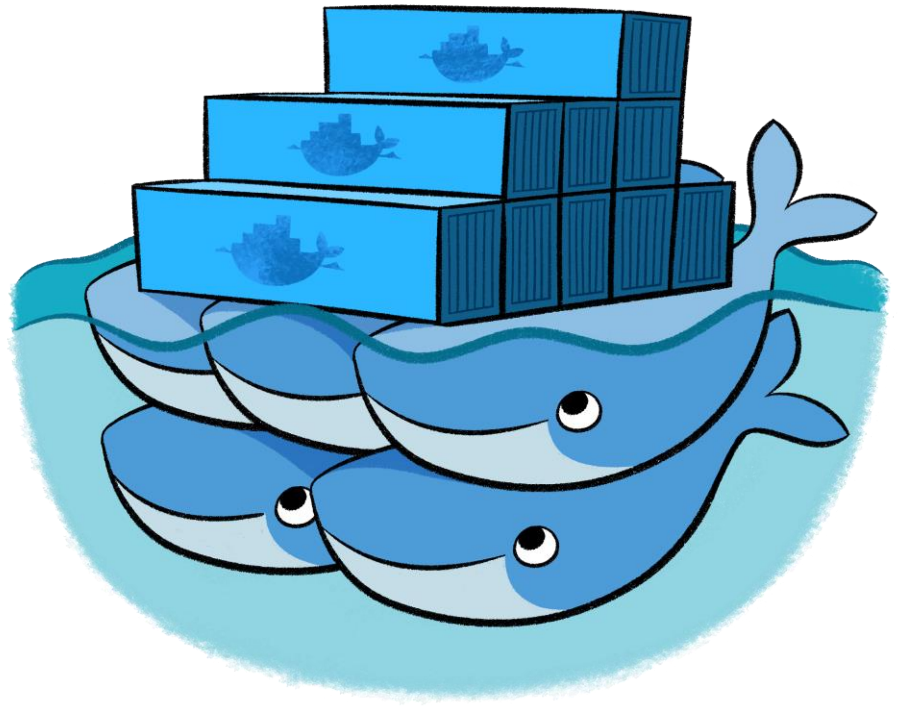
Introduction to Docker

An overview of Docker, deploying applications, industry trends, and how to containerize a Python REST API

Kyle O'Brien, Director of Software Engineering @ CruzHacks



Github.com/CruzHacks/CruzHacks_Workshops_20

Dependencies

Docker

- https://docs.docker.com/install/

Python & Pip

- https://www.python.org/downloads/

Flask

- https://www.python.org/downloads/

Git

- https://git-scm.com/downloads

About Me!

Name: Kyle O'Brien

Education: 5th Year B.A. Computer Science Major

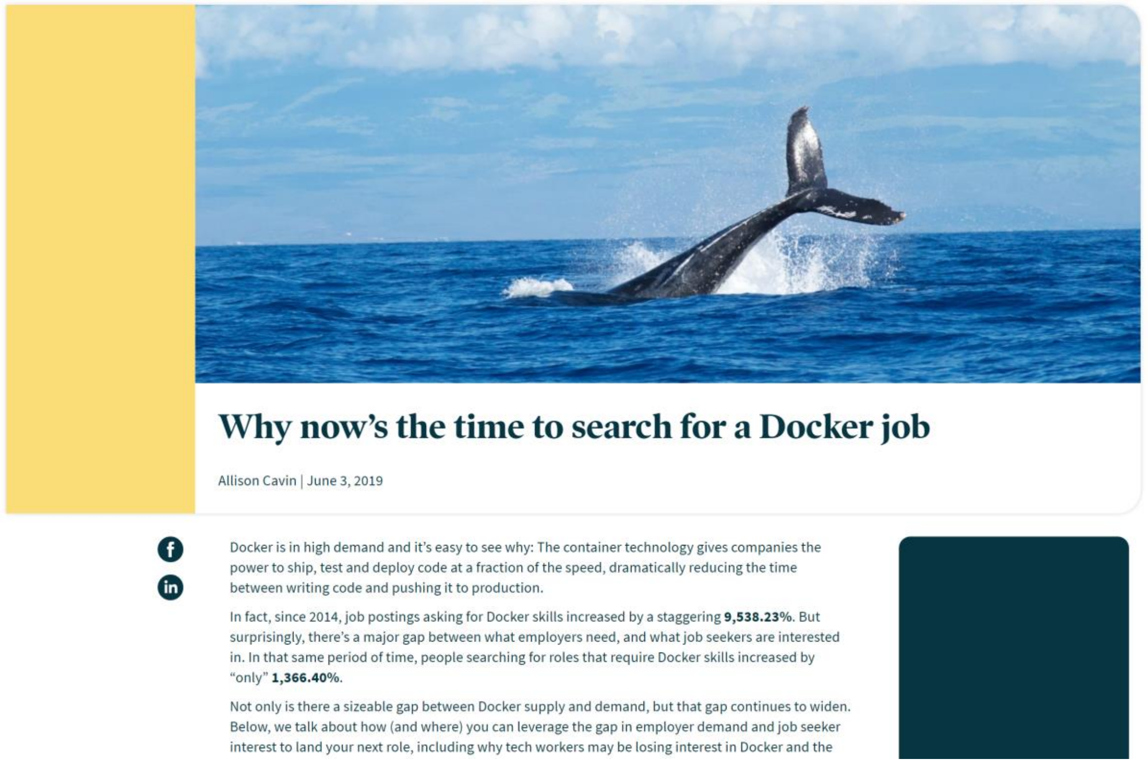
Role: Director of Software Engineering at CruzHacks

Interests: Cloud Computing and Technical Leadership

Medium: @kyleobrien1668

LinkedIn: /in/kyle1668/

Docker is in demand!



Why now's the time to search for a Docker job

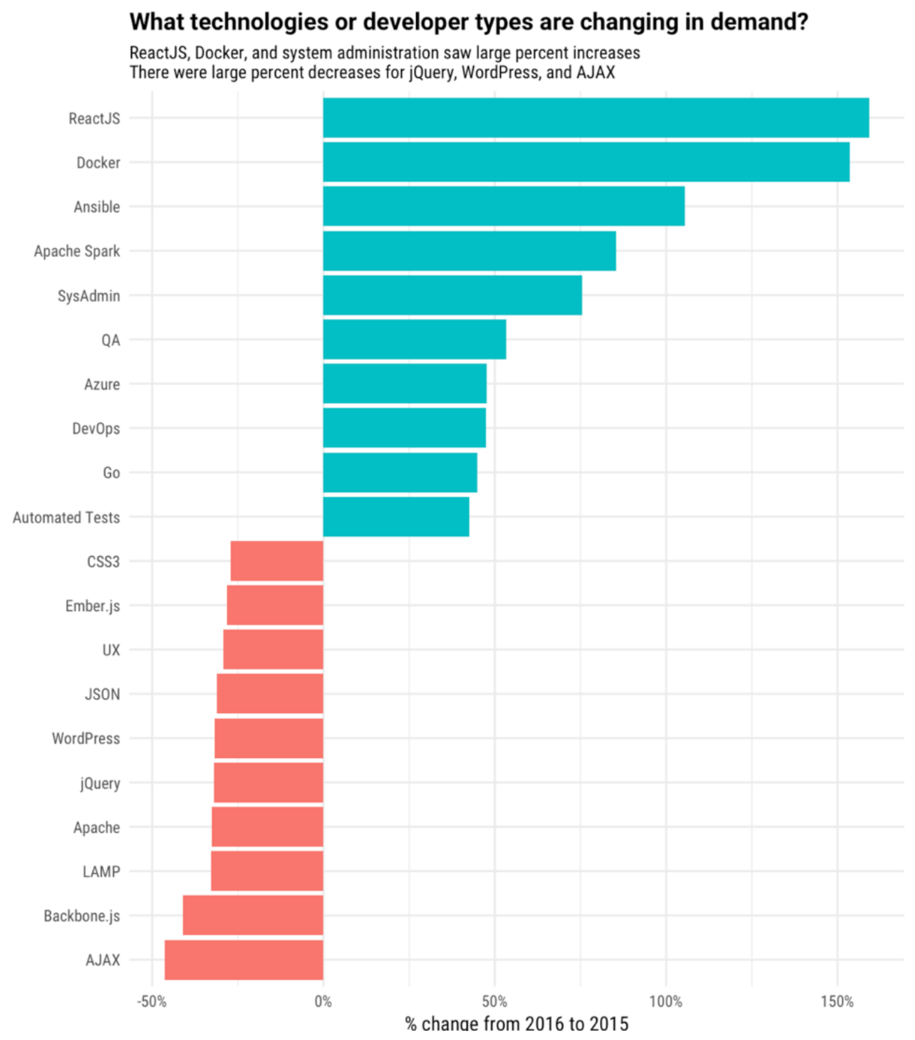
Allison Cavin | June 3, 2019

Docker is in high demand and it's easy to see why: The container technology gives companies the power to ship, test and deploy code at a fraction of the speed, dramatically reducing the time between writing code and pushing it to production.

In fact, since 2014, job postings asking for Docker skills increased by a staggering **9,538.23%**. But surprisingly, there's a major gap between what employers need, and what job seekers are interested in. In that same period of time, people searching for roles that require Docker skills increased by "only" **1,366.40%**.

Not only is there a sizeable gap between Docker supply and demand, but that gap continues to widen. Below, we talk about how (and where) you can leverage the gap in employer demand and job seeker interest to land your next role, including why tech workers may be losing interest in Docker and the

Source: beseen.com



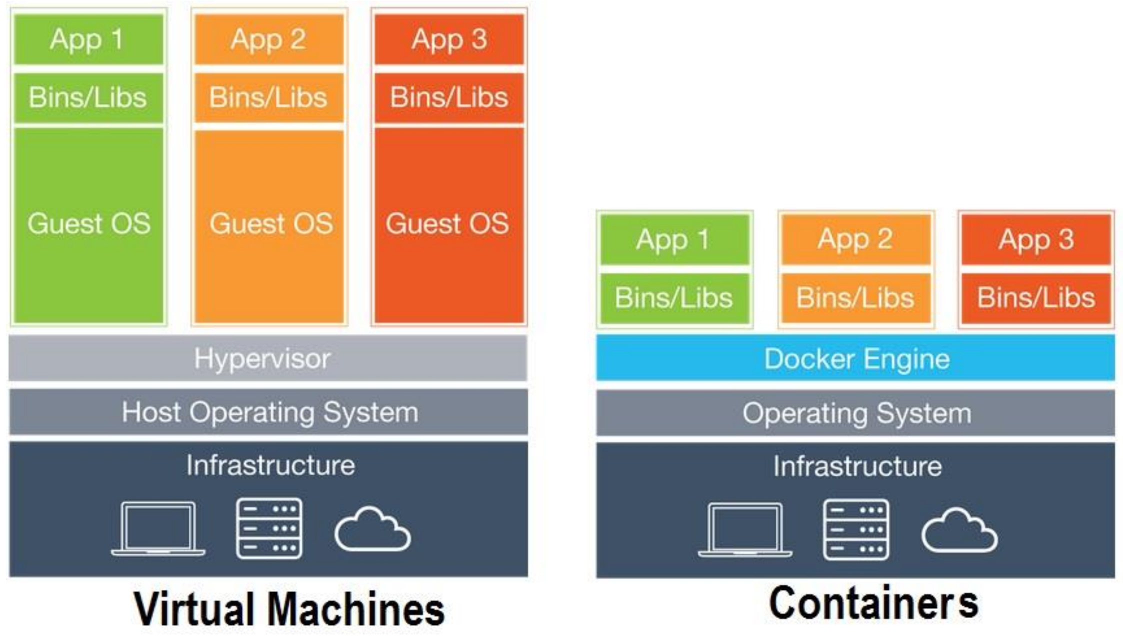
Source: 2017 Stack Overflow Develop Survey

What is Docker?

Docker is a tool used to build and deploy applications inside containers!

Containers are light weight alternatives to virtual machines.

Docker allows developers to run their applications inside virtual environments.



Benefits: Cross-Platform Compatibility

With Docker, apps run inside the containers.

Your host operating system interacts with the containers via the Docker Engine, not with the code itself.

Thus you can run apps cross platform inside of containers that would otherwise be platform dependent!

Implications
- No need to worry about what OS your app is being deployed to.

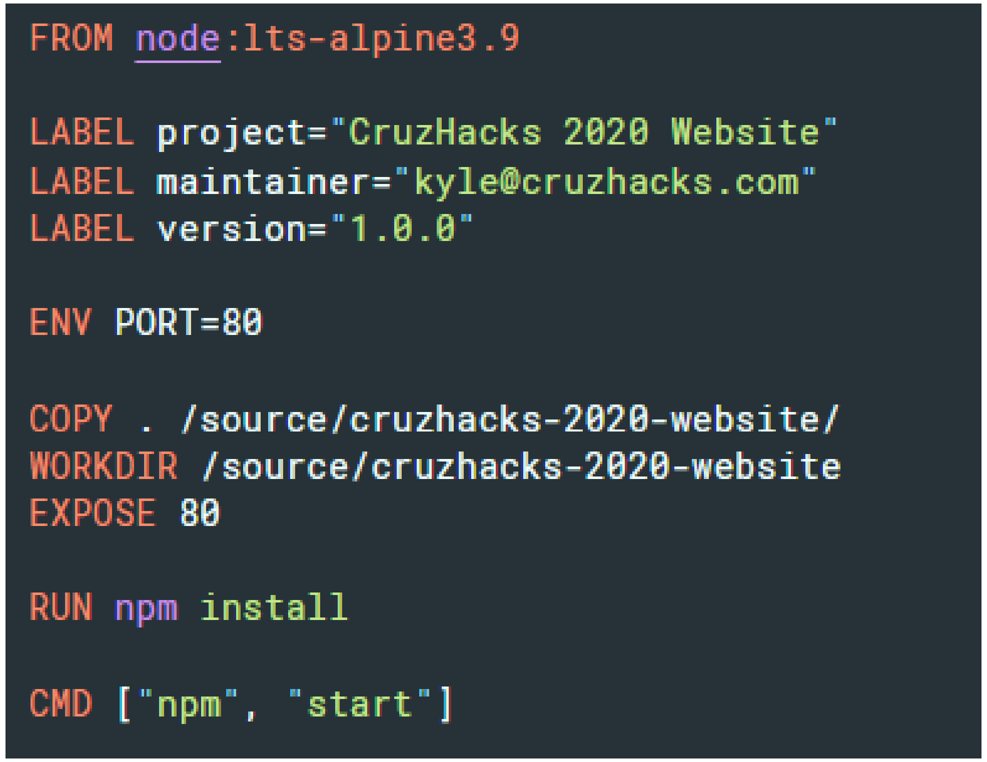
- No more, “but it runs on my machine”

Benefits: Automated Builds

Each Docker image is created using a Dockerfile.

Dockerfiles contain instructions for how to build the image.

These include files to copy from the local file system, which ports to expose, and a start command.



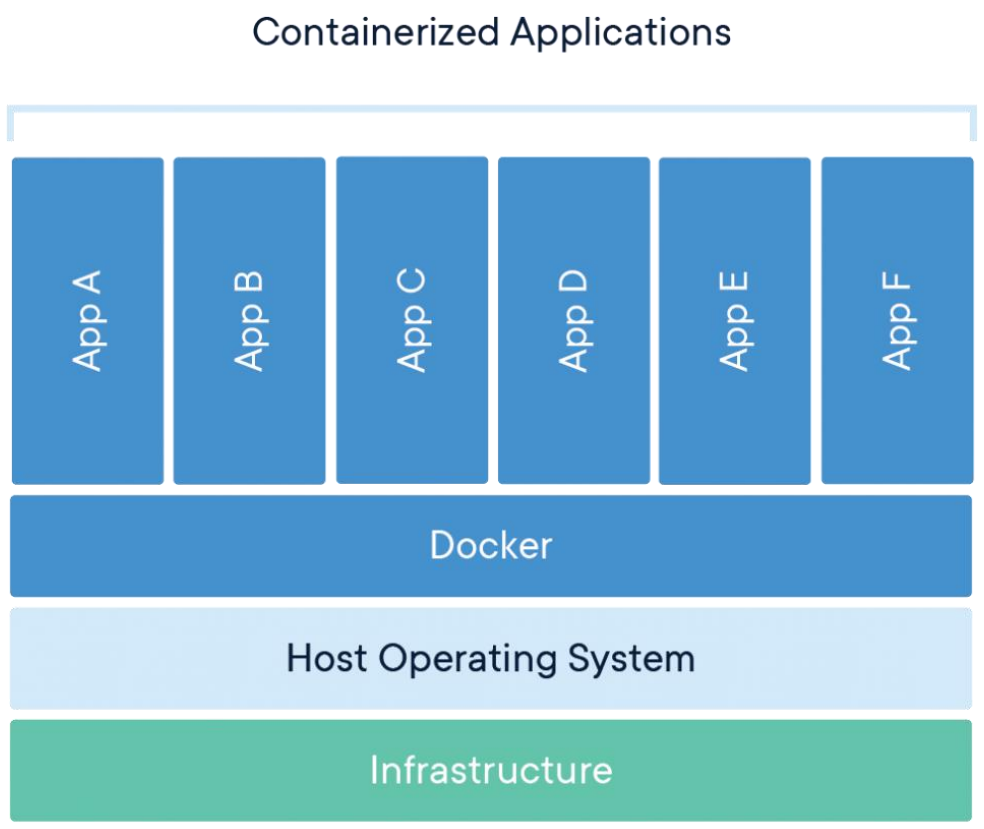
```
FROM node:lts-alpine3.9

LABEL project="CruzHacks 2020 Website"
LABEL maintainer="kyle@cruzhacks.com"
LABEL version="1.0.0"

ENV PORT=80

COPY . /source/cruzhacks-2020-website/
WORKDIR /source/cruzhacks-2020-website
EXPOSE 80

RUN npm install

CMD ["npm", "start"]
```

Benefits: Isolation and Security

Each container only includes the minimal resources it needs to run your app.

Multiple containers can run on the same host and be unaware of each other.

This isolation increases security and decreases complexity.

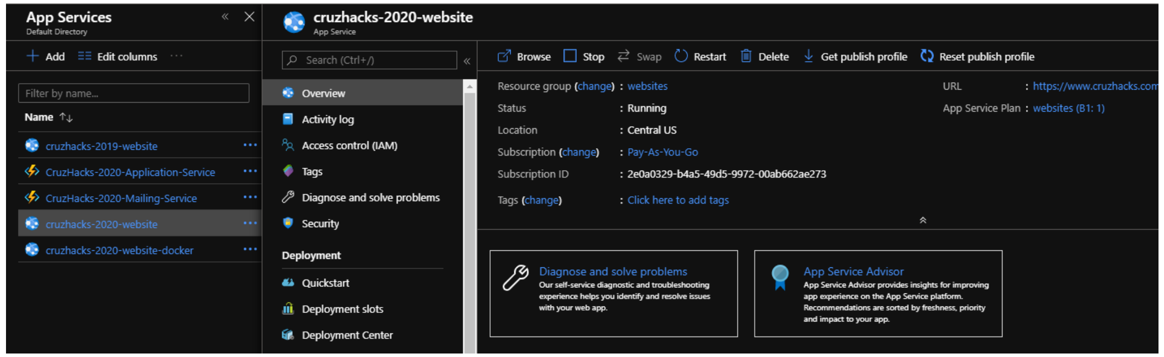


CruzHacks.com Case Study

We used an NPM package that does not work on Windows.

We wanted to be able to replicate our production environment locally.

We wanted to have consistent automated deployments on Azure.



Resource group (change) : websites
Status : Running
Location : Central US
Subscription (change) : Pay-As-You-Go
Subscription ID : 2e0a0329-b4a5-49d5-9972-00ab662ae273
Tags (change) : Click here to add tags
URL : https://www.cruzhacks.com
App Service Plan : websites (B1: 1)

Diagnose and solve problems
Our self-service diagnostic and troubleshooting experience helps you identify and resolve issues with your web app.

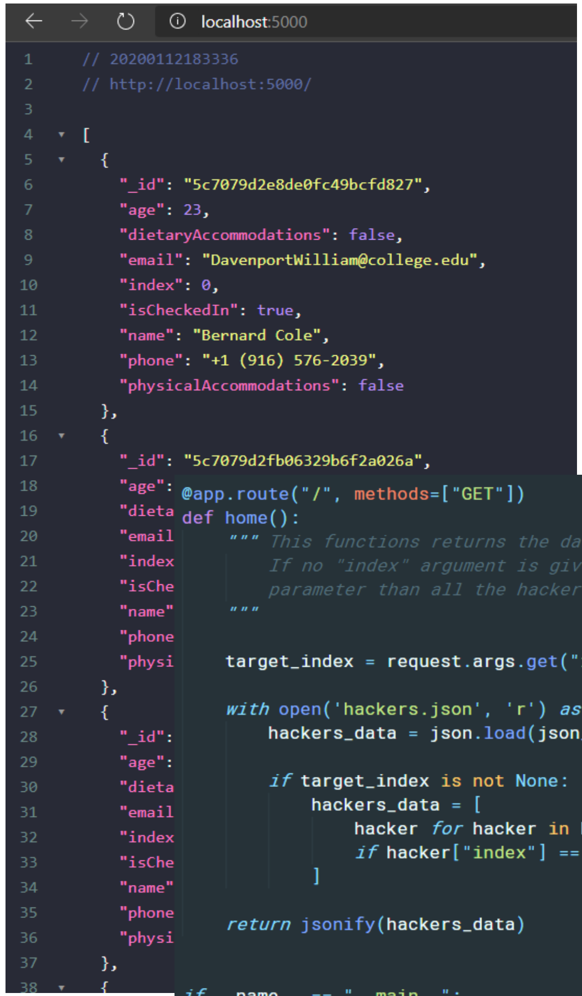
App Service Advisor
App Service Advisor provides insights for improving app experience on the App Service platform. Recommendations are sorted by freshness, priority and impact to your app.

Example: Dockerize a Python REST API

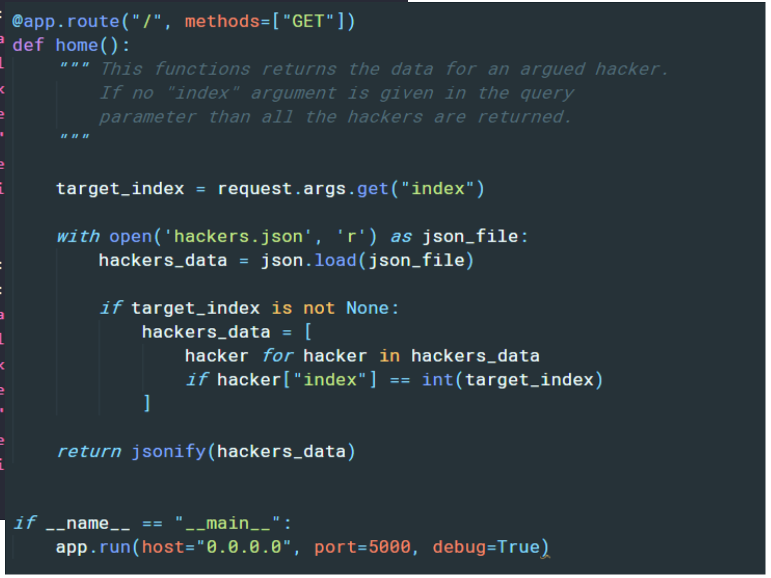
You’re an engineer on the CruzHacks team! The logistics team wants a dashboard to be able to query data about attendees (hackers). They’ve requested the engineering team to build with dashboard.

You’ve been tasked with deploying the backend REST API that the dashboard will used to get the hacker data.

However half the team uses Windows and the other MacOS. The API will be deployed on Linux. Thus it would be a good idea to use Docker.

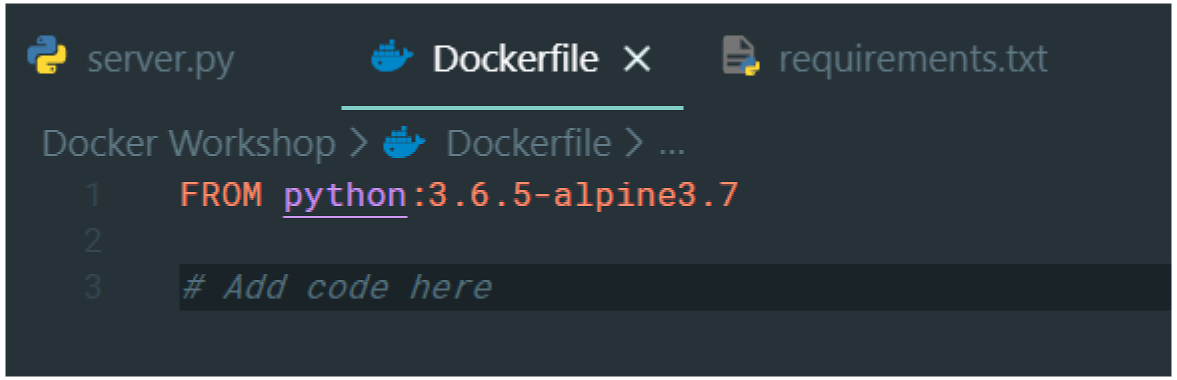


```
// 20200112183336
// http://localhost:5000/

[
  {
    "_id": "5c7079d2e8de0fc49bcfd827",
    "age": 23,
    "dietaryAccommodations": false,
    "email": "DavenportWilliam@college.edu",
    "index": 0,
    "isCheckedIn": true,
    "name": "Bernard Cole",
    "phone": "+1 (916) 576-2039",
    "physicalAccommodations": false
  },
  {
    "_id": "5c7079d2fb06329b6f2a026a",
    "age":
    "dieta
    "email
    "index
    "isChe
    "name"
    "phone
    "physi
  },
  {
    "_id":
    "age":
    "dieta
    "email
    "index
    "isChe
    "name"
    "phone
    "physi
  },
  {
```



```python
@app.route("/", methods=["GET"])
def home():
    """ This functions returns the data for an argued hacker.
        If no "index" argument is given in the query
        parameter than all the hackers are returned.
    """

    target_index = request.args.get("index")

    with open('hackers.json', 'r') as json_file:
        hackers_data = json.load(json_file)

        if target_index is not None:
            hackers_data = [
                hacker for hacker in hackers_data
                if hacker["index"] == int(target_index)
            ]

    return jsonify(hackers_data)


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```

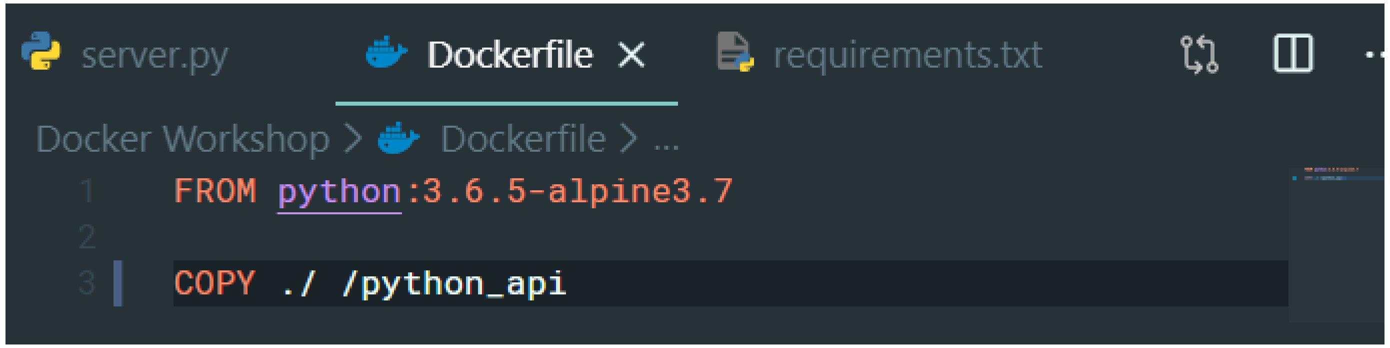
Step 1: Open the Dockerfile



```
FROM python:3.6.5-alpine3.7

# Add code here
```

Step 2: Add the COPY command

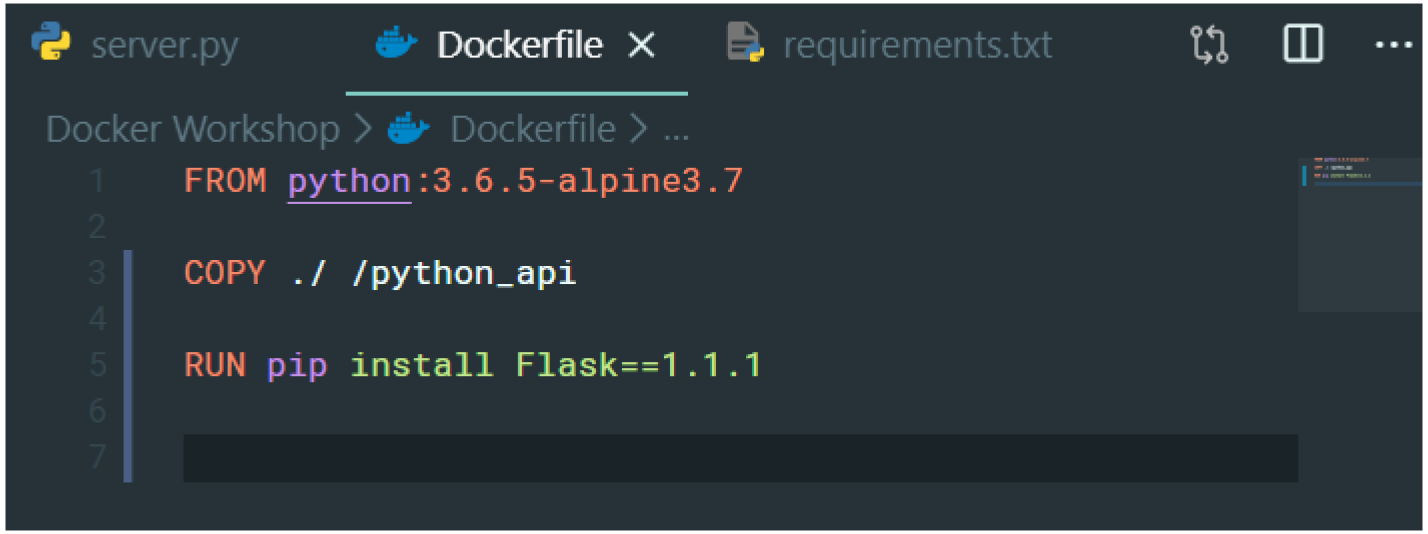
We need to add a command to copy our Python and JSON files into the Docker image's file system. Add the below line to your Dockerfile.



```
FROM python:3.6.5-alpine3.7

COPY ./ /python_api
```

Step 3: Add the RUN command

The Docker RUN command lets you run shell commands when the image is being made. We're going to use it to install Flask using Pip.

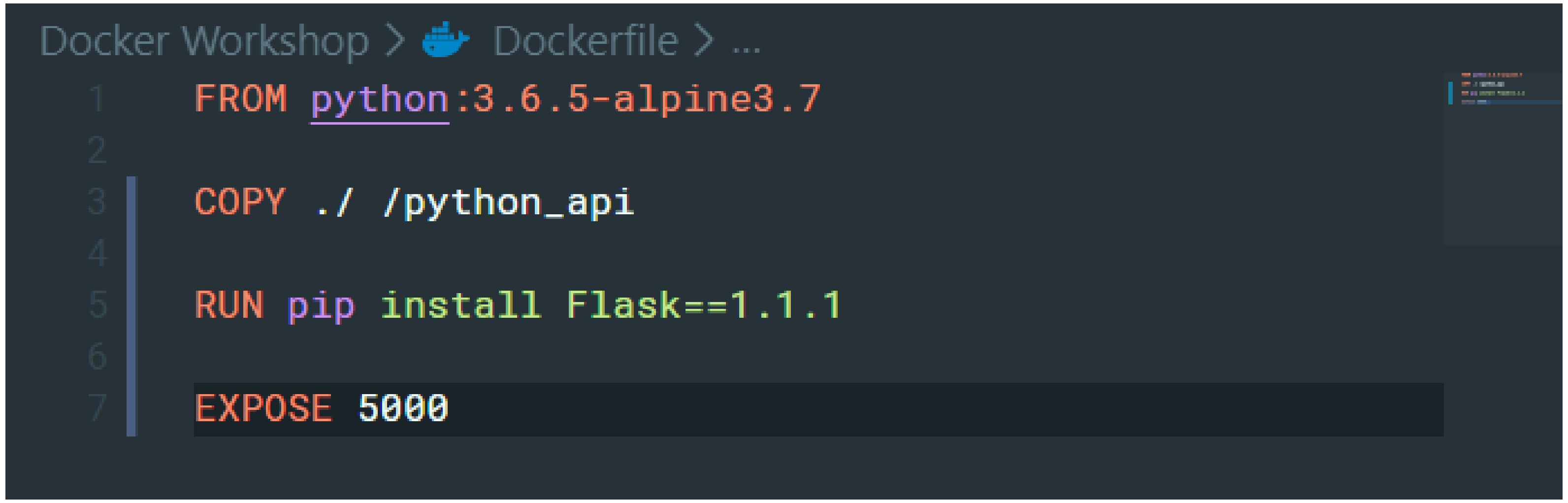


```
FROM python:3.6.5-alpine3.7

COPY ./ /python_api

RUN pip install Flask==1.1.1

```

Step 4: Expose the Container Port

Docker containers have their own network that's separate from localhost. Our server is set to run on port 5000. Thus the server will start on port 5000 INSIDE the container!

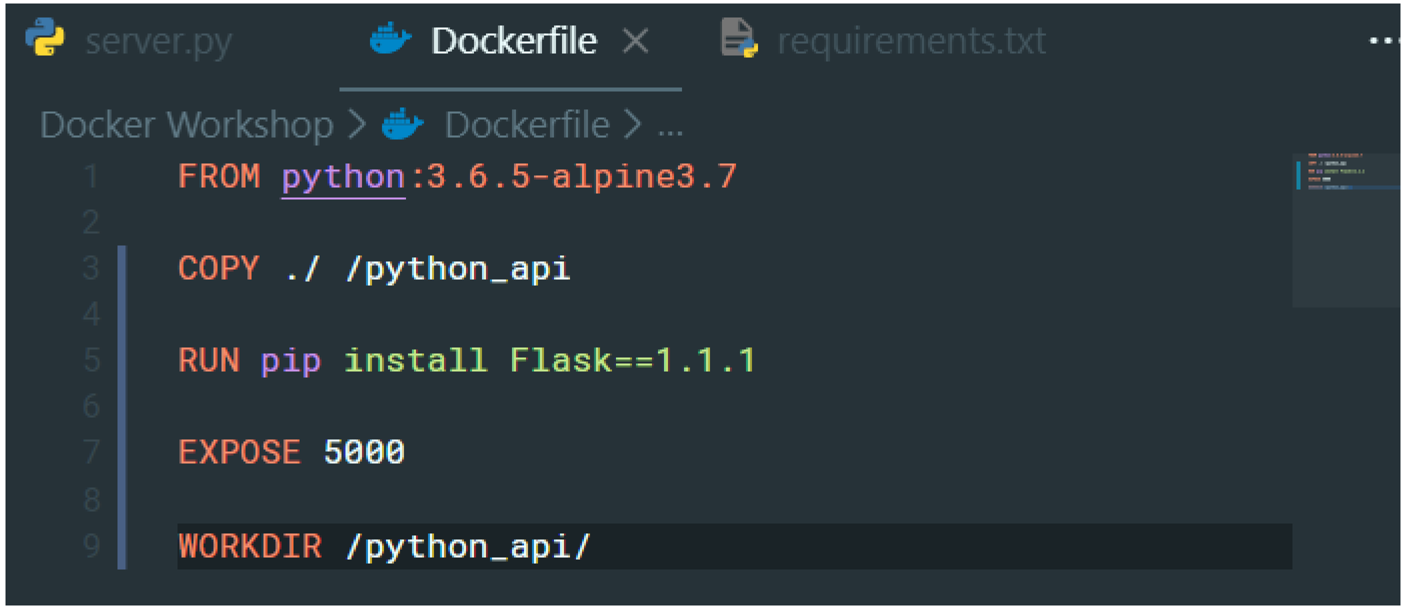
To be able to connect to the server, we need to expose that port.



```
FROM python:3.6.5-alpine3.7

COPY ./ /python_api

RUN pip install Flask==1.1.1

EXPOSE 5000
```

Step 5: Change the Working Directory

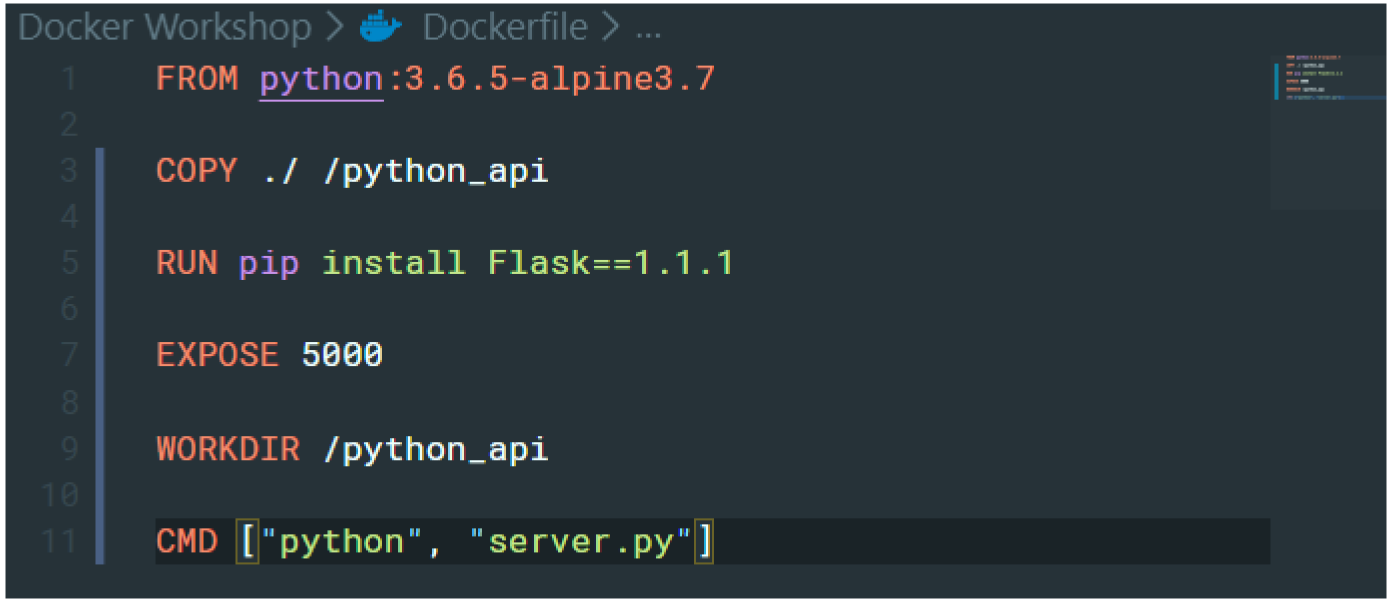
Change the current directory inside the container to where we copied our files into.



```
FROM python:3.6.5-alpine3.7

COPY ./ /python_api

RUN pip install Flask==1.1.1

EXPOSE 5000

WORKDIR /python_api/
```

Step 6: Specify the Startup Command

Finally, we’ll set the command that’s to be run whenever a new container is created. This command will start our Flask server.



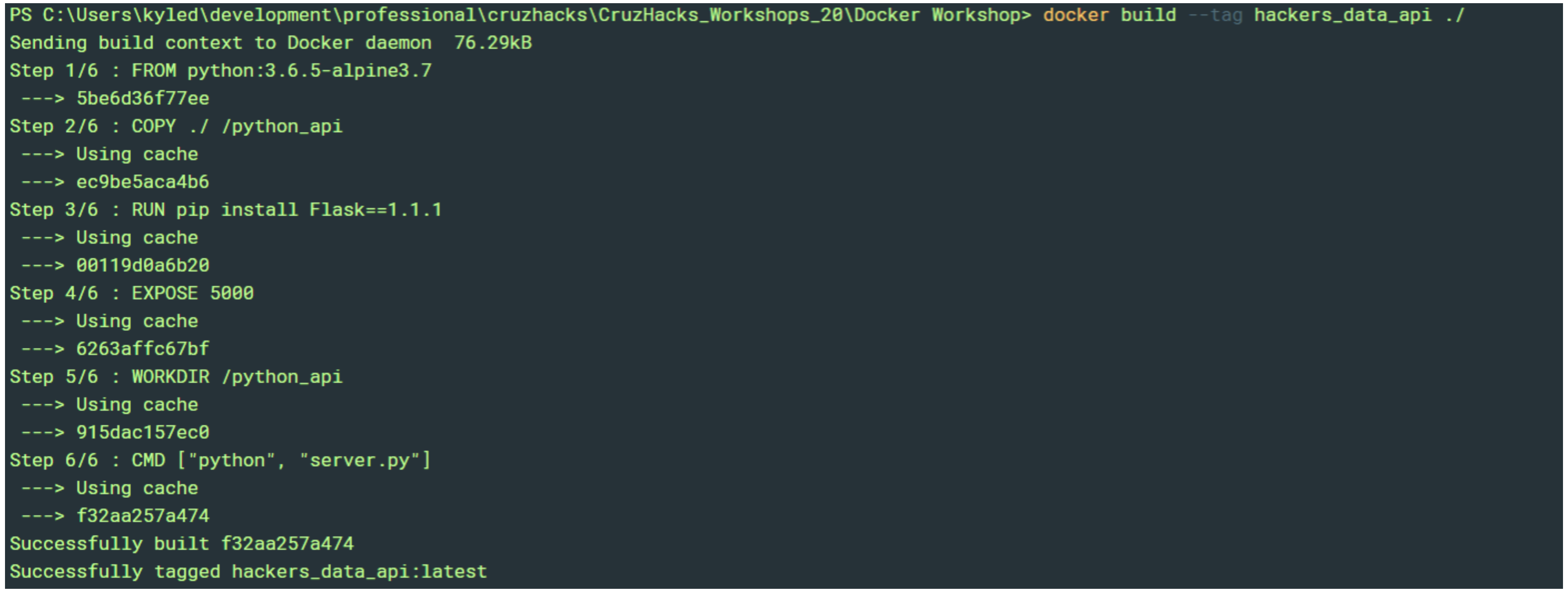
```
FROM python:3.6.5-alpine3.7

COPY ./ /python_api

RUN pip install Flask==1.1.1

EXPOSE 5000

WORKDIR /python_api

CMD ["python", "server.py"]
```

Step 7: Build the Docker Image

Build the Docker image from the Dockerfile by running.

Use the --tag flag to name the image.

```
docker build --tag hackers_data_api ./
```

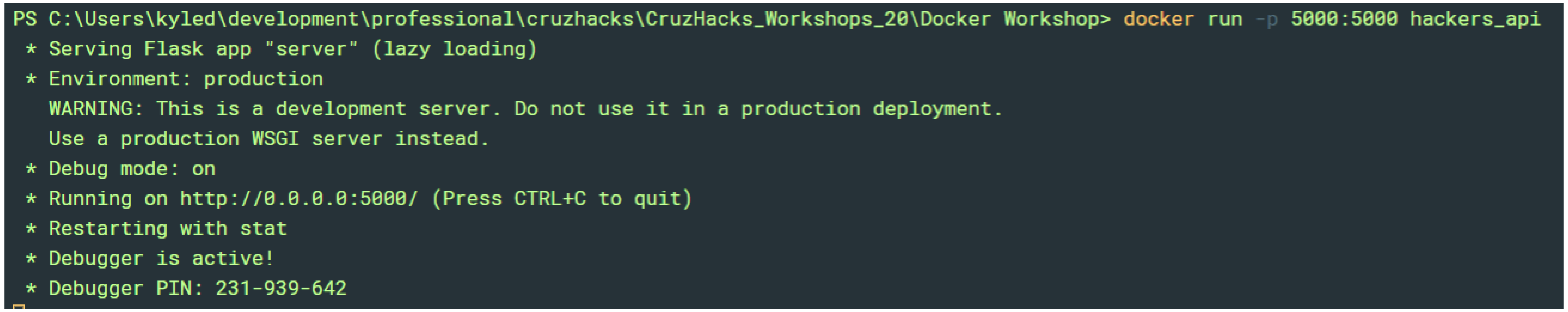


```
PS C:\Users\kyled\development\professional\cruzhacks\CruzHacks_Workshops_20\Docker Workshop> docker build --tag hackers_data_api ./
Sending build context to Docker daemon  76.29kB
Step 1/6 : FROM python:3.6.5-alpine3.7
 ---> 5be6d36f77ee
Step 2/6 : COPY ./ /python_api
 ---> Using cache
 ---> ec9be5aca4b6
Step 3/6 : RUN pip install Flask==1.1.1
 ---> Using cache
 ---> 00119d0a6b20
Step 4/6 : EXPOSE 5000
 ---> Using cache
 ---> 6263affc67bf
Step 5/6 : WORKDIR /python_api
 ---> Using cache
 ---> 915dac157ec0
Step 6/6 : CMD ["python", "server.py"]
 ---> Using cache
 ---> f32aa257a474
Successfully built f32aa257a474
Successfully tagged hackers_data_api:latest
```

Step 8: Create and Run a Docker Container

Now that we have an image we can start making containers from it. We'll map port 5000 on localhost to the exposed port inside the container.

```
docker run -p 5000:5000 hackers_api
```



```
PS C:\Users\kyled\development\professional\cruzhacks\CruzHacks_Workshops_20\Docker Workshop> docker run -p 5000:5000 hackers_api
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 231-939-642
```

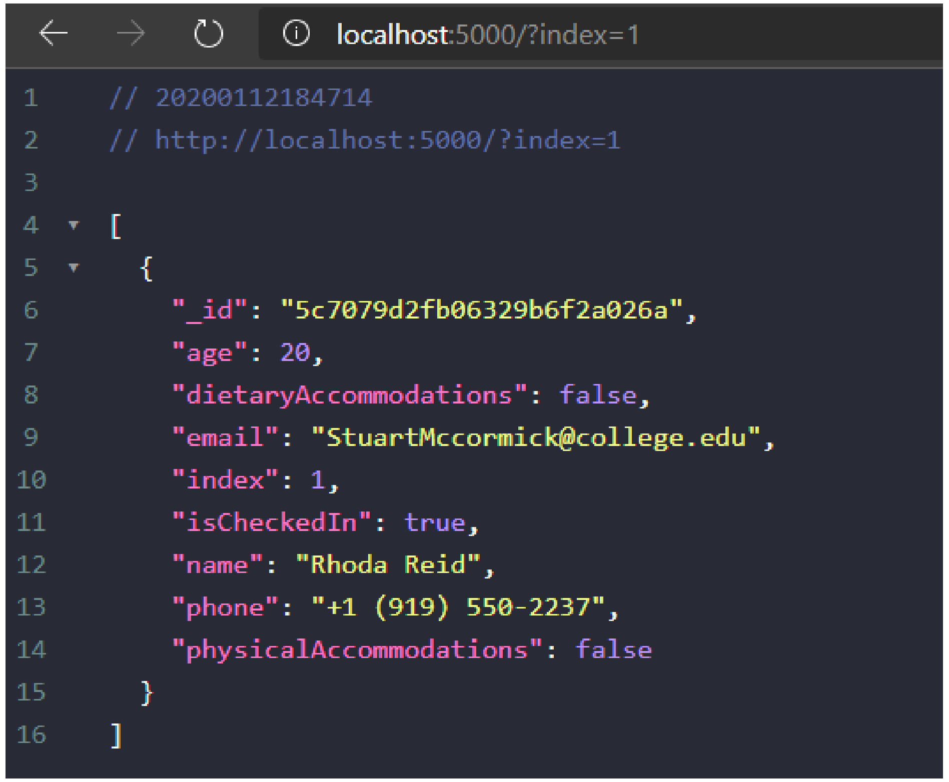
Step 9: Test our API

Go to http://localhost:5000/ to see all hackers as JSON

Go to http://localhost:5000/?index=1

To see data for the hacker with

an index of 1



```
// 20200112184714
// http://localhost:5000/?index=1

[
  {
    "_id": "5c7079d2fb06329b6f2a026a",
    "age": 20,
    "dietaryAccommodations": false,
    "email": "StuartMccormick@college.edu",
    "index": 1,
    "isCheckedIn": true,
    "name": "Rhoda Reid",
    "phone": "+1 (919) 550-2237",
    "physicalAccommodations": false
  }
]
```

Congratulations

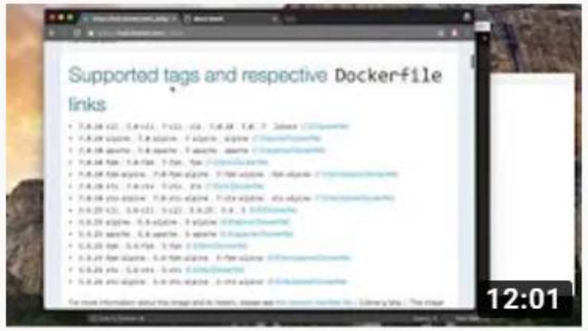
You’ve created a Docker Image from which we can now create containers from. We can now deploy this API in the cloud.

You’ve made this API more…

- Secure
- Scalable
- Maintainable

And you became more employable!

Further Learning: YouTube



Learn Docker in 12 Minutes 🐳

Jake Wright • 1.4M views • 3 years ago

Docker is all the rage right now. In **12 minutes** I'll give you comprehensive introduction to **docker**, covering: 1. What is **Docker** 2.

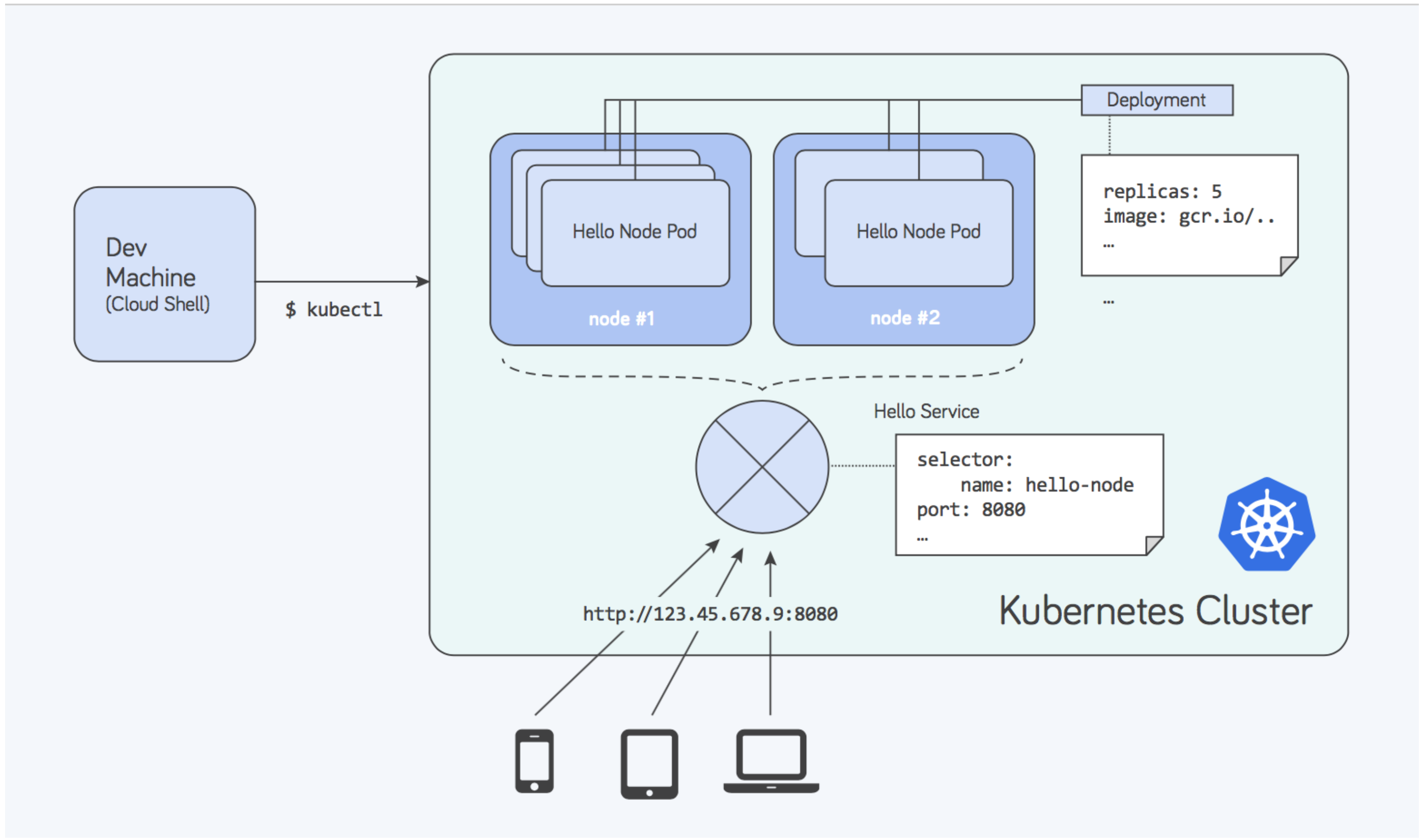
```
version: "3"

services:
  product-service:
    build: ./product
    volumes:
      - ./product:/usr/src/app
    ports:
      - 5001:80

  website:
    image: php:apache
    volumes:
      - ./website:/var/www/html
    ports:
      - 5000:80
    depends_on:
      - product-service
```

Docker Compose in 12 Minutes

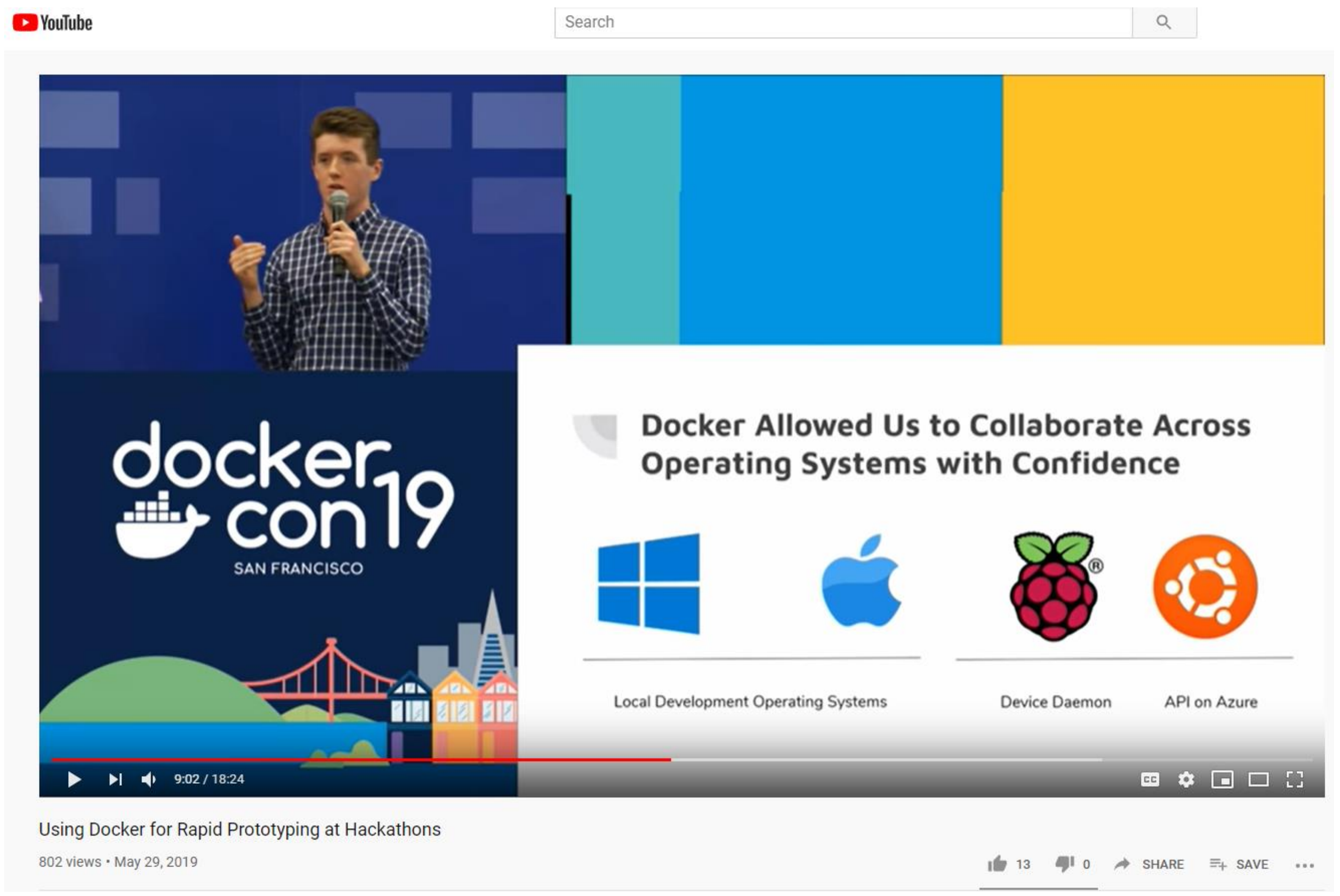
Jake Wright • 587K views • 2 years ago

Learn how to use **Docker** Compose to run multi-container applications easily. This is the second video in this **Docker** series. Learn ...

Further Learning: Container Orchestration



Further Learning: Using Docker at Hackathons



Docker Allowed Us to Collaborate Across Operating Systems with Confidence

Local Development Operating Systems

Device Daemon

API on Azure

Using Docker for Rapid Prototyping at Hackathons

802 views • May 29, 2019

End