# DOCKER AND PYTHON – A MATCH MADE IN HEAVEN

Docker and Python · Hendrik Niemeyer · © ROSEN Group · 10-Oct-2019

ROSEN
empowered by technology

# WHOAMI

**Hendrik Niemeyer**

Theoretical Physics

Technical Lead Data Science

@hniemeye

https://github.com/hniemeyer

hniemeyer@rosen-group.com

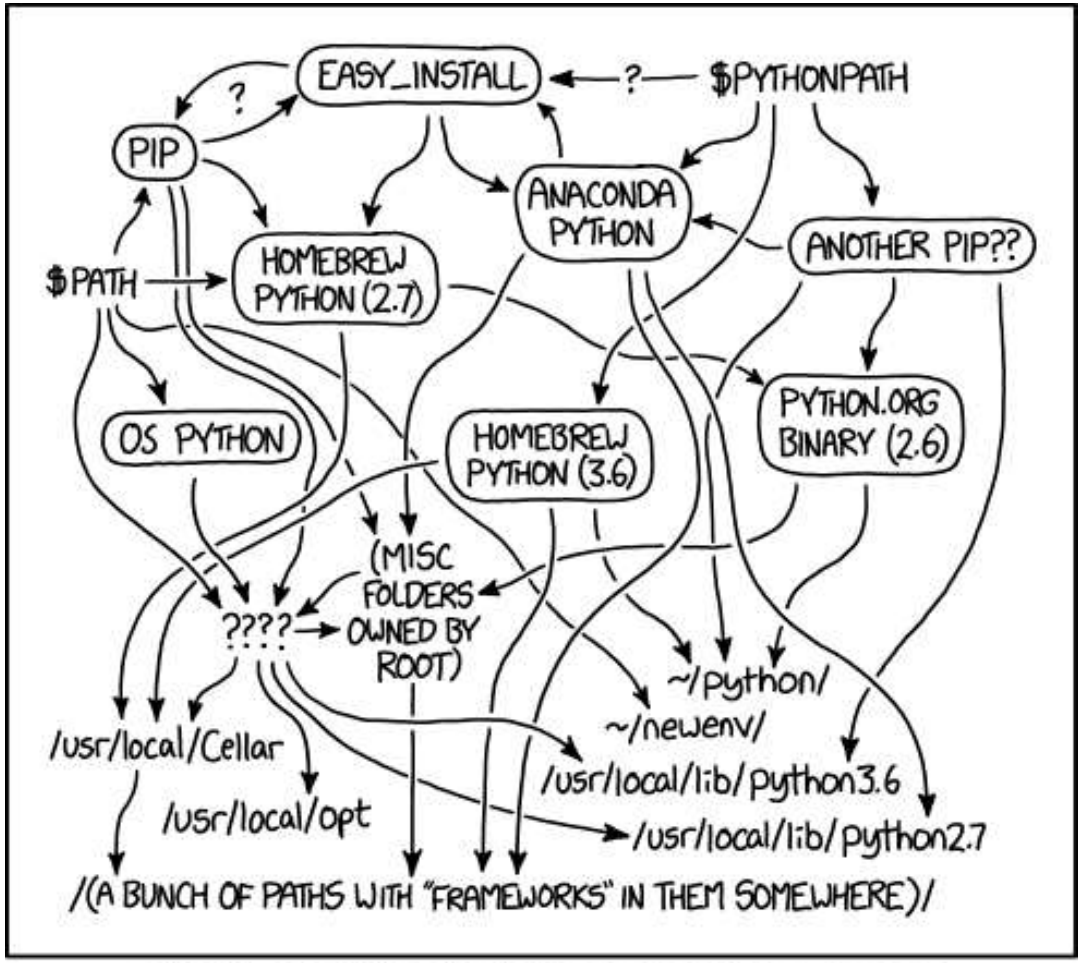
Slides: https://github.com/rosen-group/conferences

# INTRODUCING THE ROSEN GROUP

ROSEN develops and manufactures equipment, software, and methods for the inspection, diagnosis, and protection of industrial structures in a wide range of industries.

Because damage can cause serious impacts!

# MOTIVATION

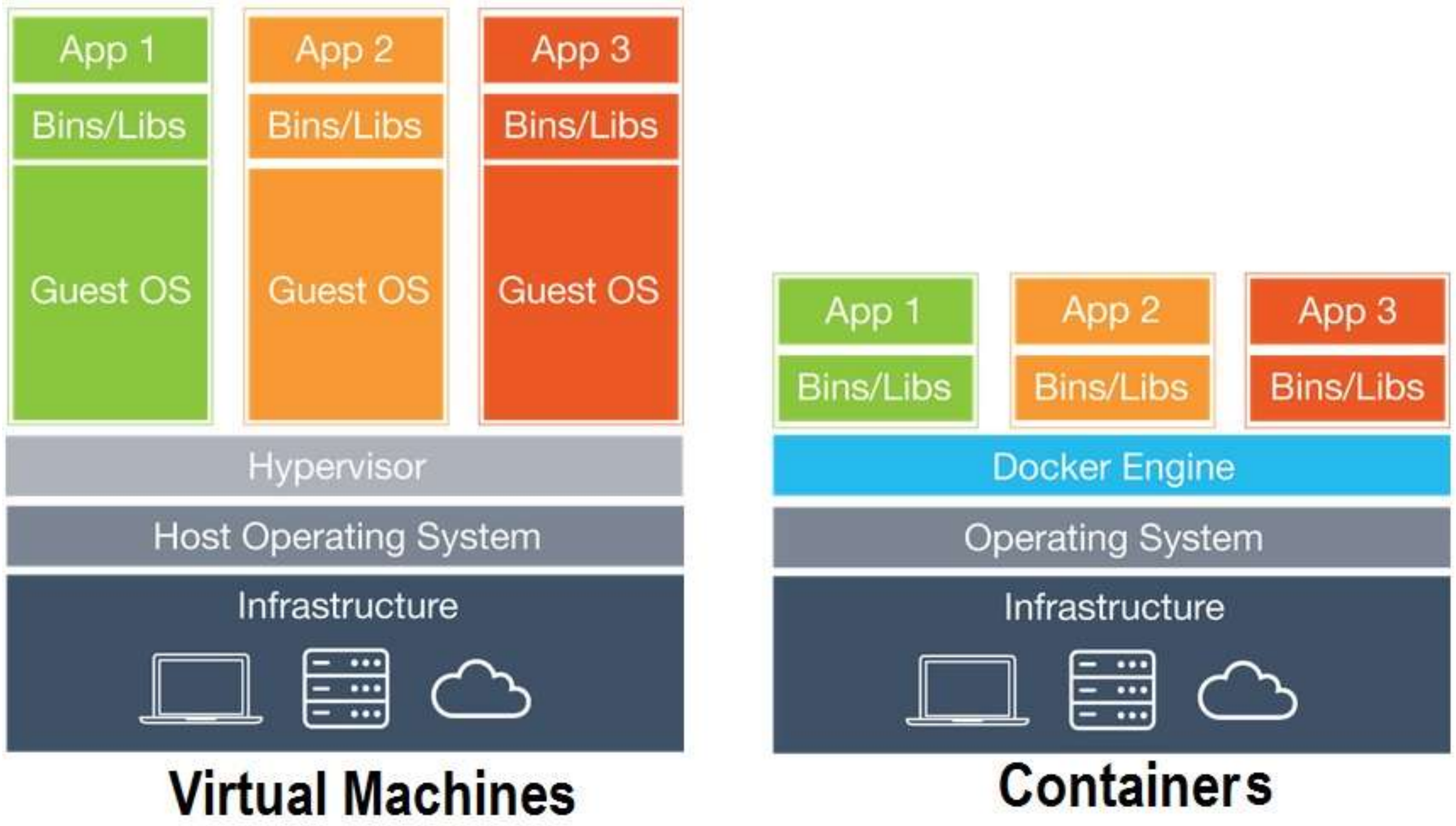


MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

# MOTIVATION

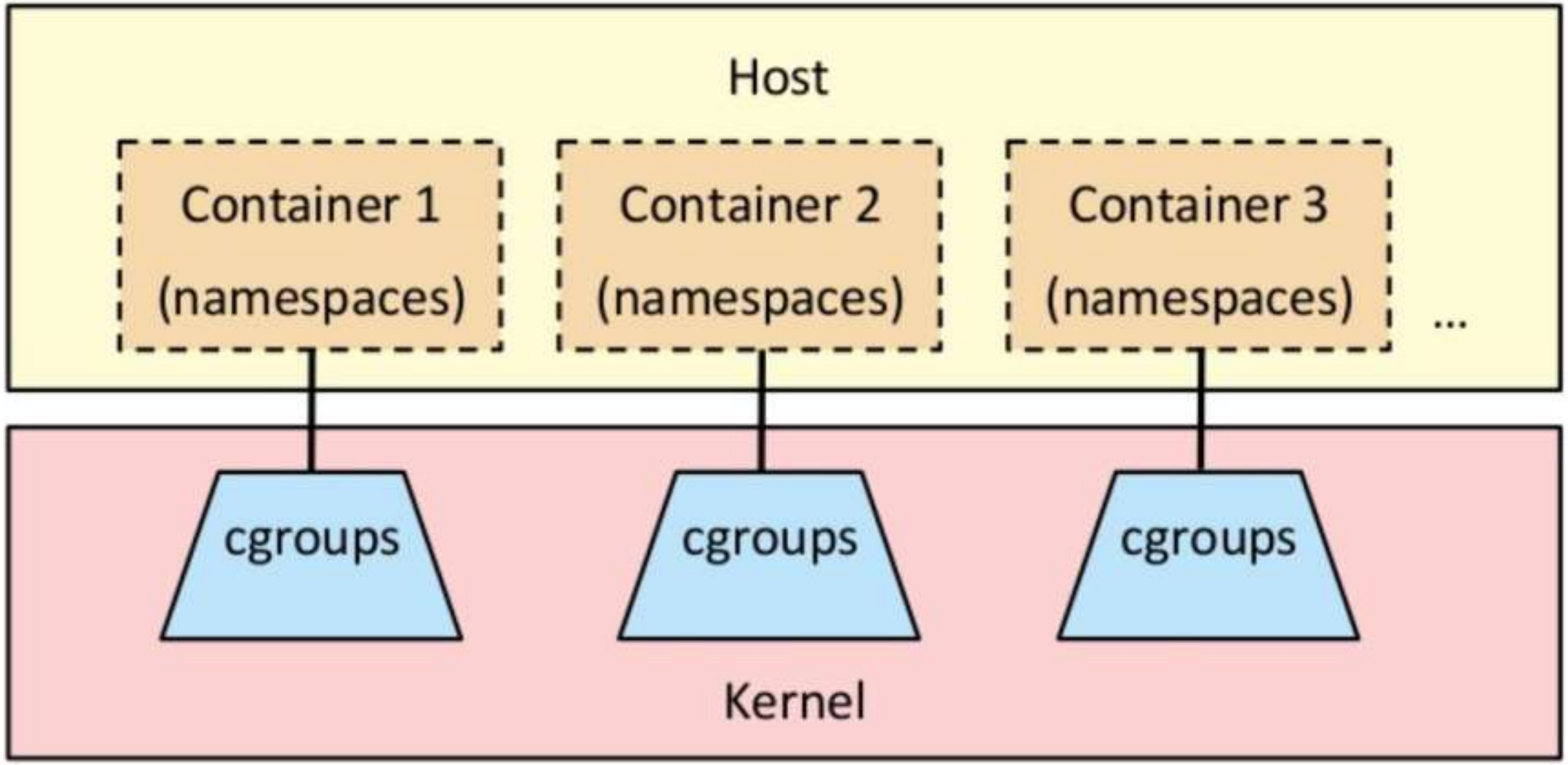
IT WORKS ON MY MACHINE
THEN WE'LL SHIP YOUR MACHINE
AND THAT IS HOW DOCKER WAS BORN
imgflip.com

# DOCKER VS VM

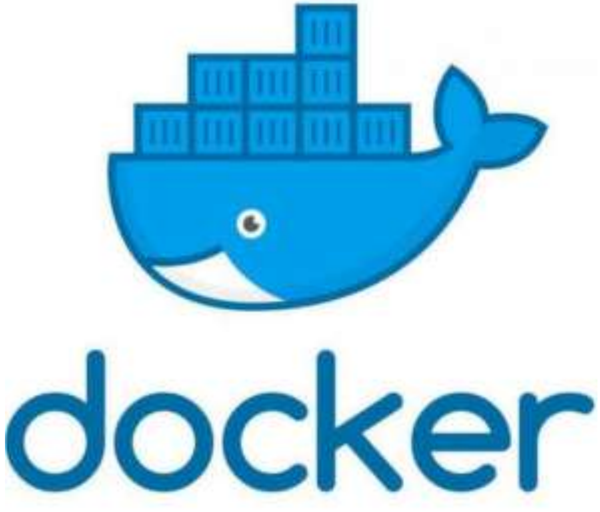

App 1
App 2
App 3
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Host Operating System
Infrastructure
Virtual Machines
App 1
App 2
App 3
Bins/Libs
Bins/Libs
Bins/Libs
Docker Engine
Operating System
Infrastructure
Containers

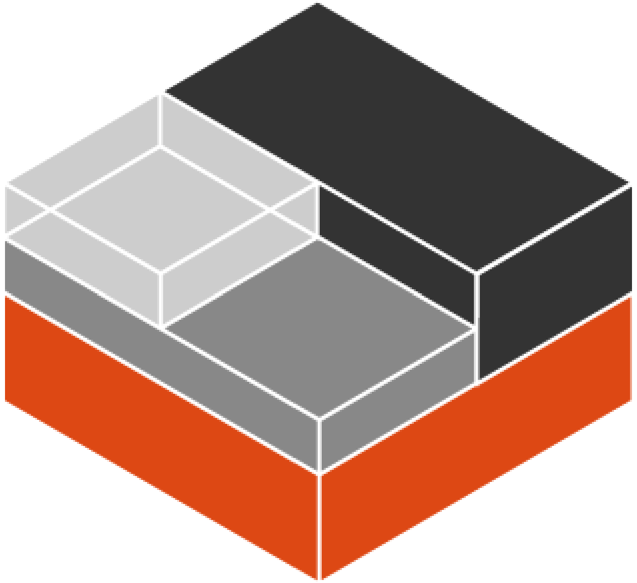
# HOW DOES DOCKER WORK?

Containers are not a first class citizen of the Linux kernel. They are a combination of namespaces and cgroups.



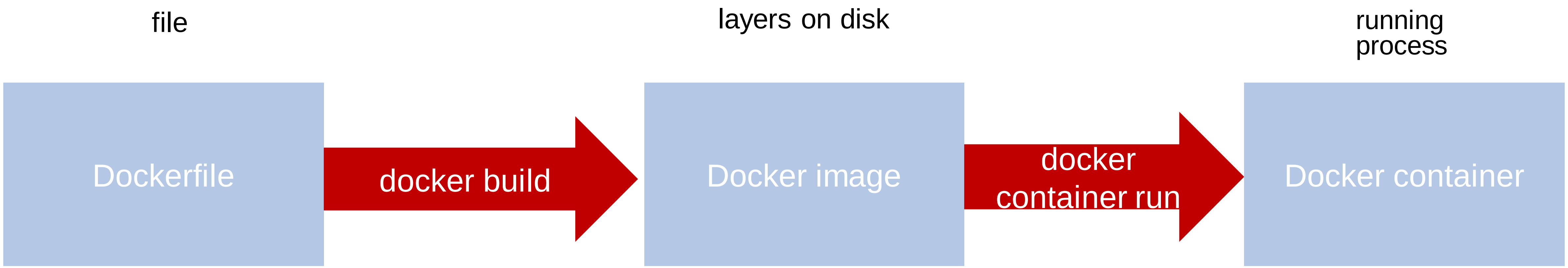
# DOCKER OR ALTERNATIVES?





# VM ALTERNATIVES

# TERMINOLOGY



file

Dockerfile

docker build

layers on disk

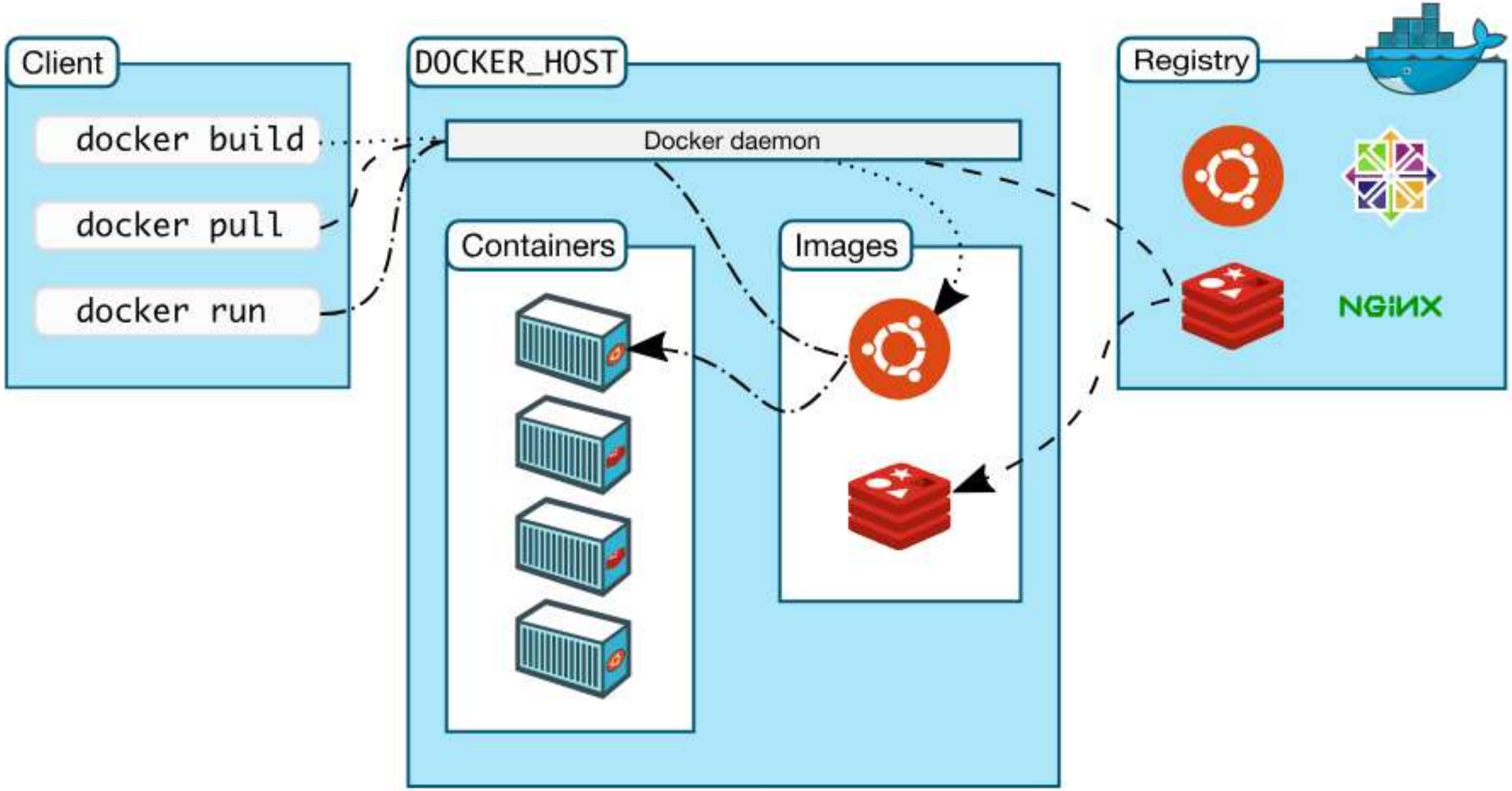
Docker image

docker container run

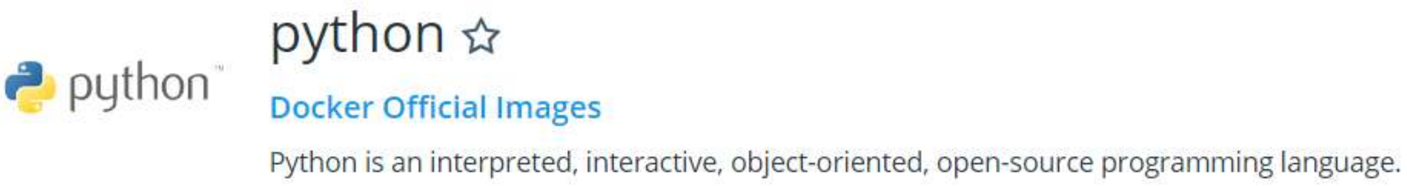
running process

Docker container

# BASIC VOCABULARY



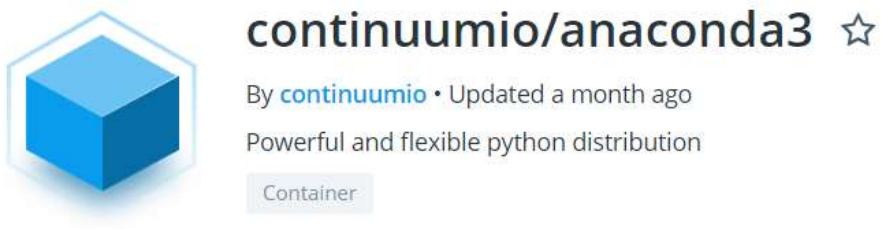
# DOCKER IMAGES FOR PYTHON



python ☆

Docker Official Images

Python is an interpreted, interactive, object-oriented, open-source programming language.

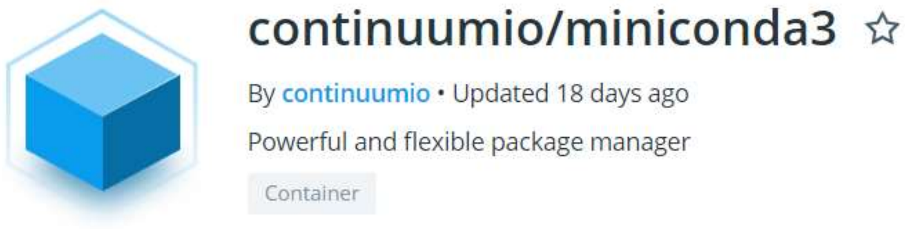


continuumio/anaconda3 ☆

By continuumio • Updated a month ago

Powerful and flexible python distribution

Container



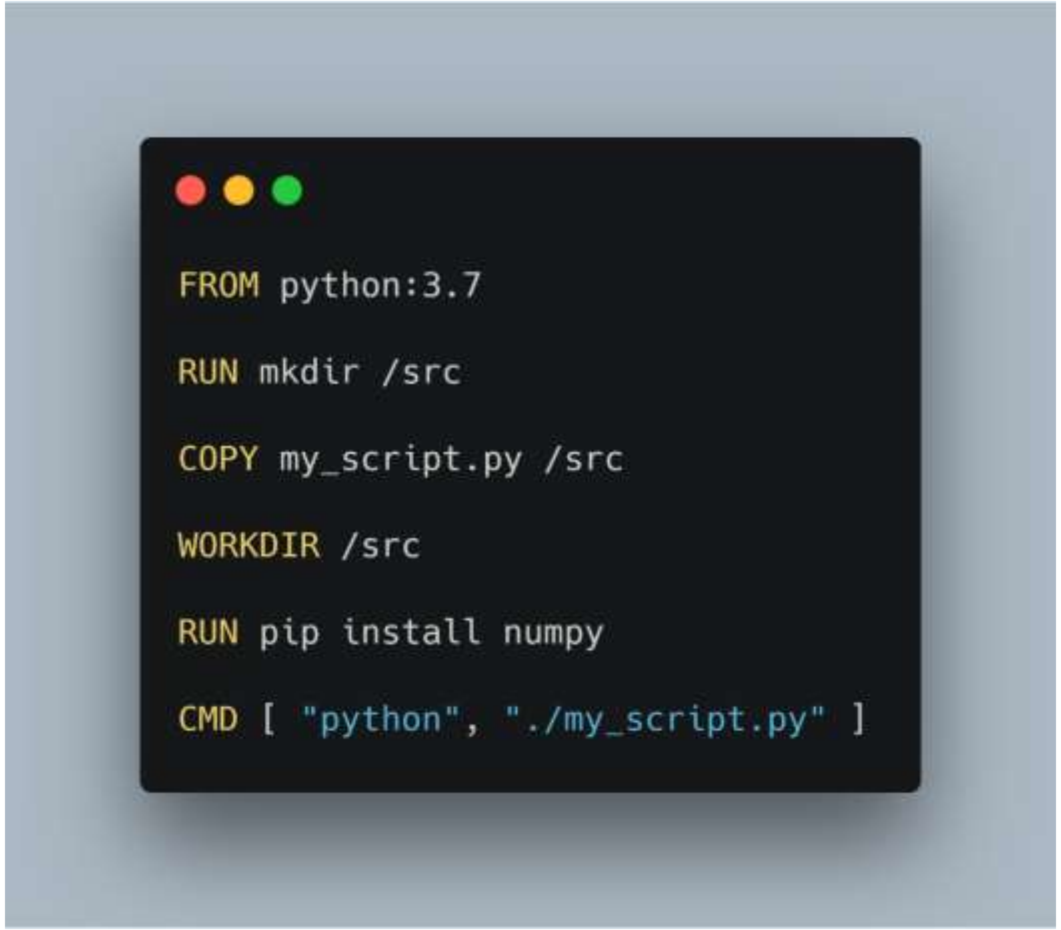
continuumio/miniconda3 ☆

By continuumio • Updated 18 days ago

Powerful and flexible package manager

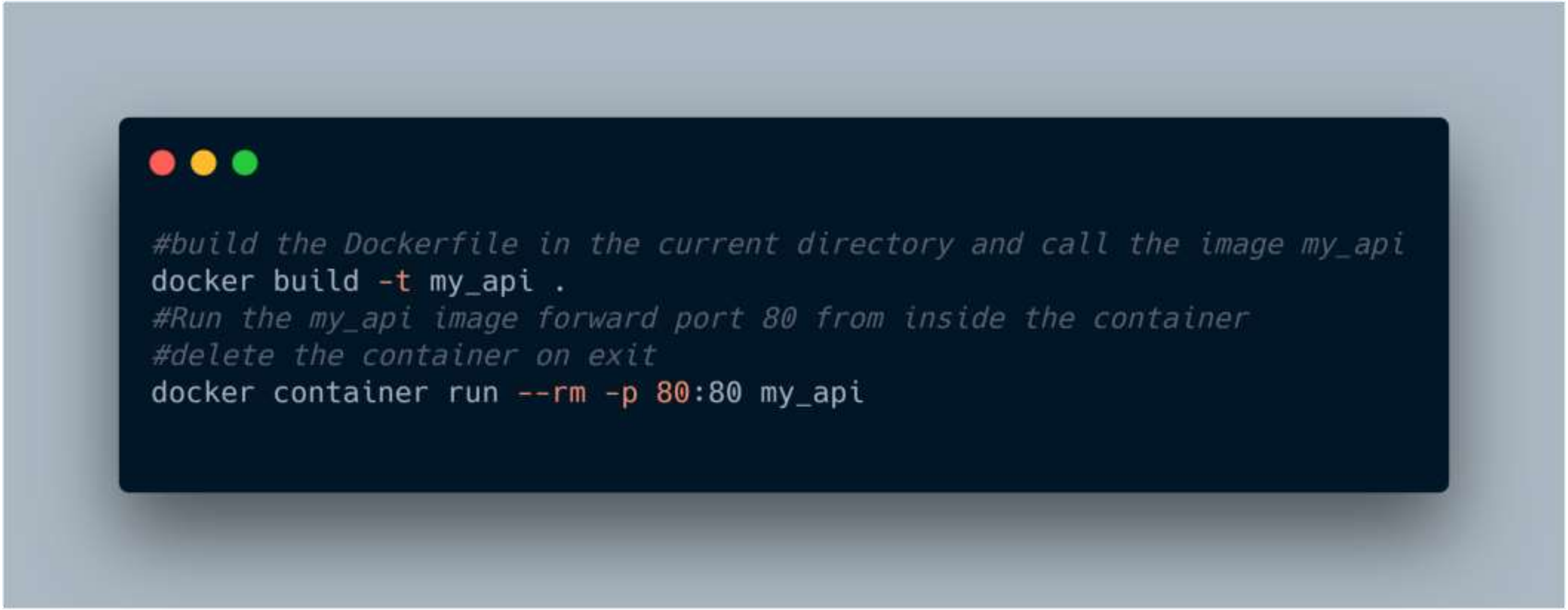
Container

# HOW TO BUILD YOUR OWN IMAGE?



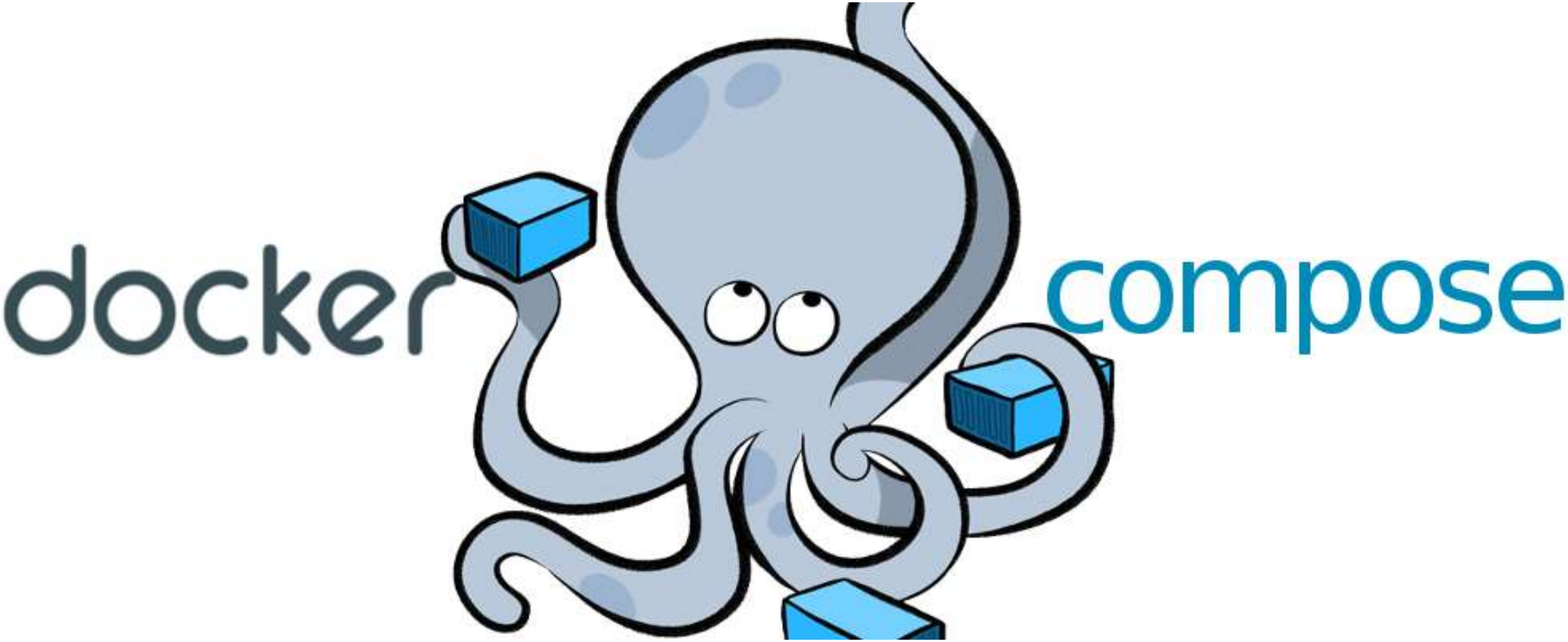
```
FROM python:3.7

RUN mkdir /src

COPY my_script.py /src

WORKDIR /src

RUN pip install numpy

CMD [ "python", "./my_script.py" ]
```

# DEMO SINGLE CONTAINER



```
#build the Dockerfile in the current directory and call the image my_api
docker build -t my_api .
#Run the my_api image forward port 80 from inside the container
#delete the container on exit
docker container run --rm -p 80:80 my_api
```

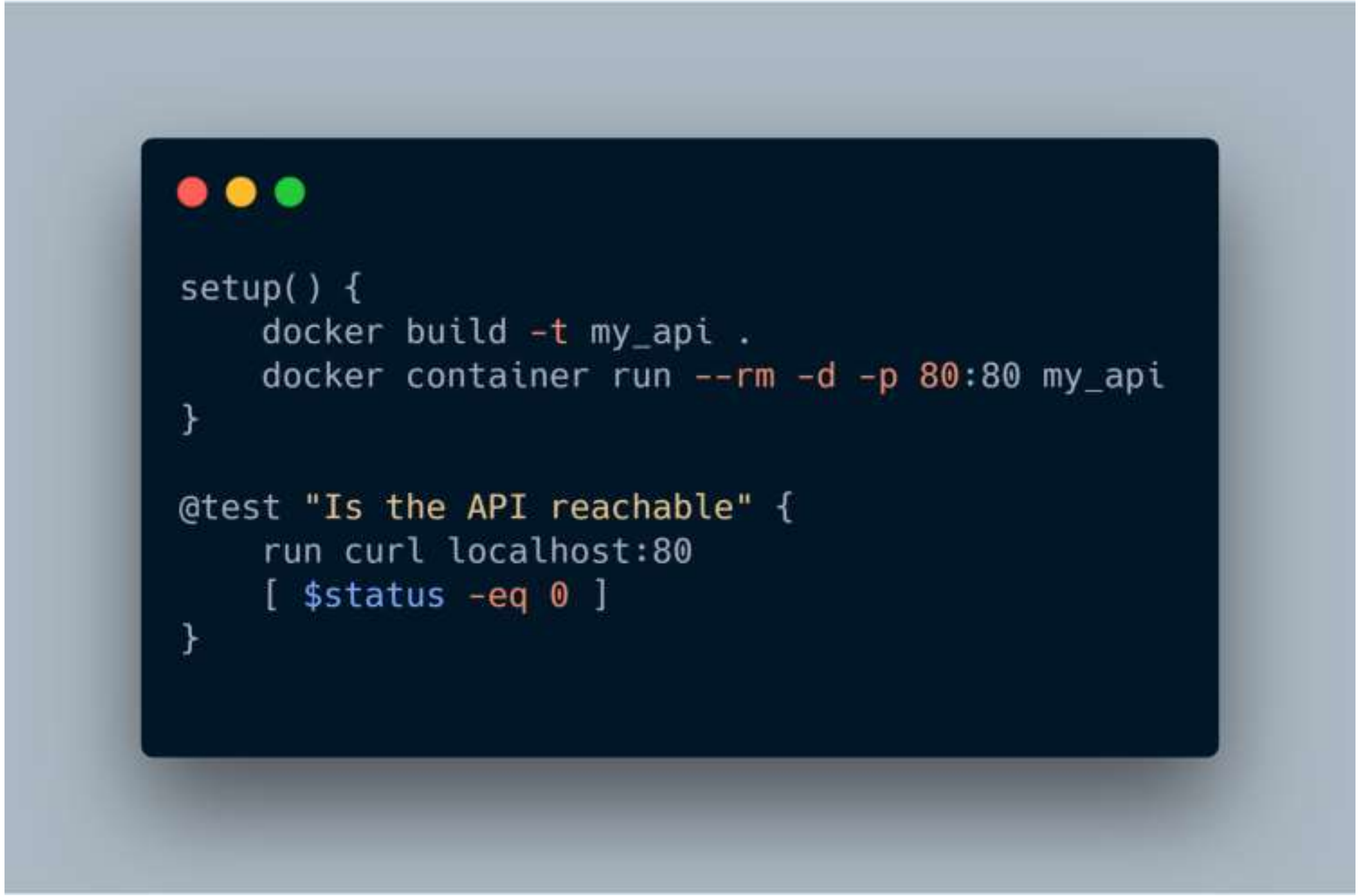
# DEMO MULTIPLE CONTAINERS



# TESTING WITH DOCKER

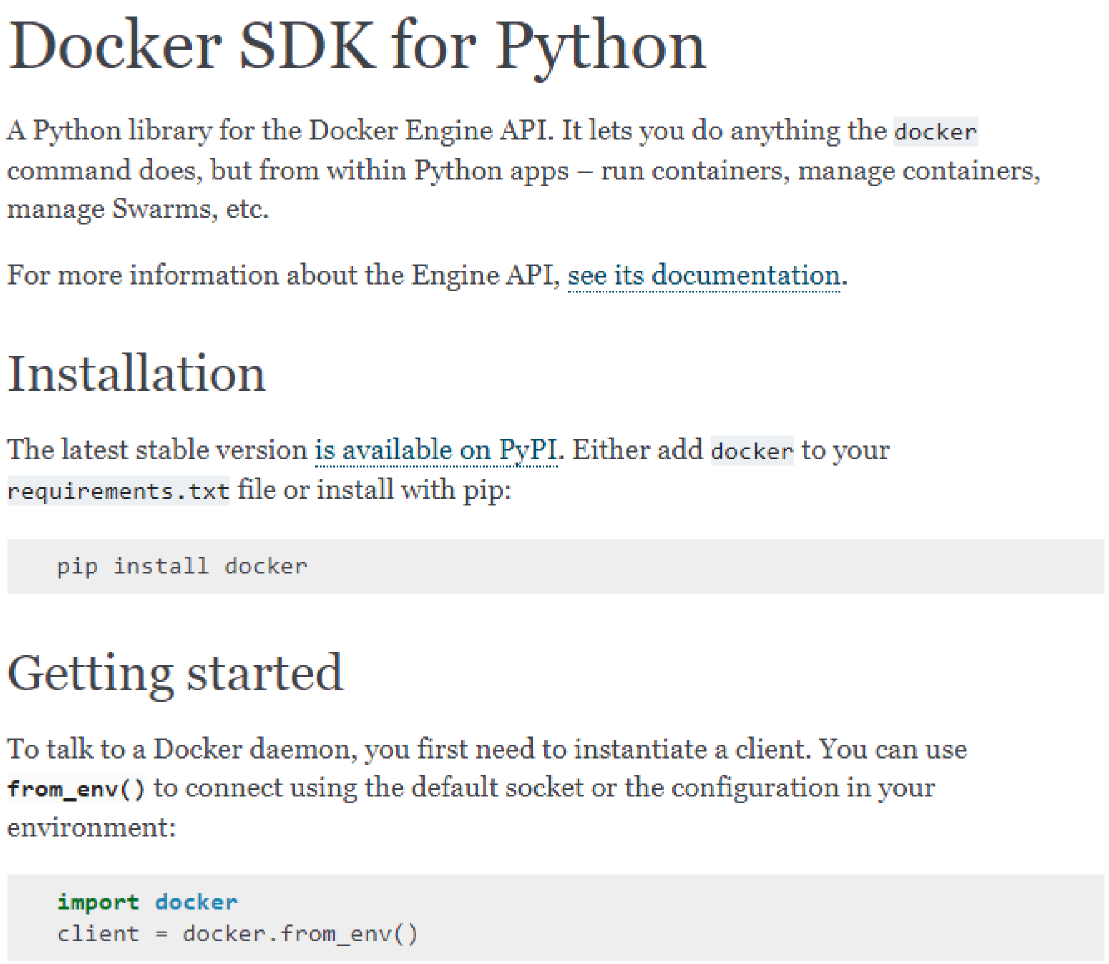
- Code inside of container needs testing
- Also in need of tests:
  - Dockerfile
  - Configuration
  - Entry point scripts of containers

# DEMO - BASH AUTOMATED TESTING SYSTEM (BATS)



```
setup() {
    docker build -t my_api .
    docker container run --rm -d -p 80:80 my_api
}

@test "Is the API reachable" {
    run curl localhost:80
    [ $status -eq 0 ]
}
```

# DEMO - TESTING WITH DOCKER SDK FOR PYTHON



# Docker SDK for Python

A Python library for the Docker Engine API. It lets you do anything the `docker` command does, but from within Python apps – run containers, manage containers, manage Swarms, etc.

For more information about the Engine API, see its documentation.

## Installation

The latest stable version is available on PyPI. Either add `docker` to your `requirements.txt` file or install with pip:

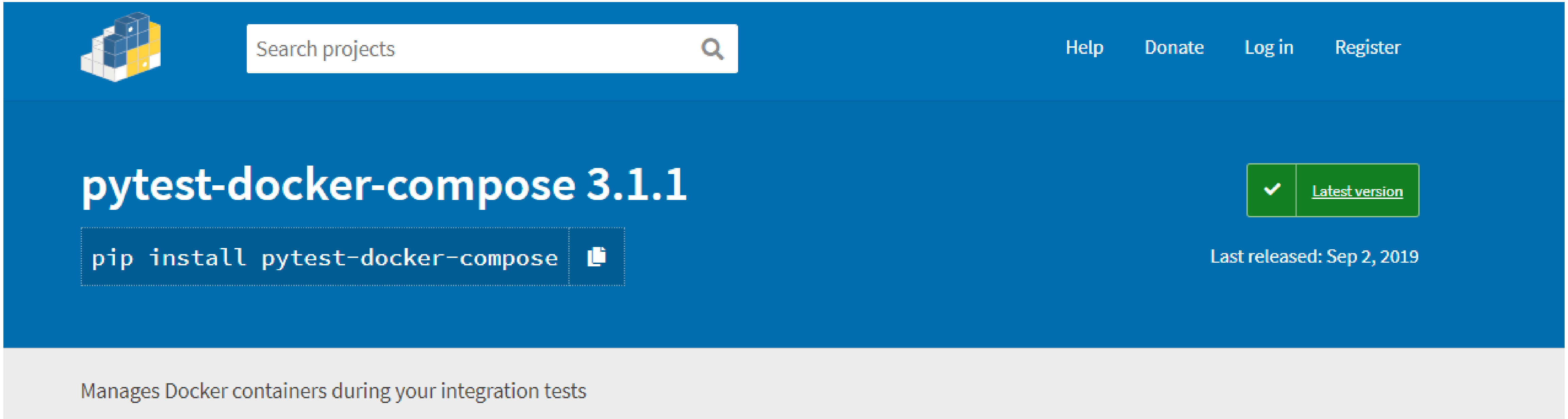
```
pip install docker
```

## Getting started

To talk to a Docker daemon, you first need to instantiate a client. You can use **`from_env()`** to connect using the default socket or the configuration in your environment:

```
import docker
client = docker.from_env()
```

# DEMO - PYTEST-COMPOSE



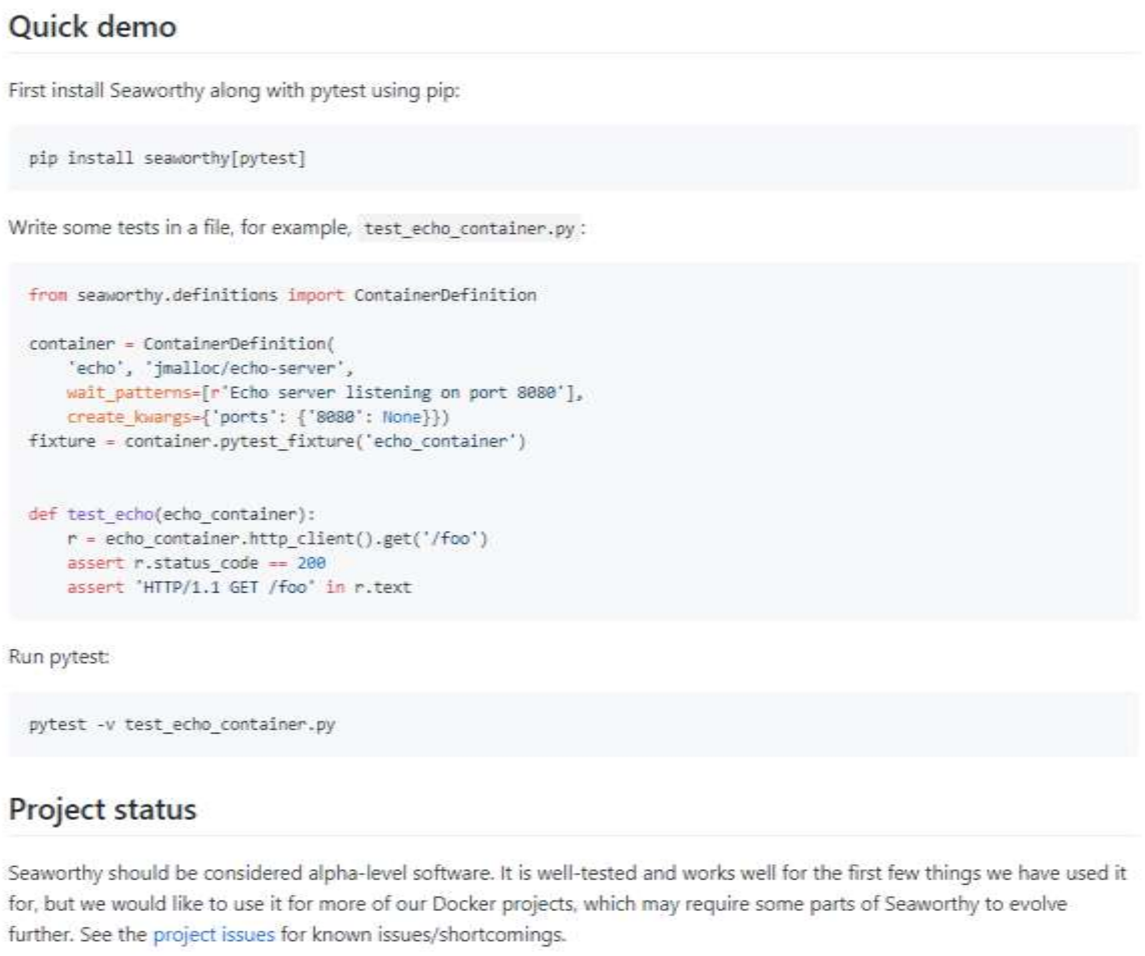
pytest-docker-compose 3.1.1

```
pip install pytest-docker-compose
```

Last released: Sep 2, 2019

Manages Docker containers during your integration tests

# SEAWORTHY



## Quick demo

First install Seaworthy along with pytest using pip:

```
pip install seaworthy[pytest]
```

Write some tests in a file, for example, `test_echo_container.py`:

```
from seaworthy.definitions import ContainerDefinition

container = ContainerDefinition(
    'echo', 'jmalloc/echo-server',
    wait_patterns=[r'Echo server listening on port 8080'],
    create_kwargs={'ports': {'8080': None}})
fixture = container.pytest_fixture('echo_container')


def test_echo(echo_container):
    r = echo_container.http_client().get('/foo')
    assert r.status_code == 200
    assert 'HTTP/1.1 GET /foo' in r.text
```

Run pytest:

```
pytest -v test_echo_container.py
```

## Project status

Seaworthy should be considered alpha-level software. It is well-tested and works well for the first few things we have used it for, but we would like to use it for more of our Docker projects, which may require some parts of Seaworthy to evolve further. See the project issues for known issues/shortcomings.

# USE CASES FOR DOCKER AND PYTHON

- Deployment of web applications, (REST) APIs, …
- Especially with other dependencies like databases
- Distribution of scientific workloads, training of machine learning models
- Continuous Integration
- Reproducible environments across different machines

**DO NOT:**

- Use Docker for security reasons (you want jails or sandboxes or VMs)
- Use Docker as an excuse for a messy environment

# THANK YOU FOR JOINING THIS PRESENTATION.

ROSEN

empowered by technology