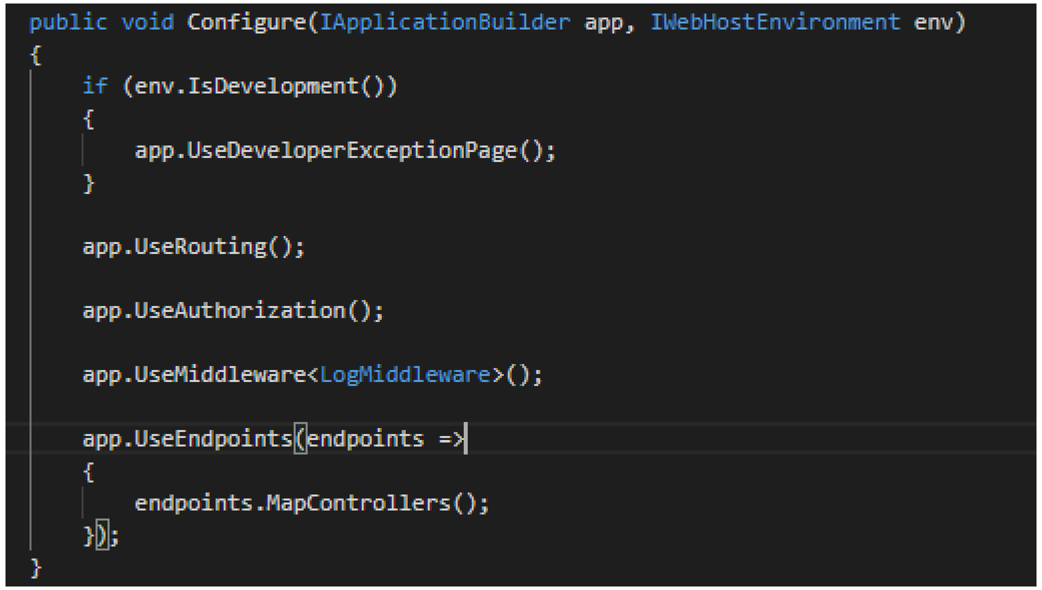
```json
{
    "Mysql": "server=localhost;database=todo_api_db;user=root;password=123456",
    "Rabbitmq": {
        "HostName": "localhost",
        "UserName": "admin",
        "Password": "123456"
    },
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft": "Warning",
            "Microsoft.Hosting.Lifetime": "Information"
        }
    },
    "AllowedHosts": "*"
}
```

TODO API – Todos controller

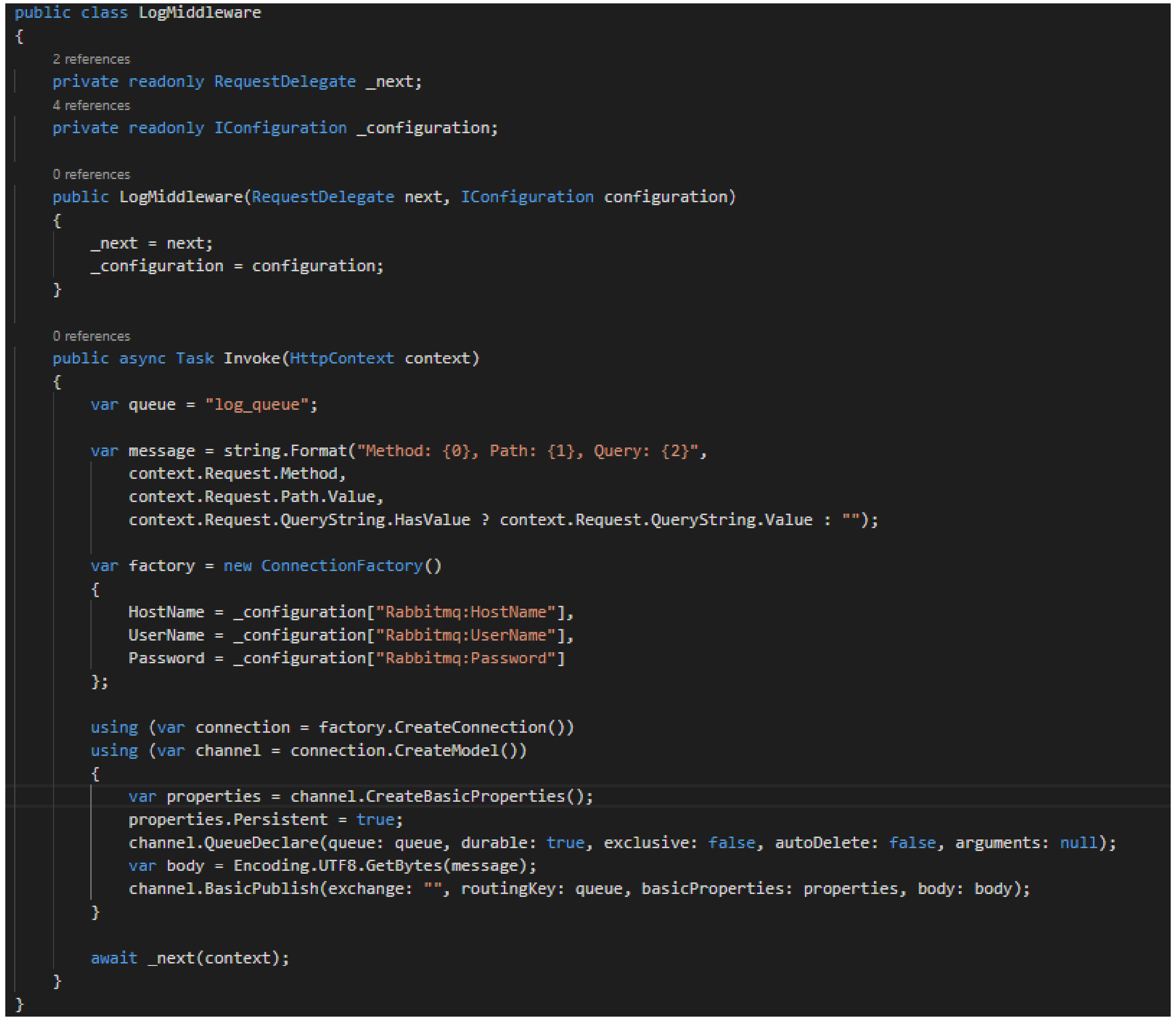


```
namespace todo_api.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    0 references
    public class TodosController : ControllerBase
    {

        10 references
        private APIDbContext _context;
        0 references
        public TodosController(APIDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        0 references
        public IActionResult Get() ···

        [HttpGet]
        [Route("{id:int}")]
        0 references
        public IActionResult GetById(int id) ···

        [HttpPost]
        0 references
        public IActionResult Post([FromBody] Todo todo) ···

        [HttpPut]
        [Route("{id:int}")]
        0 references
        public IActionResult Put(int id, [FromBody] Todo todo) ···

        [HttpDelete]
        [Route("{id:int}")]
        0 references
        public IActionResult Delete(int id) ···

        0 references
        private IEnumerable<TodoDto> readfromjson() ···
    }
}
```

TODO API – Log middleware and rabbit client

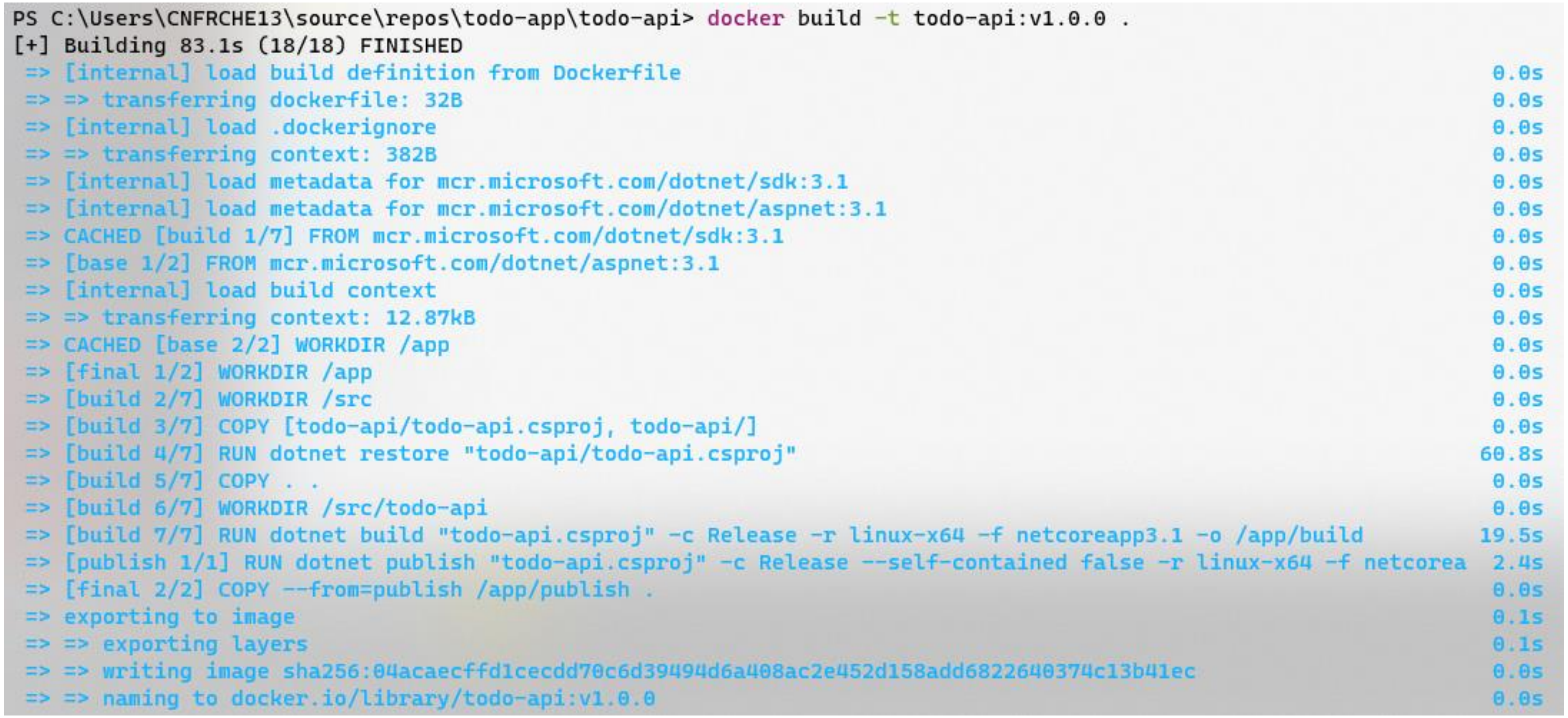


```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseRouting();

    app.UseAuthorization();

    app.UseMiddleware<LogMiddleware>();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```



```
public class LogMiddleware
{
    2 references
    private readonly RequestDelegate _next;
    4 references
    private readonly IConfiguration _configuration;

    0 references
    public LogMiddleware(RequestDelegate next, IConfiguration configuration)
    {
        _next = next;
        _configuration = configuration;
    }

    0 references
    public async Task Invoke(HttpContext context)
    {
        var queue = "log_queue";

        var message = string.Format("Method: {0}, Path: {1}, Query: {2}",
            context.Request.Method,
            context.Request.Path.Value,
            context.Request.QueryString.HasValue ? context.Request.QueryString.Value : "");

        var factory = new ConnectionFactory()
        {
            HostName = _configuration["Rabbitmq:HostName"],
            UserName = _configuration["Rabbitmq:UserName"],
            Password = _configuration["Rabbitmq:Password"]
        };

        using (var connection = factory.CreateConnection())
        using (var channel = connection.CreateModel())
        {
            var properties = channel.CreateBasicProperties();
            properties.Persistent = true;
            channel.QueueDeclare(queue: queue, durable: true, exclusive: false, autoDelete: false, arguments: null);
            var body = Encoding.UTF8.GetBytes(message);
            channel.BasicPublish(exchange: "", routingKey: queue, basicProperties: properties, body: body);
        }

        await _next(context);
    }
}
```

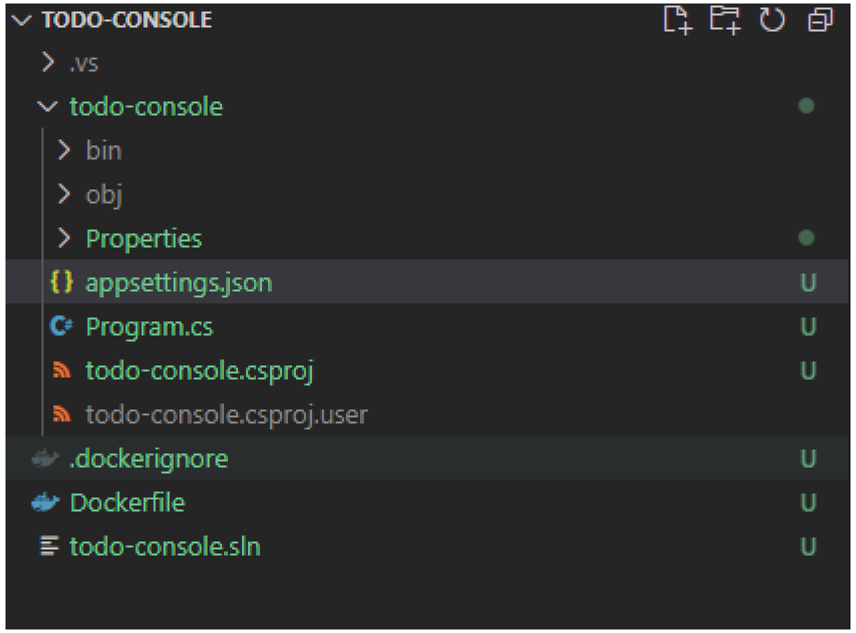
TODO API – Build image

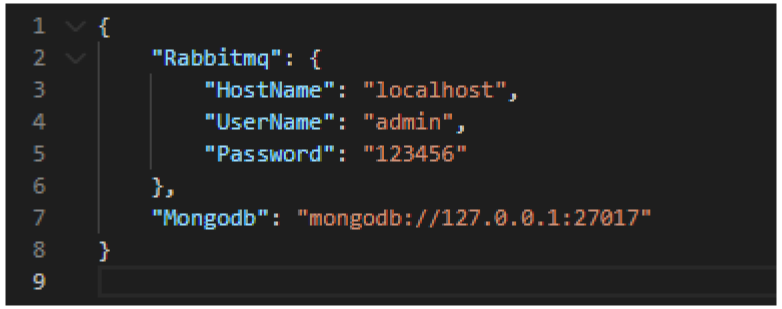


```
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-api> docker build -t todo-api:v1.0.0 .
[+] Building 83.1s (18/18) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 32B                                                                     0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 382B                                                                       0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:3.1                                       0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:3.1                                    0.0s
 => CACHED [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:3.1                                            0.0s
 => [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:3.1                                                 0.0s
 => [internal] load build context                                                                       0.0s
 => => transferring context: 12.87kB                                                                    0.0s
 => CACHED [base 2/2] WORKDIR /app                                                                      0.0s
 => [final 1/2] WORKDIR /app                                                                            0.0s
 => [build 2/7] WORKDIR /src                                                                            0.0s
 => [build 3/7] COPY [todo-api/todo-api.csproj, todo-api/]                                              0.0s
 => [build 4/7] RUN dotnet restore "todo-api/todo-api.csproj"                                          60.8s
 => [build 5/7] COPY . .                                                                                0.0s
 => [build 6/7] WORKDIR /src/todo-api                                                                   0.0s
 => [build 7/7] RUN dotnet build "todo-api.csproj" -c Release -r linux-x64 -f netcoreapp3.1 -o /app/build  19.5s
 => [publish 1/1] RUN dotnet publish "todo-api.csproj" -c Release --self-contained false -r linux-x64 -f netcorea  2.4s
 => [final 2/2] COPY --from=publish /app/publish .                                                      0.0s
 => exporting to image                                                                                  0.1s
 => => exporting layers                                                                                 0.1s
 => => writing image sha256:04acaecffd1cecdd70c6d39494d6a408ac2e452d158add6822640374c13b41ec             0.0s
 => => naming to docker.io/library/todo-api:v1.0.0                                                      0.0s
```

TODO Console

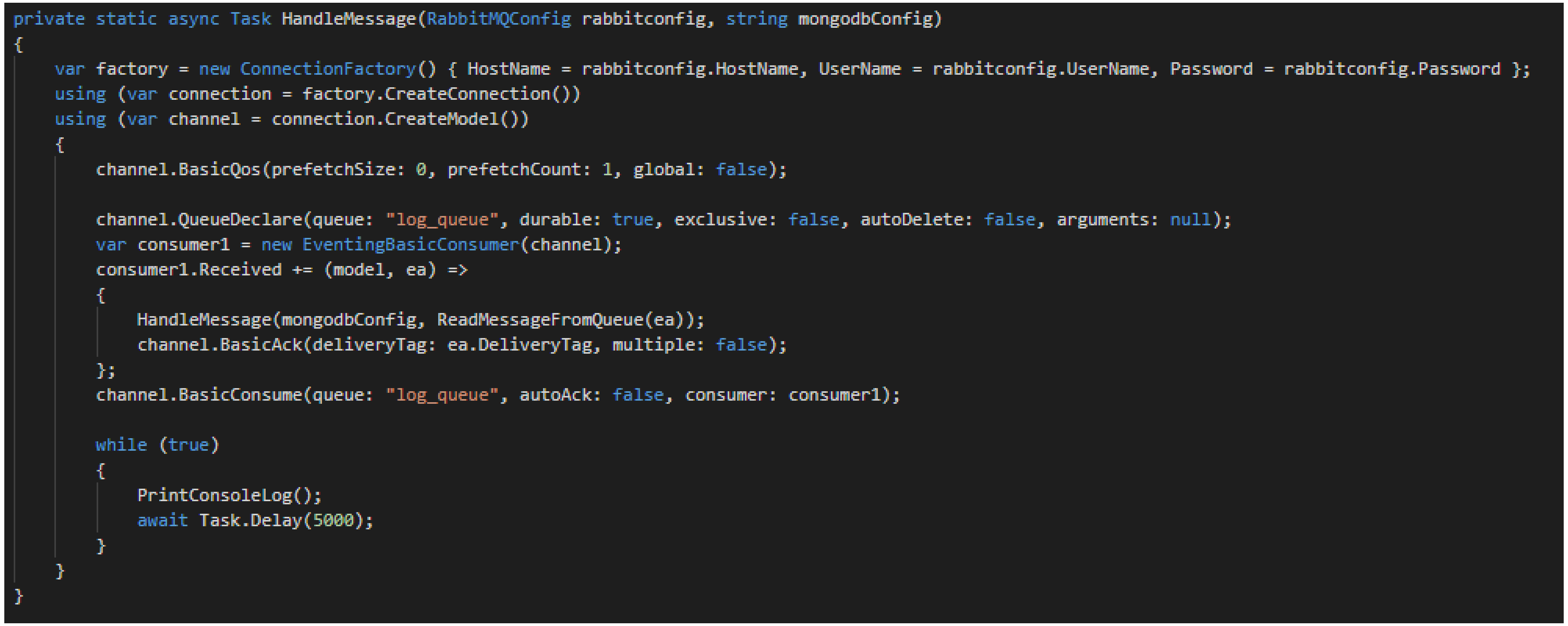
- app-settings.json
- Mongodb
- Rabbit client
- Dockerfile
- Build image





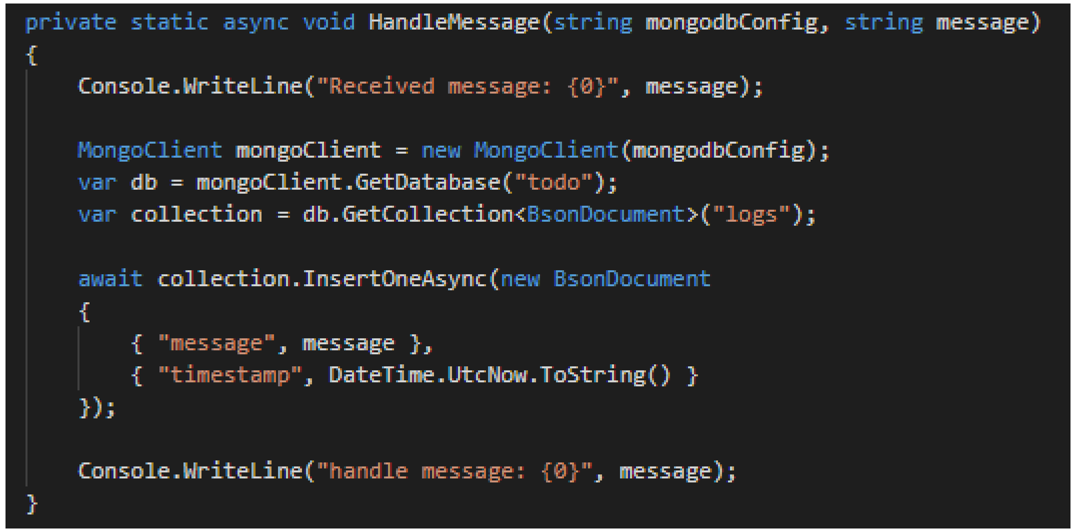
```json
{
    "Rabbitmq": {
        "HostName": "localhost",
        "UserName": "admin",
        "Password": "123456"
    },
    "Mongodb": "mongodb://127.0.0.1:27017"
}
```

TODO Console – Rabbit client



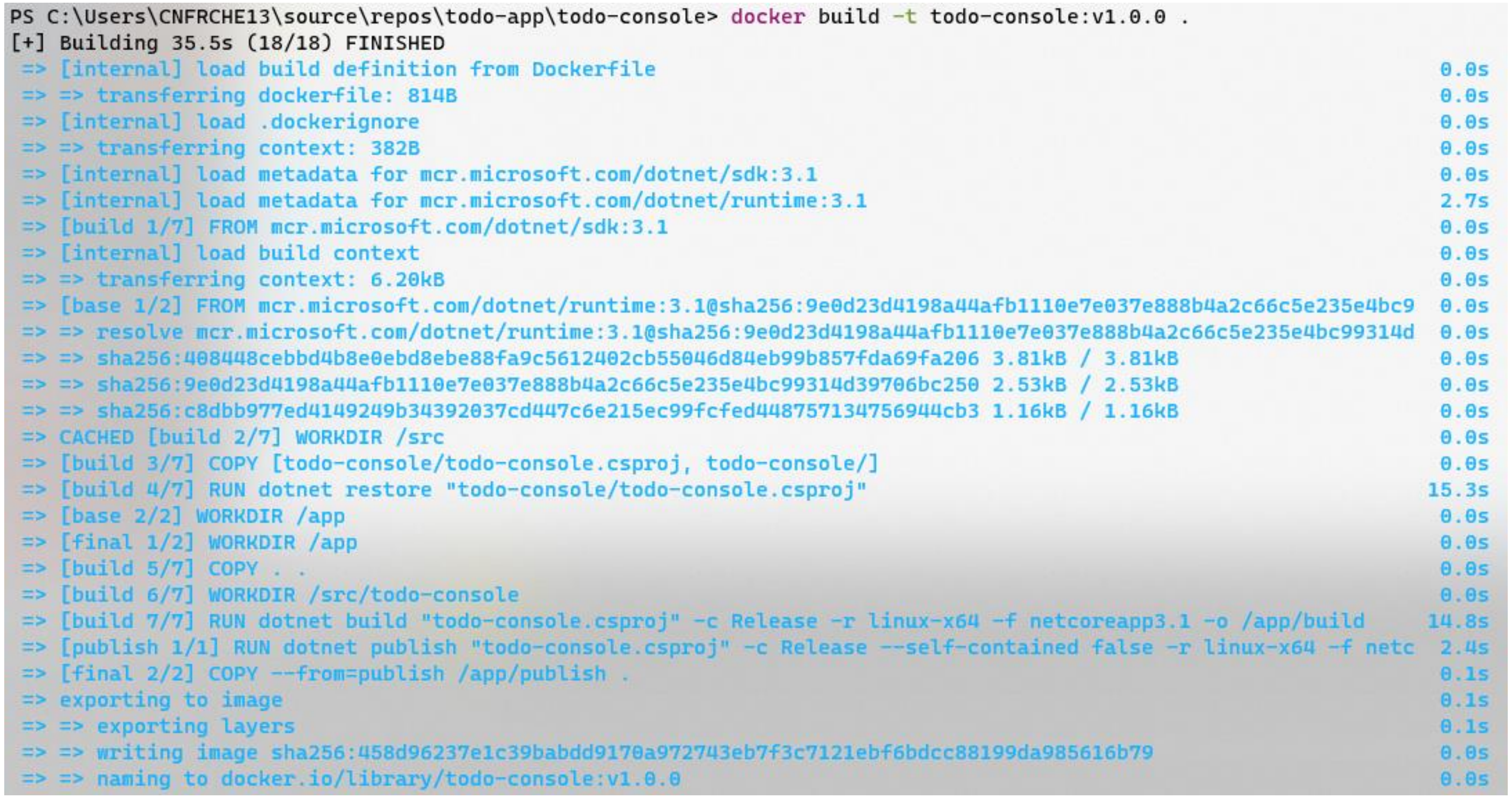
```csharp
private static async Task HandleMessage(RabbitMQConfig rabbitconfig, string mongodbConfig)
{
    var factory = new ConnectionFactory() { HostName = rabbitconfig.HostName, UserName = rabbitconfig.UserName, Password = rabbitconfig.Password };
    using (var connection = factory.CreateConnection())
    using (var channel = connection.CreateModel())
    {
        channel.BasicQos(prefetchSize: 0, prefetchCount: 1, global: false);

        channel.QueueDeclare(queue: "log_queue", durable: true, exclusive: false, autoDelete: false, arguments: null);
        var consumer1 = new EventingBasicConsumer(channel);
        consumer1.Received += (model, ea) =>
        {
            HandleMessage(mongodbConfig, ReadMessageFromQueue(ea));
            channel.BasicAck(deliveryTag: ea.DeliveryTag, multiple: false);
        };
        channel.BasicConsume(queue: "log_queue", autoAck: false, consumer: consumer1);

        while (true)
        {
            PrintConsoleLog();
            await Task.Delay(5000);
        }
    }
}
```

TODO Console – Mongodb



```
private static async void HandleMessage(string mongodbConfig, string message)
{
    Console.WriteLine("Received message: {0}", message);

    MongoClient mongoClient = new MongoClient(mongodbConfig);
    var db = mongoClient.GetDatabase("todo");
    var collection = db.GetCollection<BsonDocument>("logs");

    await collection.InsertOneAsync(new BsonDocument
    {
        { "message", message },
        { "timestamp", DateTime.UtcNow.ToString() }
    });

    Console.WriteLine("handle message: {0}", message);
}
```

TODO Console – Build image



```
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-console> docker build -t todo-console:v1.0.0 .
[+] Building 35.5s (18/18) FINISHED
 => [internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 814B                                                                          0.0s
 => [internal] load .dockerignore                                                                             0.0s
 => => transferring context: 382B                                                                             0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:3.1                                             0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/runtime:3.1                                         2.7s
 => [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:3.1                                                         0.0s
 => [internal] load build context                                                                             0.0s
 => => transferring context: 6.20kB                                                                           0.0s
 => [base 1/2] FROM mcr.microsoft.com/dotnet/runtime:3.1@sha256:9e0d23d4198a44afb1110e7e037e888b4a2c66c5e235e4bc9  0.0s
 => => resolve mcr.microsoft.com/dotnet/runtime:3.1@sha256:9e0d23d4198a44afb1110e7e037e888b4a2c66c5e235e4bc99314d  0.0s
 => => sha256:408448cebbd4b8e0ebd8ebe88fa9c5612402cb55046d84eb99b857fda69fa206 3.81kB / 3.81kB                0.0s
 => => sha256:9e0d23d4198a44afb1110e7e037e888b4a2c66c5e235e4bc99314d39706bc250 2.53kB / 2.53kB                0.0s
 => => sha256:c8dbb977ed4149249b34392037cd447c6e215ec99fcfed448757134756944cb3 1.16kB / 1.16kB                0.0s
 => CACHED [build 2/7] WORKDIR /src                                                                           0.0s
 => [build 3/7] COPY [todo-console/todo-console.csproj, todo-console/]                                        0.0s
 => [build 4/7] RUN dotnet restore "todo-console/todo-console.csproj"                                        15.3s
 => [base 2/2] WORKDIR /app                                                                                   0.0s
 => [final 1/2] WORKDIR /app                                                                                  0.0s
 => [build 5/7] COPY . .                                                                                      0.0s
 => [build 6/7] WORKDIR /src/todo-console                                                                     0.0s
 => [build 7/7] RUN dotnet build "todo-console.csproj" -c Release -r linux-x64 -f netcoreapp3.1 -o /app/build    14.8s
 => [publish 1/1] RUN dotnet publish "todo-console.csproj" -c Release --self-contained false -r linux-x64 -f netc  2.4s
 => [final 2/2] COPY --from=publish /app/publish .                                                            0.1s
 => exporting to image                                                                                        0.1s
 => => exporting layers                                                                                       0.1s
 => => writing image sha256:458d96237e1c39babdd9170a972743eb7f3c7121ebf6bdcc88199da985616b79                  0.0s
 => => naming to docker.io/library/todo-console:v1.0.0                                                        0.0s
```

Deploy

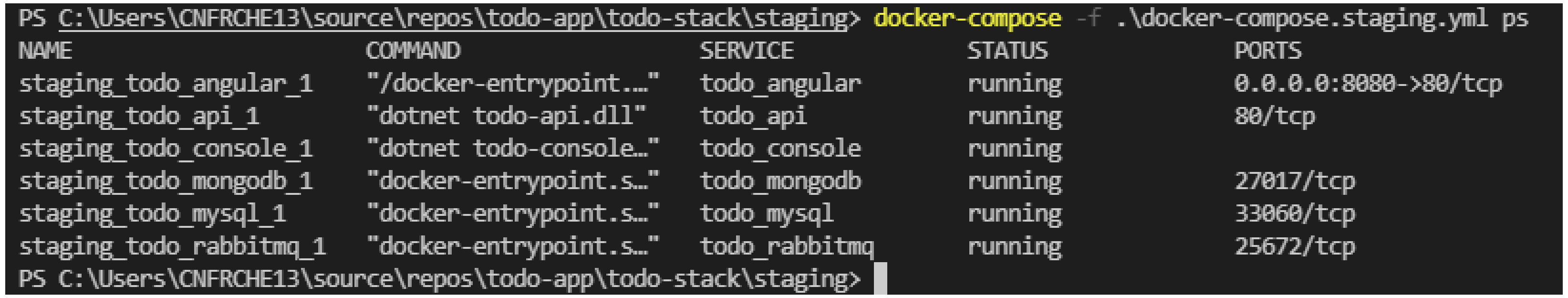
```yaml
version: '3.3'
services:

    todo_mysql:
        image: mysql
        volumes:
            - ../db/:/docker-entrypoint-initdb.d/
        environment:
            - "MYSQL_ROOT_PASSWORD=123456"
        expose:
            - "3306"

    todo_mongodb:
        image:  mongo
        expose:
            - "27017"

    todo_rabbitmq:
        hostname: todo_rabbitmq_host
        image: rabbitmq:management
        environment:
            - RABBITMQ_DEFAULT_USER=admin
            - RABBITMQ_DEFAULT_PASS=123456
        expose:
            - "5672"
            - "15672"

networks:
  default:
      external: true
      name: todo-app-staging
```

```yaml
version: '3.3'
services:

    todo_api:
        image: todo-api:v1.0.0
        expose:
            - "80"
        volumes:
            - ./todo-api.appsettings.json:/app/appsettings.json
        depends_on:
          - todo_mysql
          - todo_rabbitmq

    todo_console:
        image: todo-console:v1.0.0
        depends_on:
          - todo_api
        volumes:
            - ./todo-console.appsettings.json:/app/appsettings.json

    todo_angular:
        image: todo-angular:v1.0.0
        ports:
            - "8080:80"
        depends_on:
            - todo_console
            - todo_api_db
        volumes:
            - ./nginx.conf:/etc/nginx/nginx.conf

networks:
  default:
      external: true
      name: todo-app-staging
```

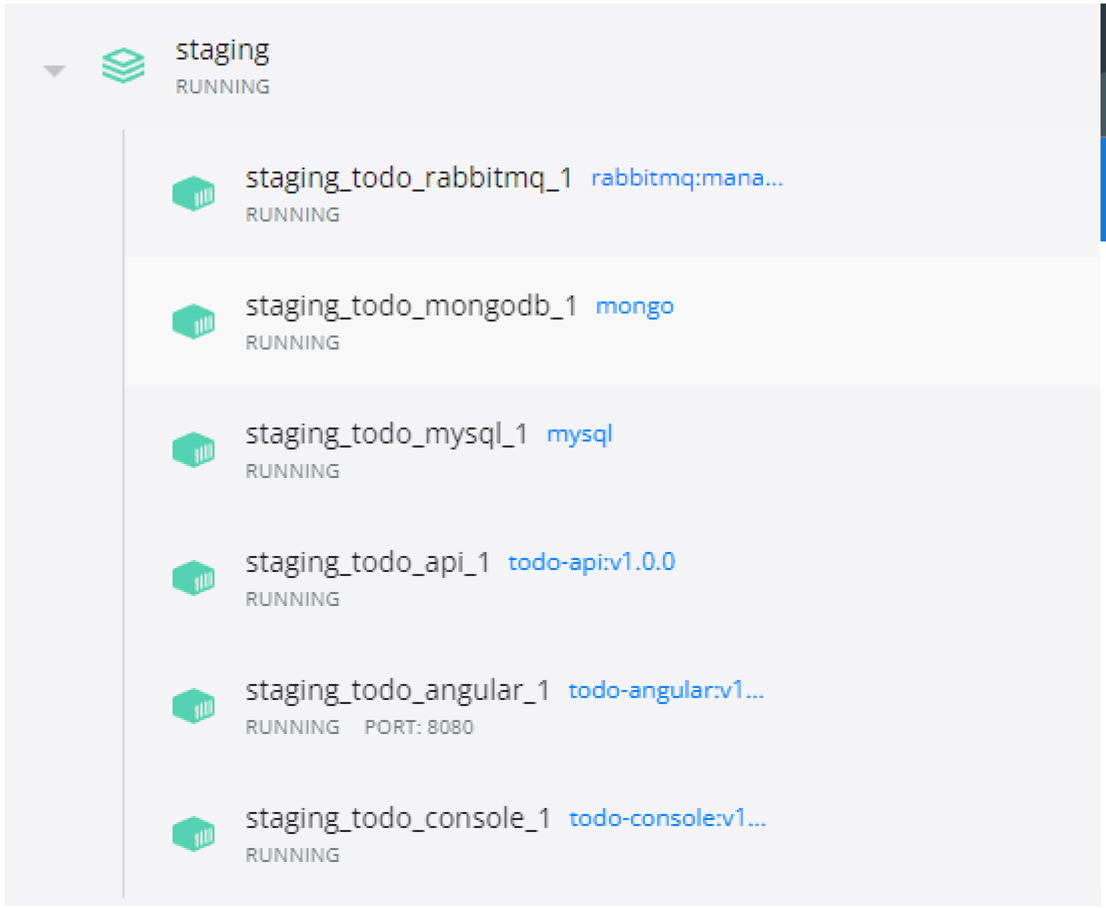
Deploy in staging

docker-compose -f .\docker-compose.infra.yml up -d

docker-compose -f .\docker-compose.staging.yml up -d



```
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-stack\staging> docker-compose -f .\docker-compose.staging.yml ps
NAME                     COMMAND                   SERVICE          STATUS       PORTS
staging_todo_angular_1   "/docker-entrypoint.…"    todo_angular     running      0.0.0.0:8080->80/tcp
staging_todo_api_1       "dotnet todo-api.dll"     todo_api         running      80/tcp
staging_todo_console_1   "dotnet todo-console…"    todo_console     running
staging_todo_mongodb_1   "docker-entrypoint.s…"    todo_mongodb     running      27017/tcp
staging_todo_mysql_1     "docker-entrypoint.s…"    todo_mysql       running      33060/tcp
staging_todo_rabbitmq_1  "docker-entrypoint.s…"    todo_rabbitmq    running      25672/tcp
PS C:\Users\CNFRCHE13\source\repos\todo-app\todo-stack\staging>
```



staging
RUNNING

staging_todo_rabbitmq_1 rabbitmq:mana...
RUNNING

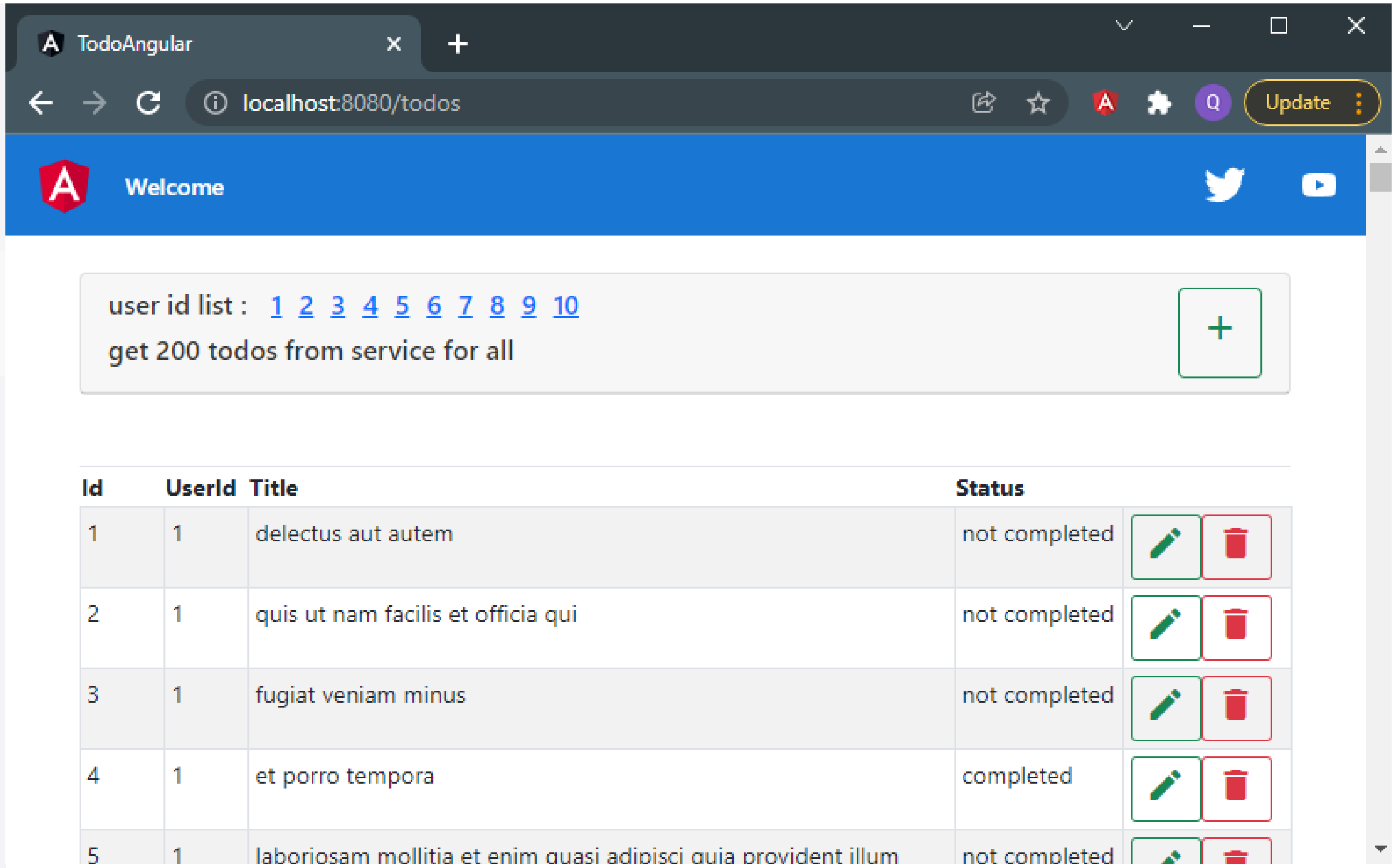
staging_todo_mongodb_1 mongo
RUNNING

staging_todo_mysql_1 mysql
RUNNING

staging_todo_api_1 todo-api:v1.0.0
RUNNING

staging_todo_angular_1 todo-angular:v1...
RUNNING PORT: 8080

staging_todo_console_1 todo-console:v1...
RUNNING

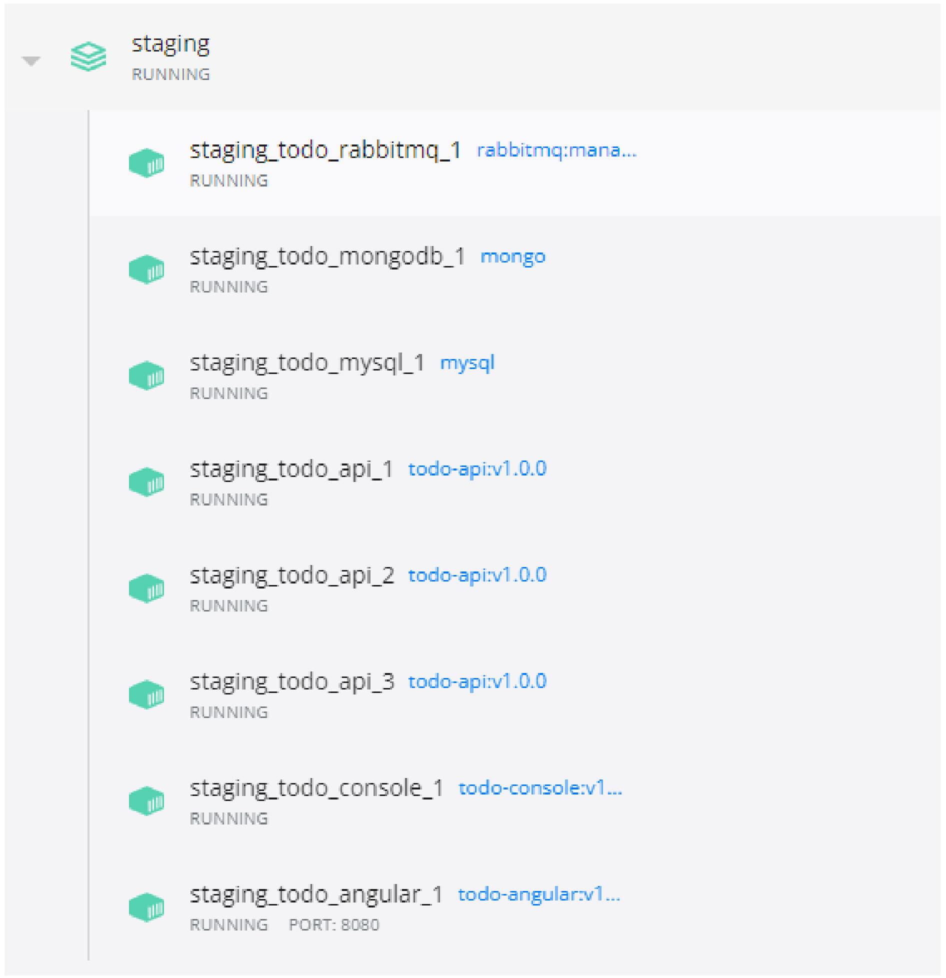


user id list : 1 2 3 4 5 6 7 8 9 10
get 200 todos from service for all

Id	UserId	Title	Status
1	1	delectus aut autem	not completed
2	1	quis ut nam facilis et officia qui	not completed
3	1	fugiat veniam minus	not completed
4	1	et porro tempora	completed
5	1	laboriosam mollitia et enim quasi adipisci quia provident illum	not completed

Deploy - Scale up/down

```
docker-compose -f .\docker-compose.staging.yml up -d --scale todo_api=3
docker-compose -f .\docker-compose.staging.yml up -d --scale todo_api=1
```



staging
RUNNING

staging_todo_rabbitmq_1 rabbitmq:mana...
RUNNING

staging_todo_mongodb_1 mongo
RUNNING

staging_todo_mysql_1 mysql
RUNNING

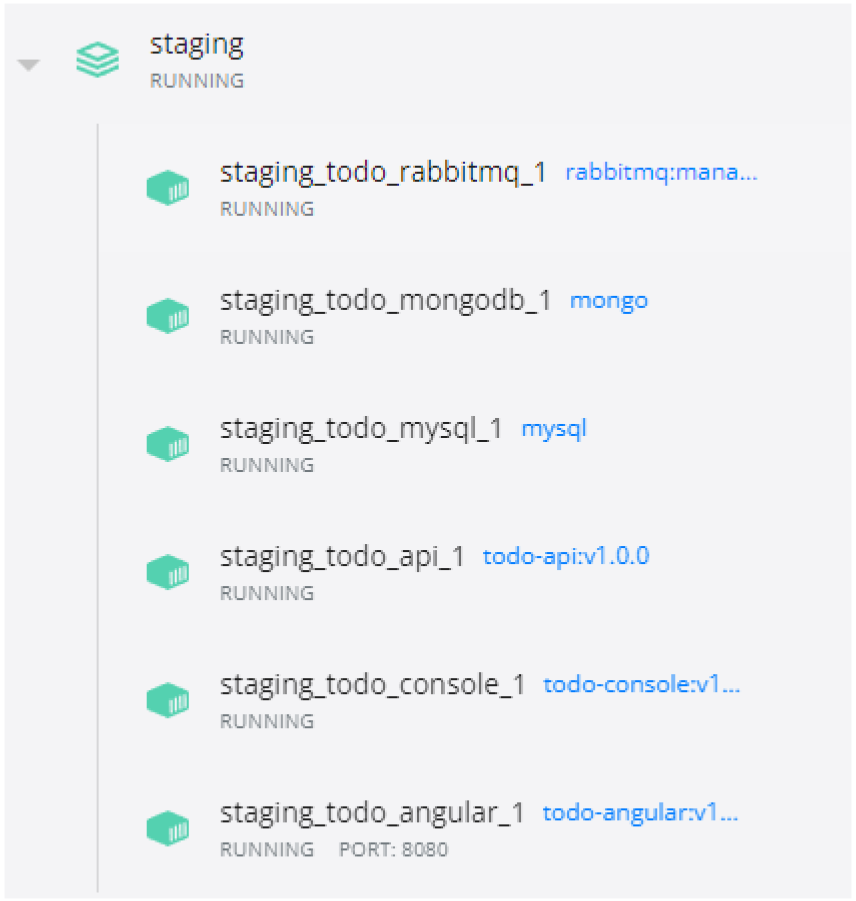
staging_todo_api_1 todo-api:v1.0.0
RUNNING

staging_todo_api_2 todo-api:v1.0.0
RUNNING

staging_todo_api_3 todo-api:v1.0.0
RUNNING

staging_todo_console_1 todo-console:v1...
RUNNING

staging_todo_angular_1 todo-angular:v1...
RUNNING PORT: 8080



staging
RUNNING

staging_todo_rabbitmq_1 rabbitmq:mana...
RUNNING

staging_todo_mongodb_1 mongo
RUNNING

staging_todo_mysql_1 mysql
RUNNING

staging_todo_api_1 todo-api:v1.0.0
RUNNING

staging_todo_console_1 todo-console:v1...
RUNNING

staging_todo_angular_1 todo-angular:v1...
RUNNING PORT: 8080

Q&A