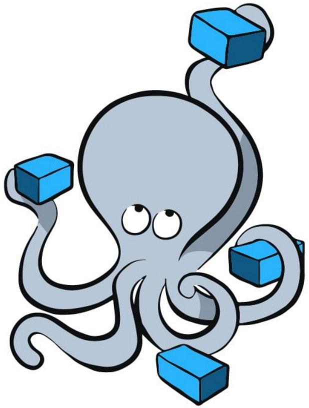


docker Compose

IN **10** MINUTES

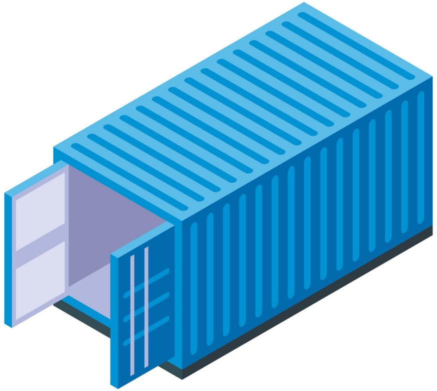
https://github.com/motero2k/docker-compose-tutorial

docker
Compose

Docker compose overview

Docker Compose is a **tool for defining and running multi-container applications**. It is the key to unlocking a streamlined and efficient development and deployment experience.

https://docs.docker.com/compose/

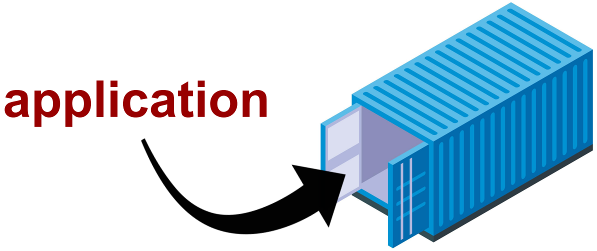


CONTAINERS?

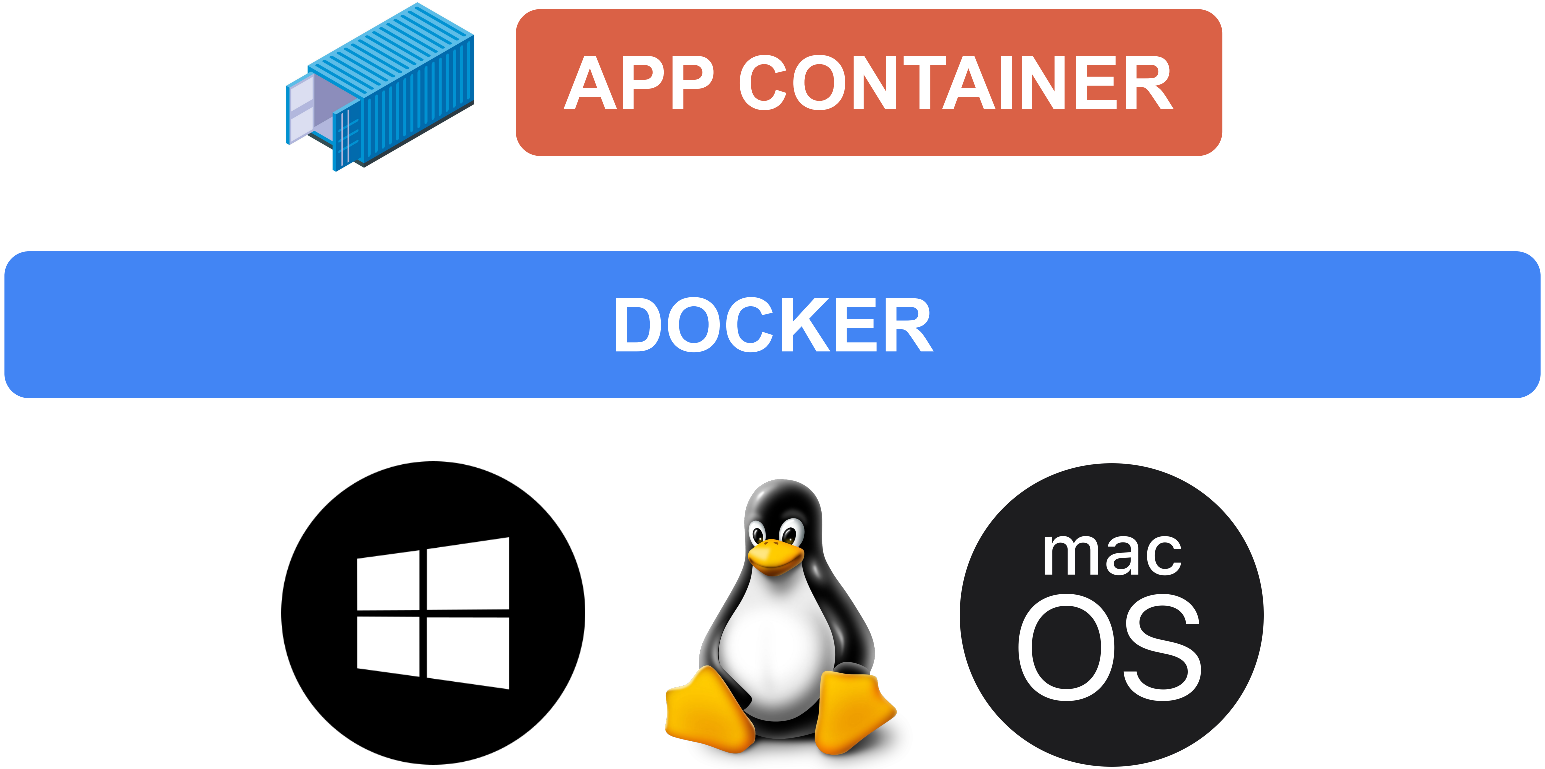
Docker overview

Docker is an open **platform** for developing, shipping, and running **applications**.

Docker enables you to **separate your applications from your infrastructure** so you can deliver software quickly.



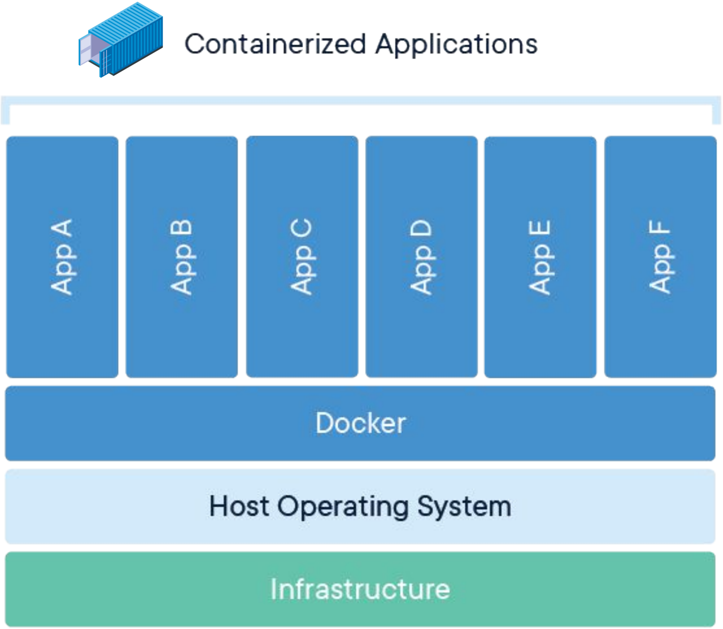
https://docs.docker.com/get-started/overview/

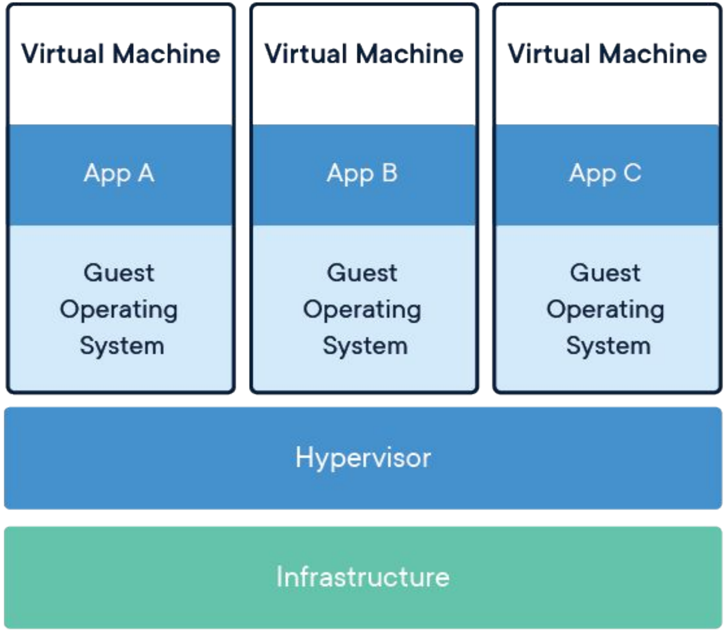


APP CONTAINER

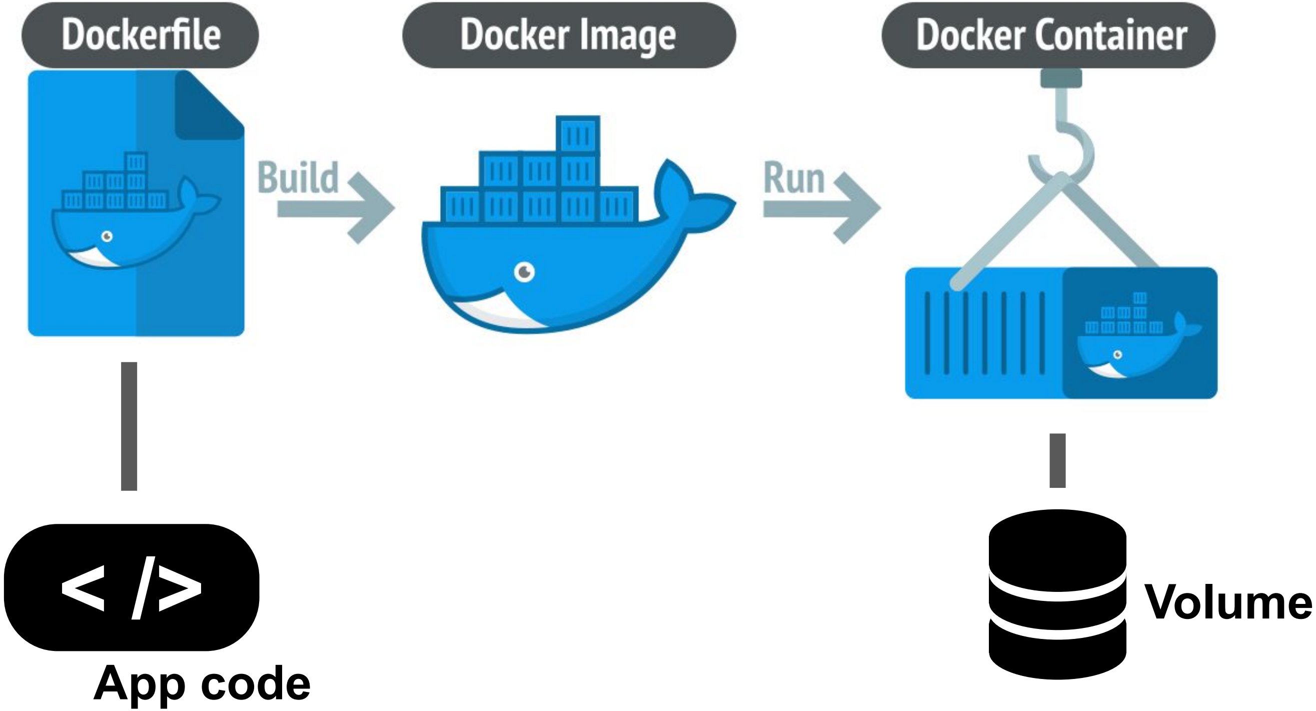
DOCKER

https://www.docker.com/resources/what-container/





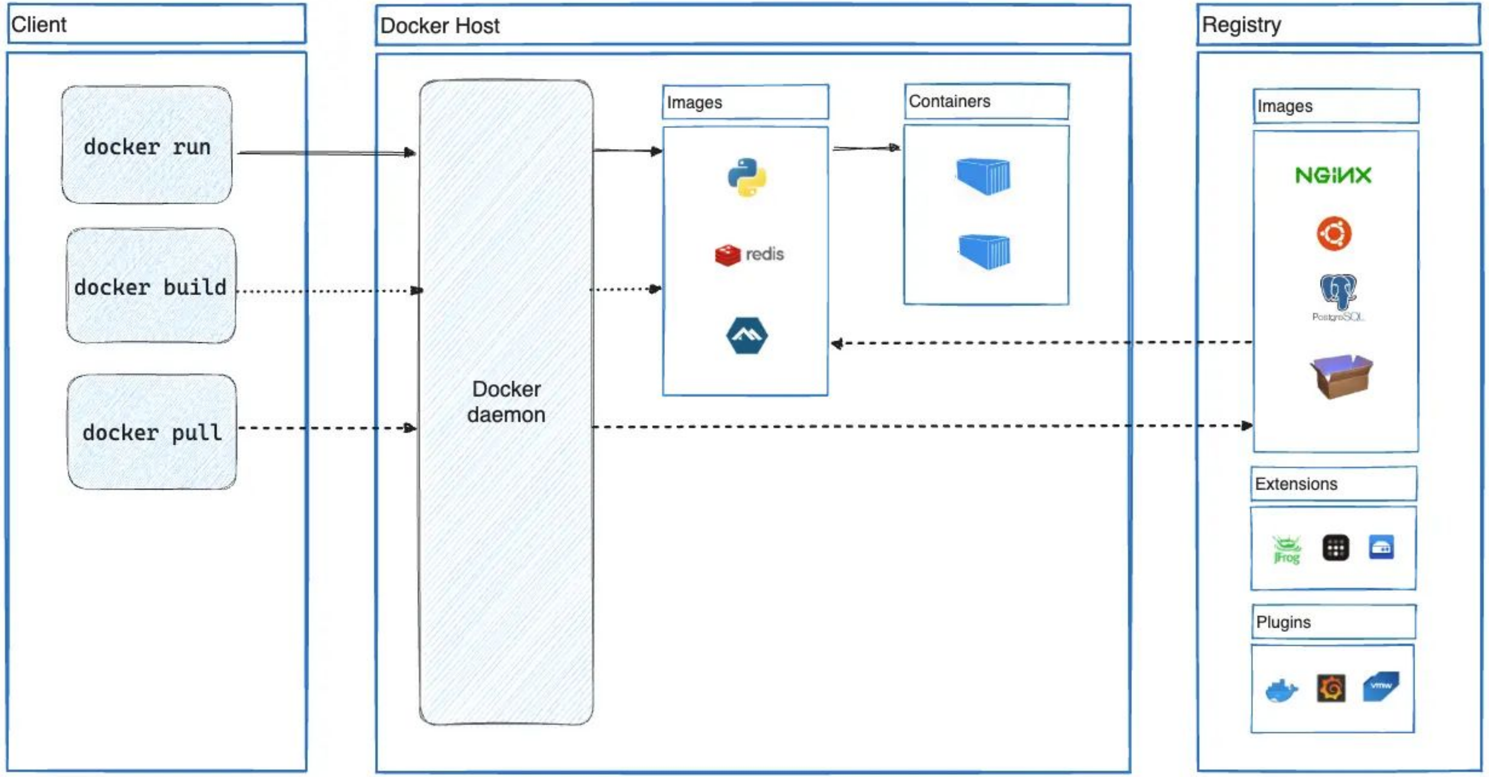
https://www.docker.com/resources/what-container/



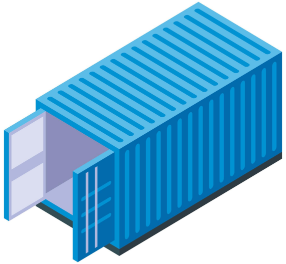
App code

Volume

https://linuxiac.com/what-is-docker-container/



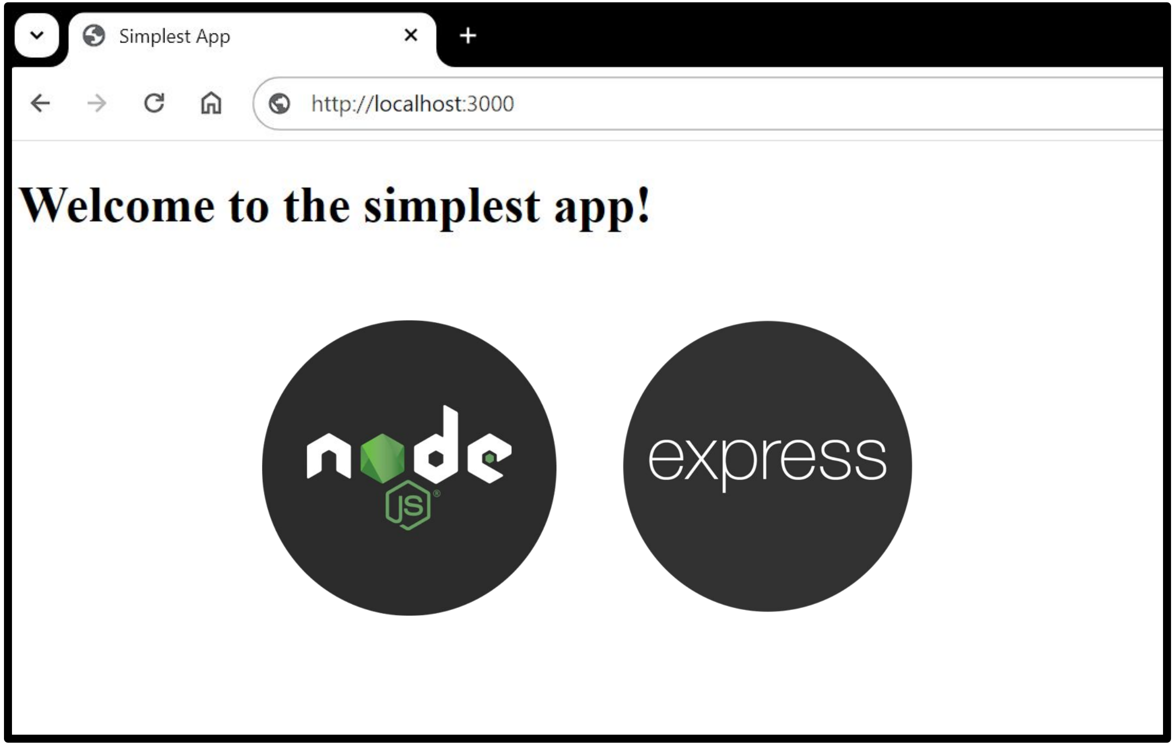
https://docs.docker.com/get-started/overview/#docker-architecture



CONTAINERIZE AN APPLICATION

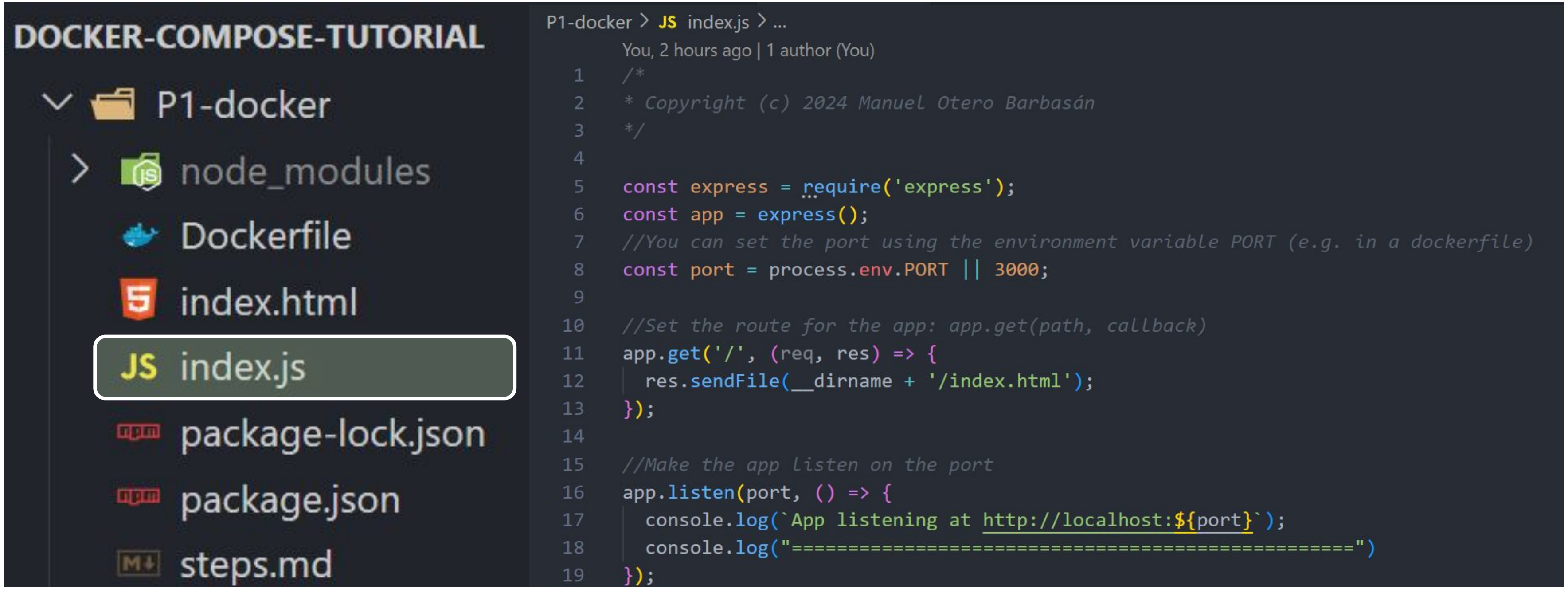
1. Code the app
2. Create Dockerfile
3. Build app image
4. Run the image

1. OUR SIMPLE APP



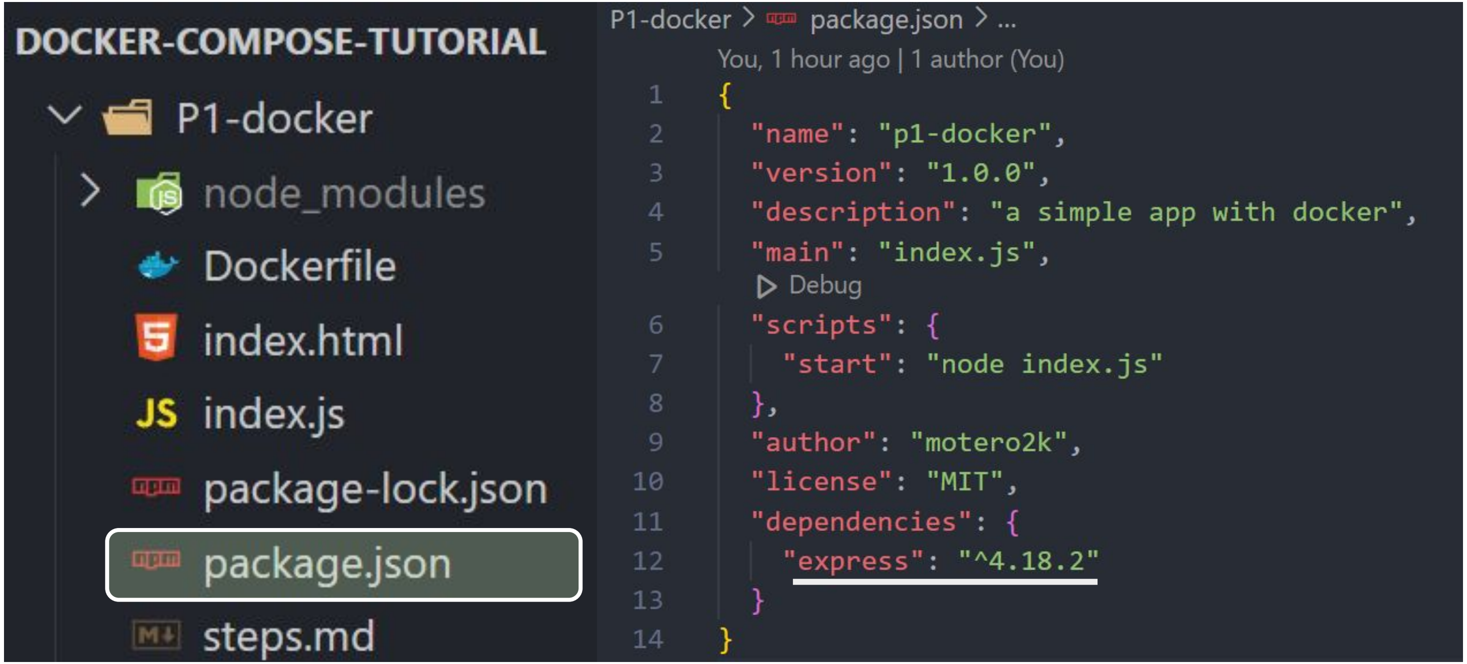
Welcome to the simplest app!

1. OUR SIMPLE APP



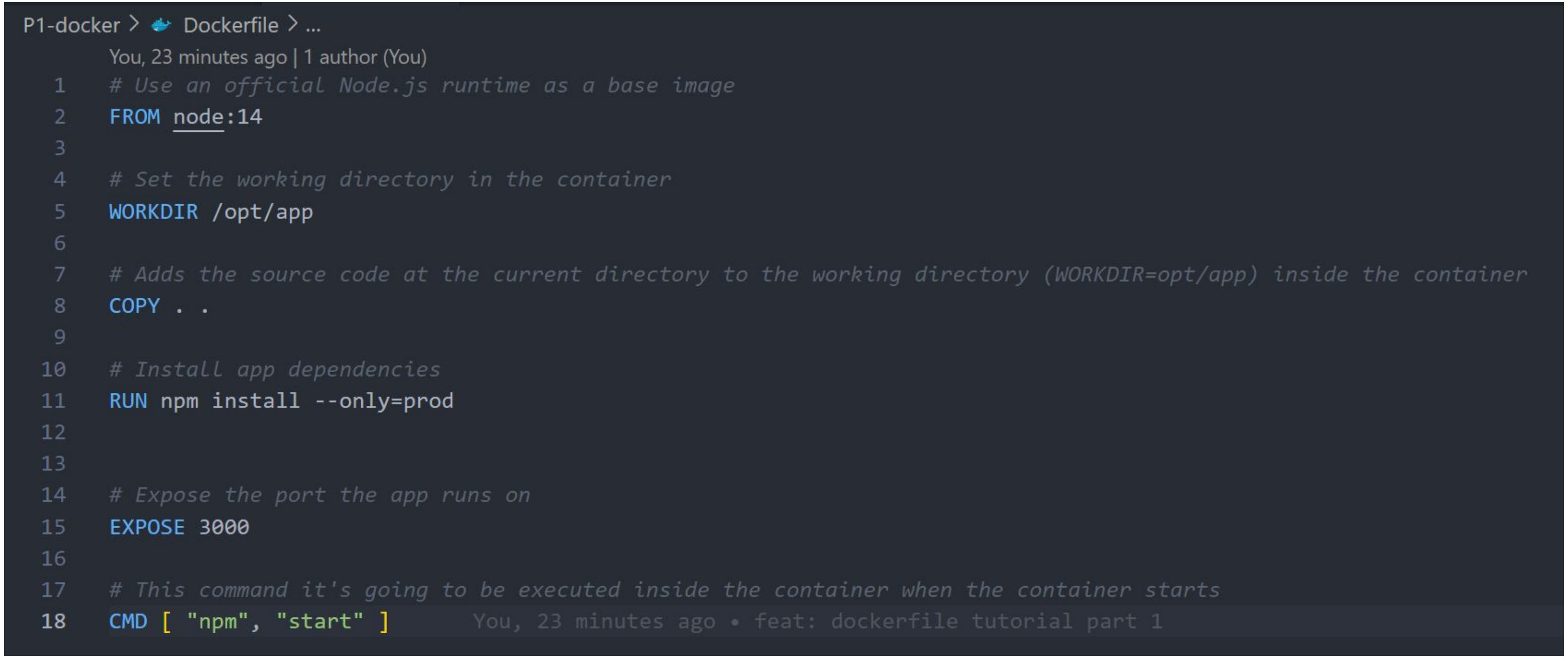
```javascript
/*
* Copyright (c) 2024 Manuel Otero Barbasán
*/

const express = require('express');
const app = express();
//You can set the port using the environment variable PORT (e.g. in a dockerfile)
const port = process.env.PORT || 3000;

//Set the route for the app: app.get(path, callback)
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/index.html');
});

//Make the app listen on the port
app.listen(port, () => {
  console.log(`App listening at http://localhost:${port}`);
  console.log("==================================================")
});
```

1. OUR SIMPLE APP



```json
{
  "name": "p1-docker",
  "version": "1.0.0",
  "description": "a simple app with docker",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "author": "motero2k",
  "license": "MIT",
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

2. DOCKERFILE

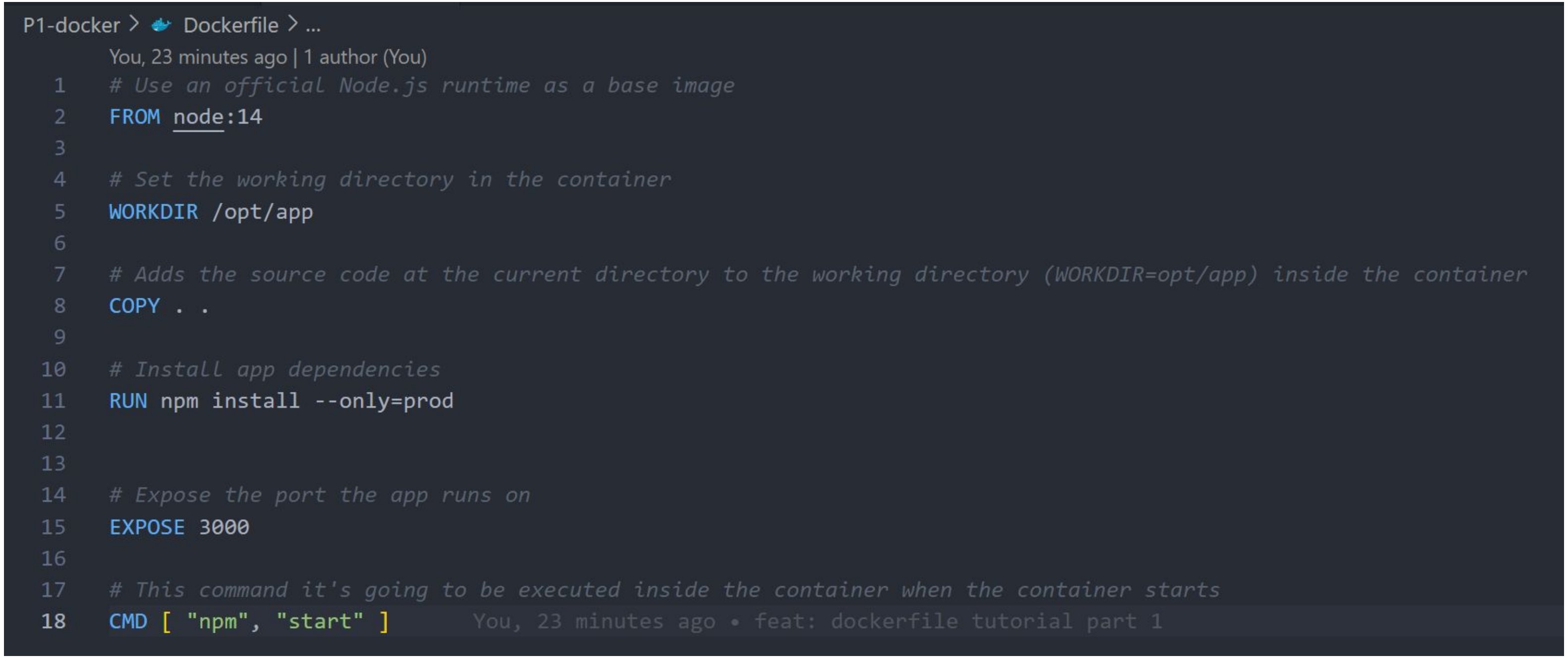


You, 23 minutes ago | 1 author (You)

```dockerfile
# Use an official Node.js runtime as a base image
FROM node:14

# Set the working directory in the container
WORKDIR /opt/app

# Adds the source code at the current directory to the working directory (WORKDIR=opt/app) inside the container
COPY . .

# Install app dependencies
RUN npm install --only=prod


# Expose the port the app runs on
EXPOSE 3000

# This command it's going to be executed inside the container when the container starts
CMD [ "npm", "start" ]      You, 23 minutes ago • feat: dockerfile tutorial part 1
```

https://docs.docker.com/engine/reference/builder/

2. DOCKERFILE



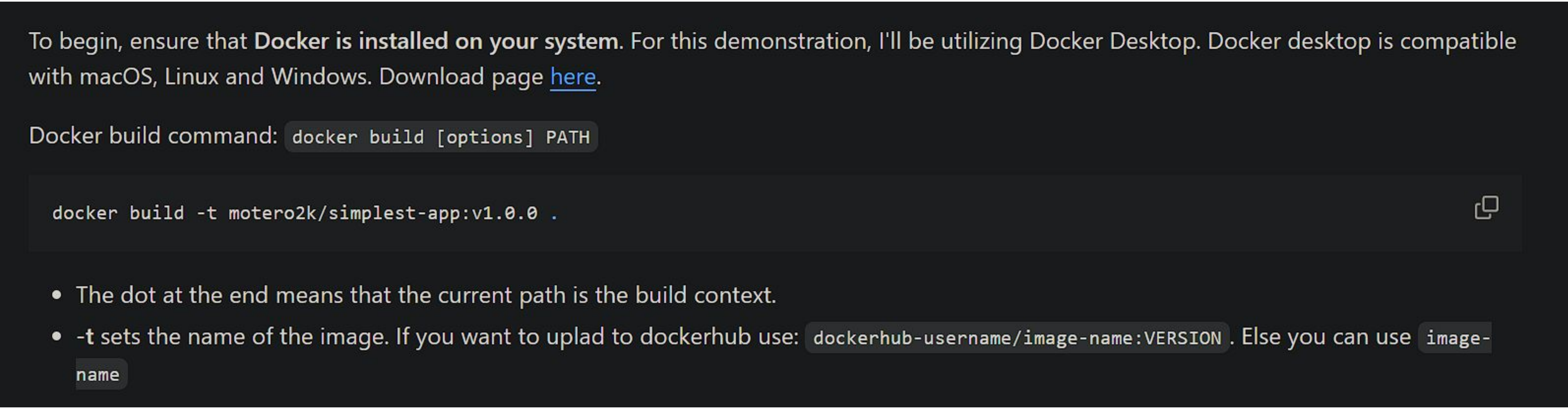
P1-docker > Dockerfile > ...

You, 23 minutes ago | 1 author (You)

```
# Use an official Node.js runtime as a base image
FROM node:14

# Set the working directory in the container
WORKDIR /opt/app

# Adds the source code at the current directory to the working directory (WORKDIR=opt/app) inside the container
COPY . .

# Install app dependencies
RUN npm install --only=prod


# Expose the port the app runs on
EXPOSE 3000

# This command it's going to be executed inside the container when the container starts
CMD [ "npm", "start" ]      You, 23 minutes ago • feat: dockerfile tutorial part 1
```

https://docs.docker.com/engine/reference/builder/

3. BUILD APP IMAGE



To begin, ensure that **Docker is installed on your system**. For this demonstration, I'll be utilizing Docker Desktop. Docker desktop is compatible with macOS, Linux and Windows. Download page here.

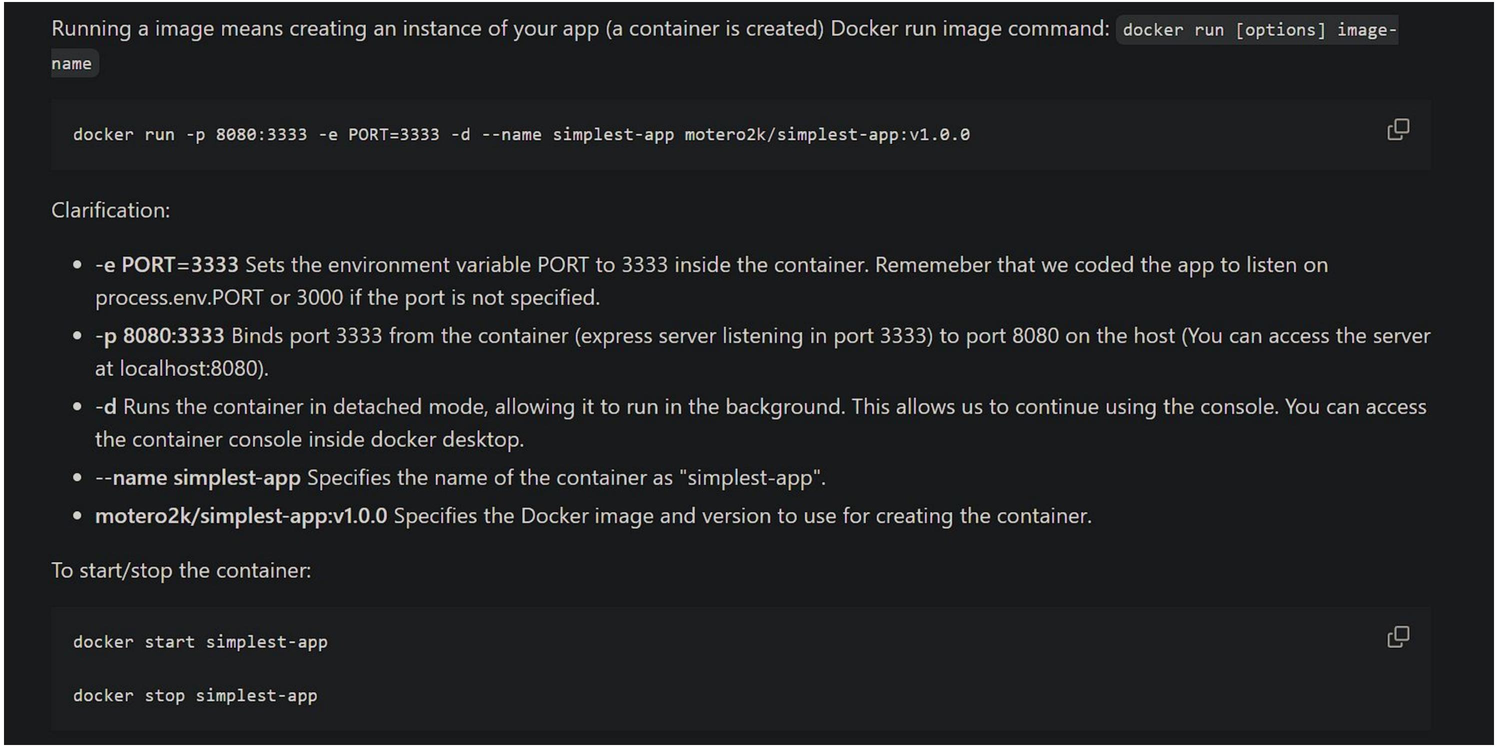
Docker build command: `docker build [options] PATH`

```
docker build -t motero2k/simplest-app:v1.0.0 .
```

- The dot at the end means that the current path is the build context.
- **-t** sets the name of the image. If you want to uplad to dockerhub use: `dockerhub-username/image-name:VERSION`. Else you can use `image-name`

https://github.com/motero2k/docker-compose-tutorial/blob/develop/P1-docker/steps.md

3. RUN IMAGE



Running a image means creating an instance of your app (a container is created) Docker run image command: `docker run [options] image-name`

```
docker run -p 8080:3333 -e PORT=3333 -d --name simplest-app motero2k/simplest-app:v1.0.0
```

Clarification:

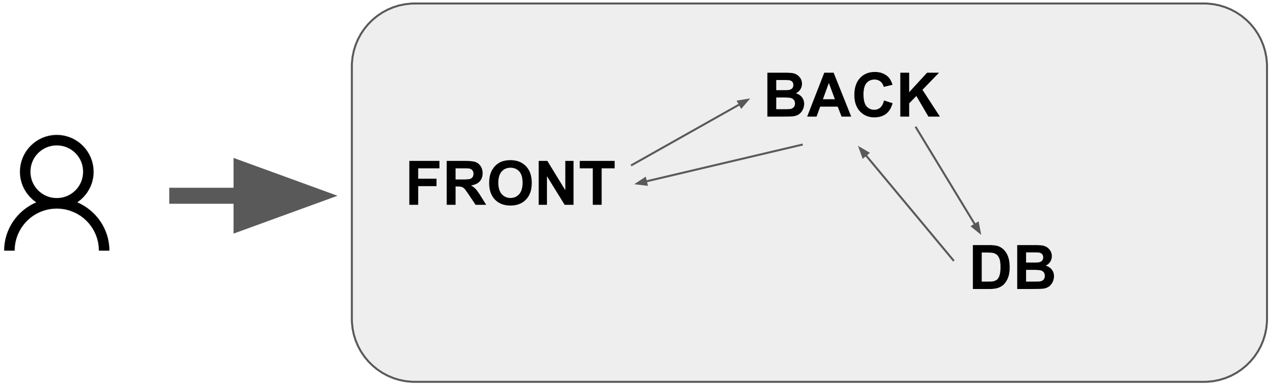
- **-e PORT=3333** Sets the environment variable PORT to 3333 inside the container. Rememeber that we coded the app to listen on process.env.PORT or 3000 if the port is not specified.
- **-p 8080:3333** Binds port 3333 from the container (express server listening in port 3333) to port 8080 on the host (You can access the server at localhost:8080).
- **-d** Runs the container in detached mode, allowing it to run in the background. This allows us to continue using the console. You can access the container console inside docker desktop.
- --**name simplest-app** Specifies the name of the container as "simplest-app".
- **motero2k/simplest-app:v1.0.0** Specifies the Docker image and version to use for creating the container.

To start/stop the container:

```
docker start simplest-app

docker stop simplest-app
```

https://github.com/motero2k/docker-compose-tutorial/blob/develop/P1-docker/steps.md

docker Compose

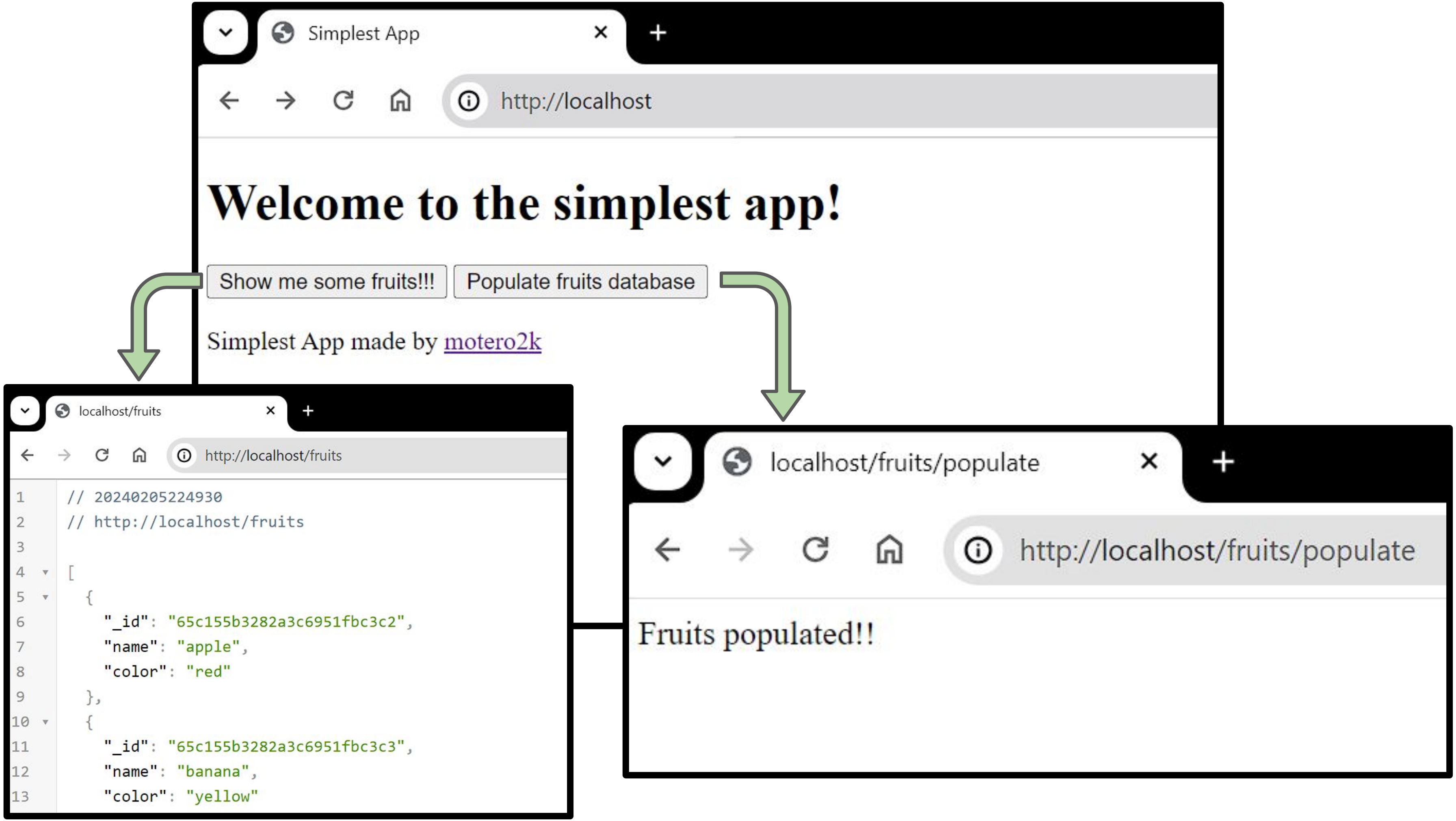


BACK

FRONT

DB

https://docs.docker.com/compose/gettingstarted/



Welcome to the simplest app!

Simplest App made by motero2k

```
// 20240205224930
// http://localhost/fruits

[
  {
    "_id": "65c155b3282a3c6951fbc3c2",
    "name": "apple",
    "color": "red"
  },
  {
    "_id": "65c155b3282a3c6951fbc3c3",
    "name": "banana",
    "color": "yellow"
```

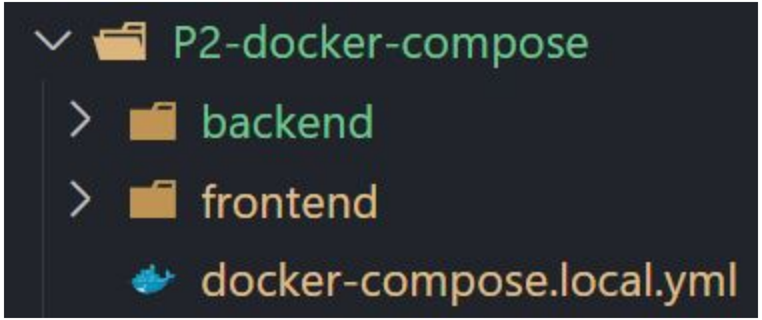
Fruits populated!!

docker
Compose

COMPOSE MICROSERVICES

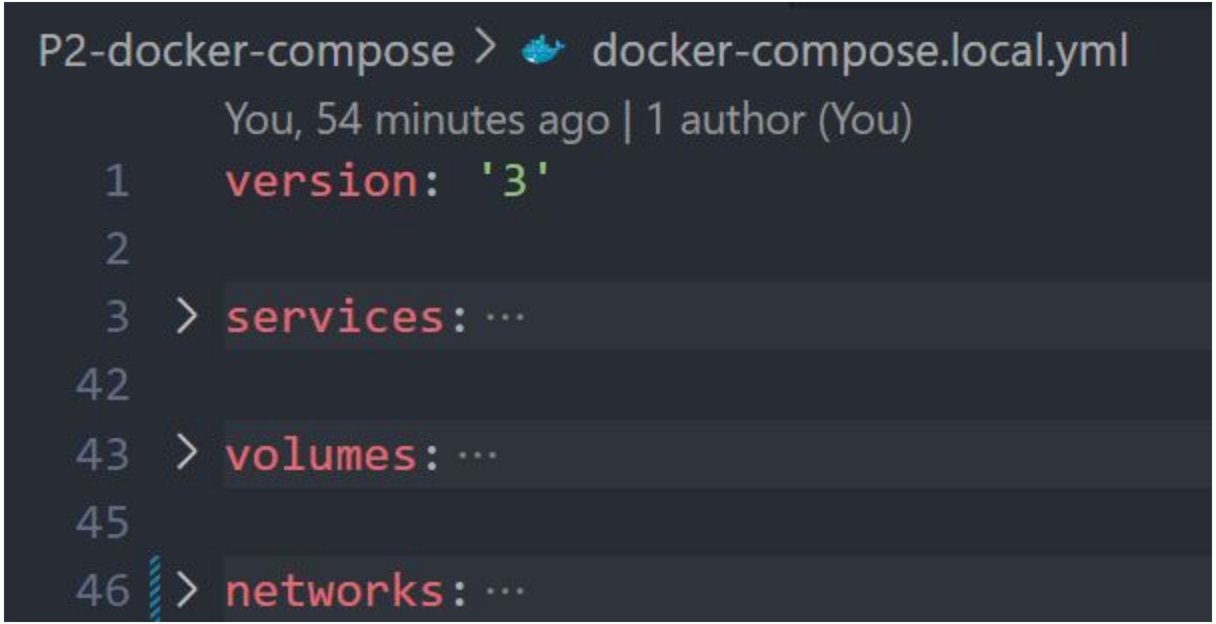
1. Code the microservices
2. Create yml
3. Deploy infrastructure

1. MICROSERVICES



```
P2-docker-compose
  backend
  frontend
  docker-compose.local.yml
```

2. yml file

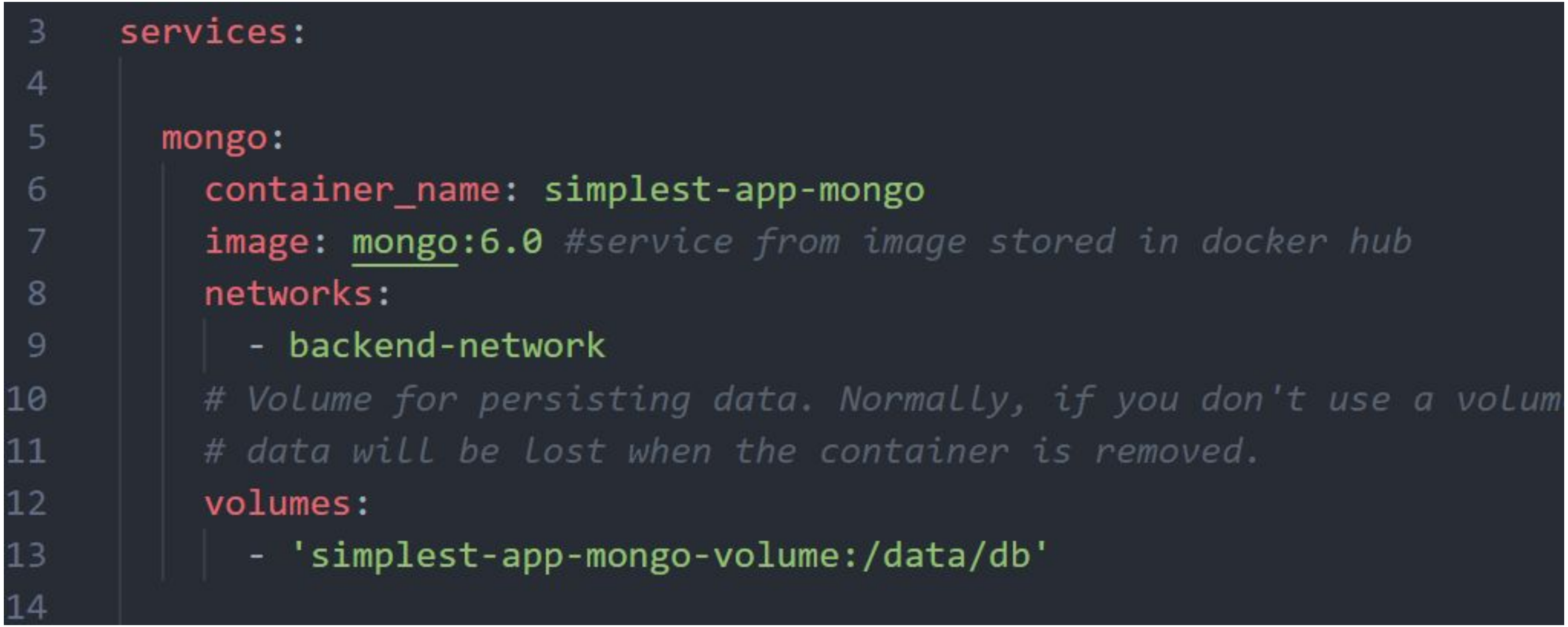


P2-docker-compose > docker-compose.local.yml

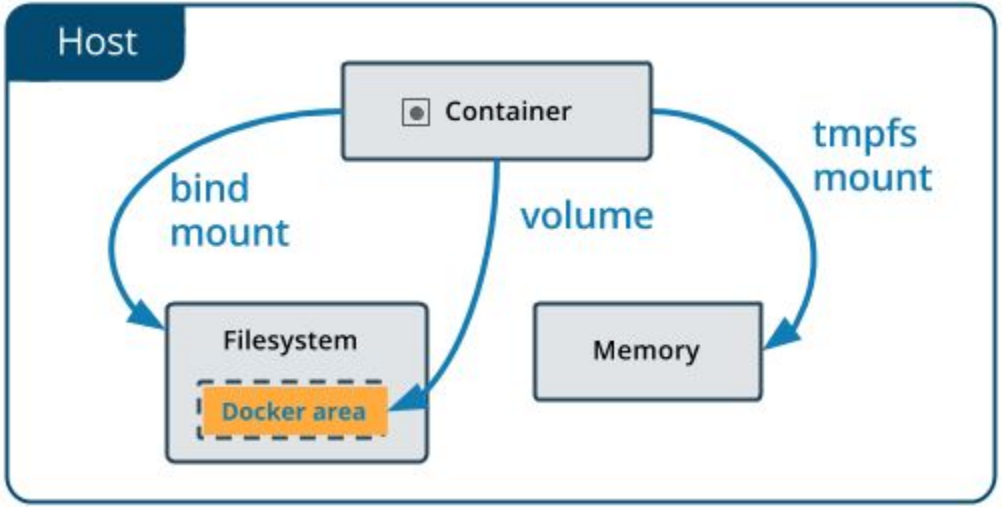
You, 54 minutes ago | 1 author (You)

```yaml
version: '3'

services: ···

volumes: ···

networks: ···
```

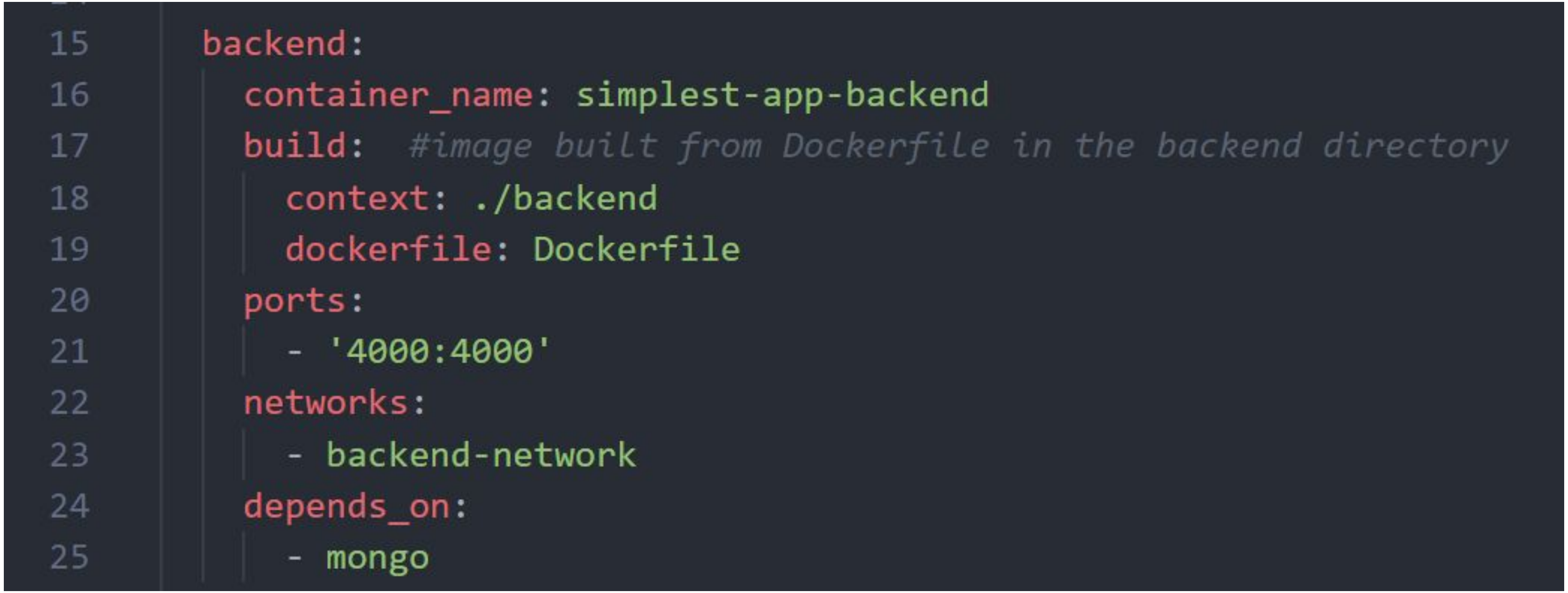
2. yml file



```yaml
services:

  mongo:
    container_name: simplest-app-mongo
    image: mongo:6.0 #service from image stored in docker hub
    networks:
      - backend-network
    # Volume for persisting data. Normally, if you don't use a volum
    # data will be lost when the container is removed.
    volumes:
      - 'simplest-app-mongo-volume:/data/db'

```

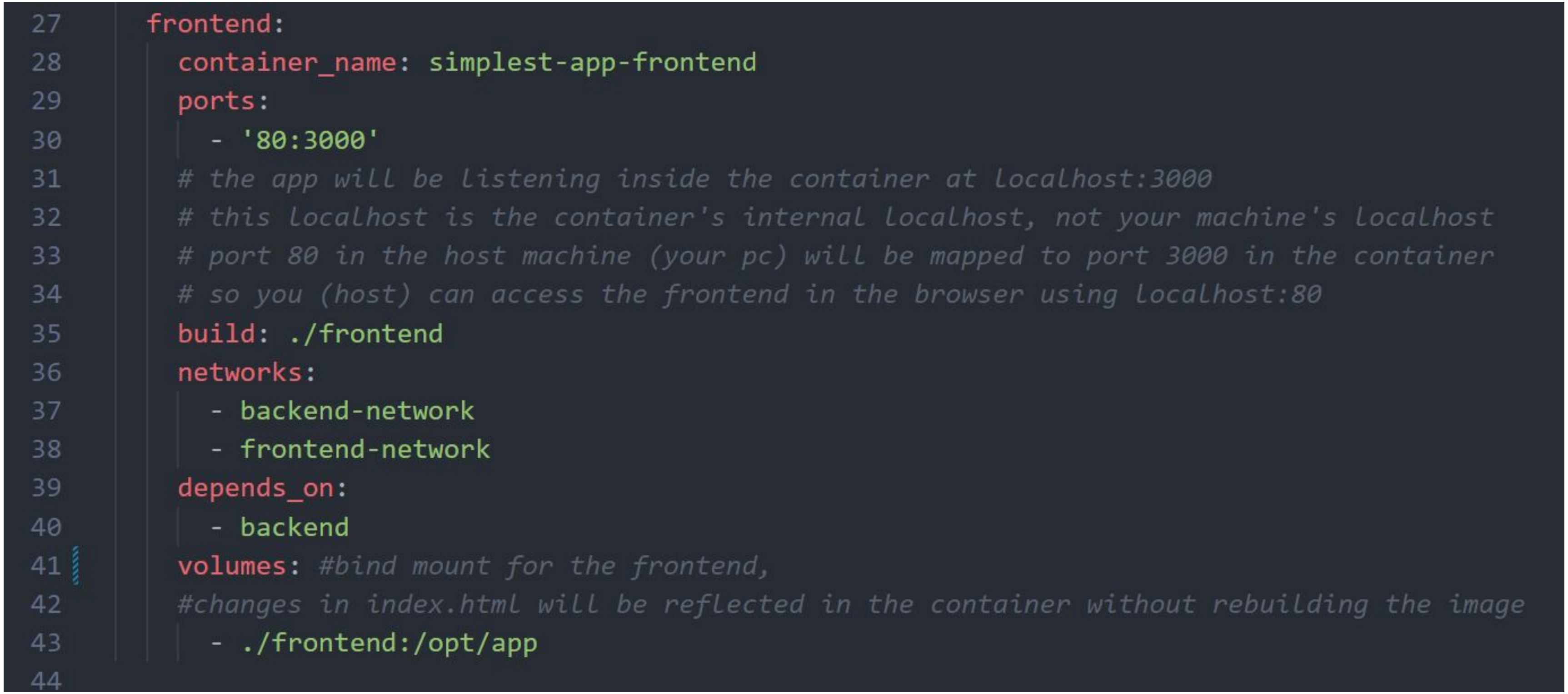

Host
Container
bind mount
volume
tmpfs mount
Filesystem
Docker area
Memory

2. yml file



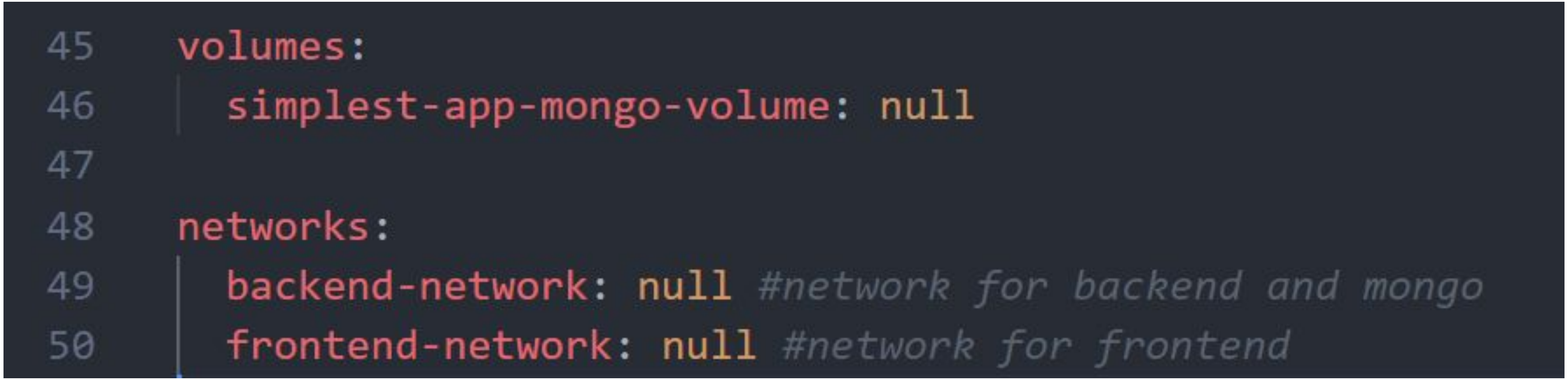
```yaml
  backend:
    container_name: simplest-app-backend
    build:  #image built from Dockerfile in the backend directory
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - '4000:4000'
    networks:
      - backend-network
    depends_on:
      - mongo
```

2. yml file



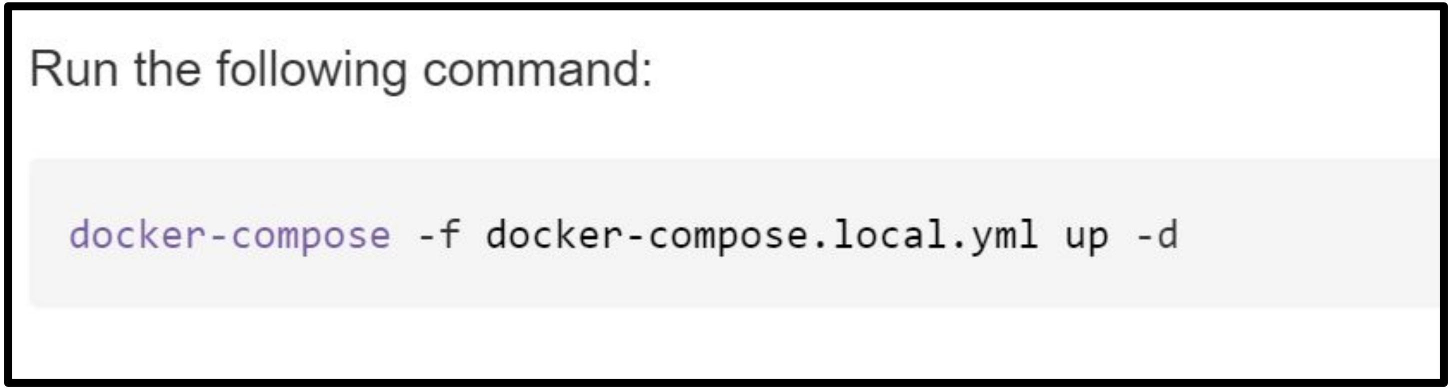
```yaml
  frontend:
    container_name: simplest-app-frontend
    ports:
      - '80:3000'
    # the app will be listening inside the container at localhost:3000
    # this localhost is the container's internal localhost, not your machine's localhost
    # port 80 in the host machine (your pc) will be mapped to port 3000 in the container
    # so you (host) can access the frontend in the browser using localhost:80
    build: ./frontend
    networks:
      - backend-network
      - frontend-network
    depends_on:
      - backend
    volumes: #bind mount for the frontend,
    #changes in index.html will be reflected in the container without rebuilding the image
      - ./frontend:/opt/app

```

2. yml file



```yaml
volumes:
  simplest-app-mongo-volume: null

networks:
  backend-network: null #network for backend and mongo
  frontend-network: null #network for frontend
```

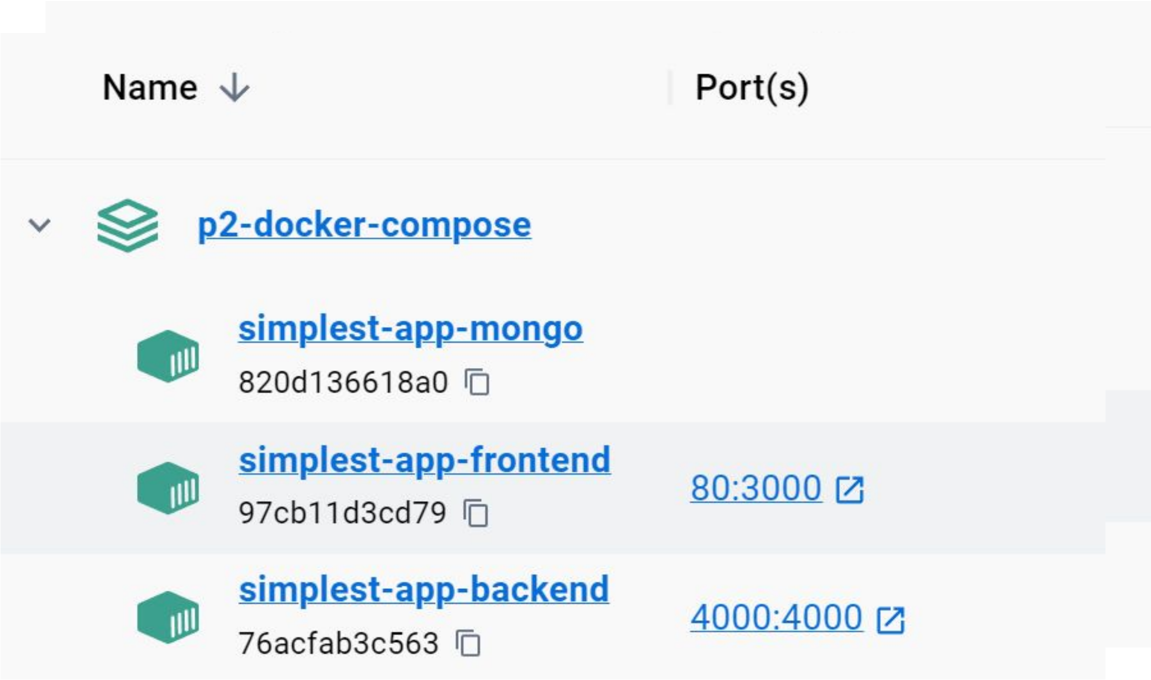
3. Deploy infrastructure



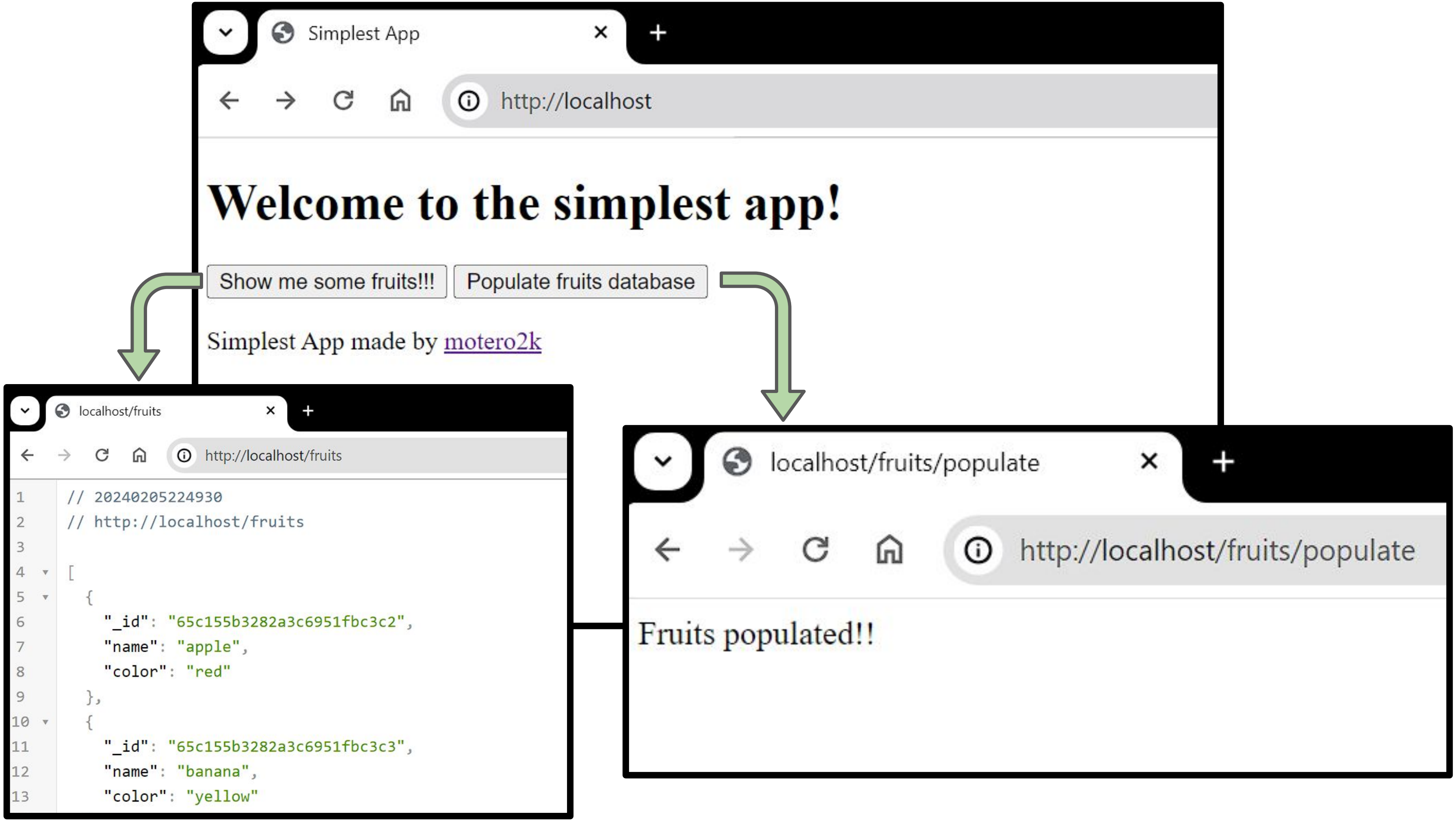
Run the following command:

```
docker-compose -f docker-compose.local.yml up -d
```

3. Deploy infrastructure



Name ↓	Port(s)
p2-docker-compose	
simplest-app-mongo 820d136618a0	
simplest-app-frontend 97cb11d3cd79	80:3000
simplest-app-backend 76acfab3c563	4000:4000



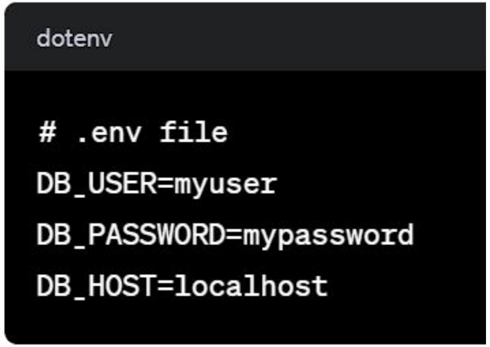
Welcome to the simplest app!

Simplest App made by motero2k

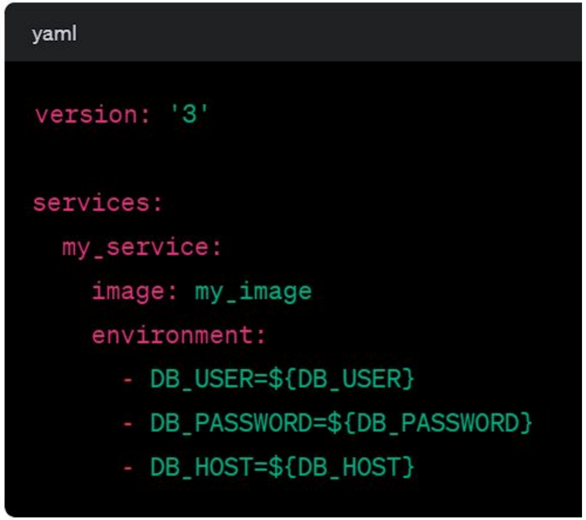
```
// 20240205224930
// http://localhost/fruits

[
  {
    "_id": "65c155b3282a3c6951fbc3c2",
    "name": "apple",
    "color": "red"
  },
  {
    "_id": "65c155b3282a3c6951fbc3c3",
    "name": "banana",
    "color": "yellow"
```

Fruits populated!!

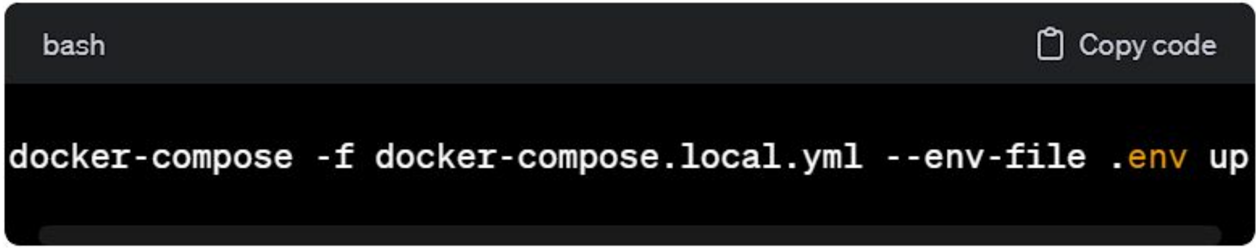
3. Deploy infrastructure



```
# .env file
DB_USER=myuser
DB_PASSWORD=mypassword
DB_HOST=localhost
```



```
version: '3'

services:
  my_service:
    image: my_image
    environment:
      - DB_USER=${DB_USER}
      - DB_PASSWORD=${DB_PASSWORD}
      - DB_HOST=${DB_HOST}
```



```
docker-compose -f docker-compose.local.yml --env-file .env up
```

THE END