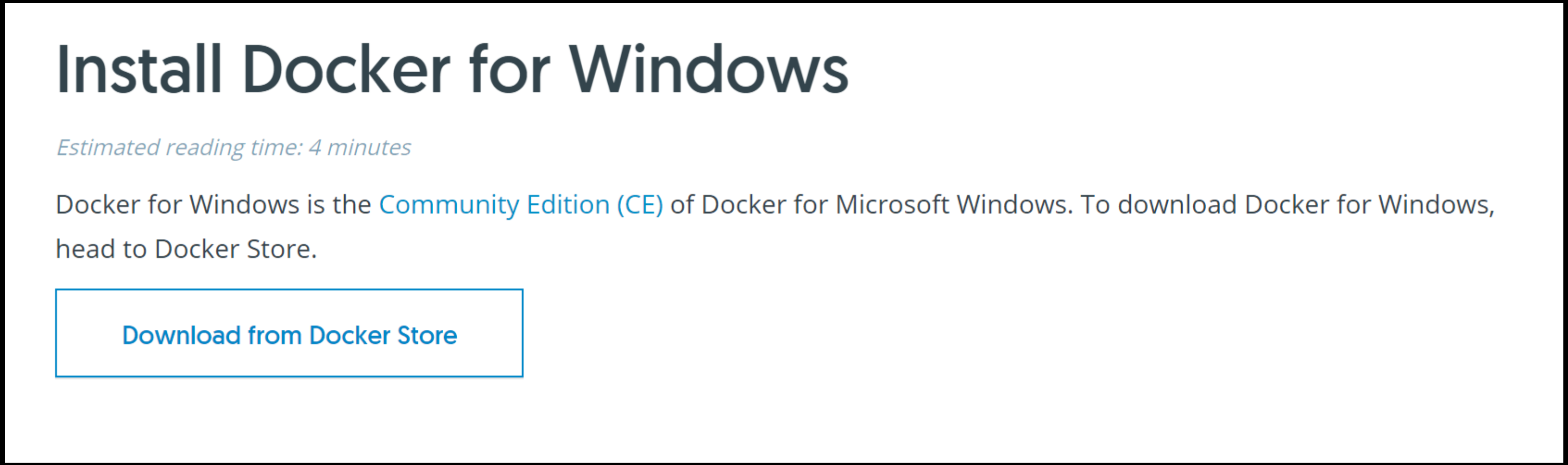
How to use Docker as a Data Analyst

Download your work environment

Download and install Docker



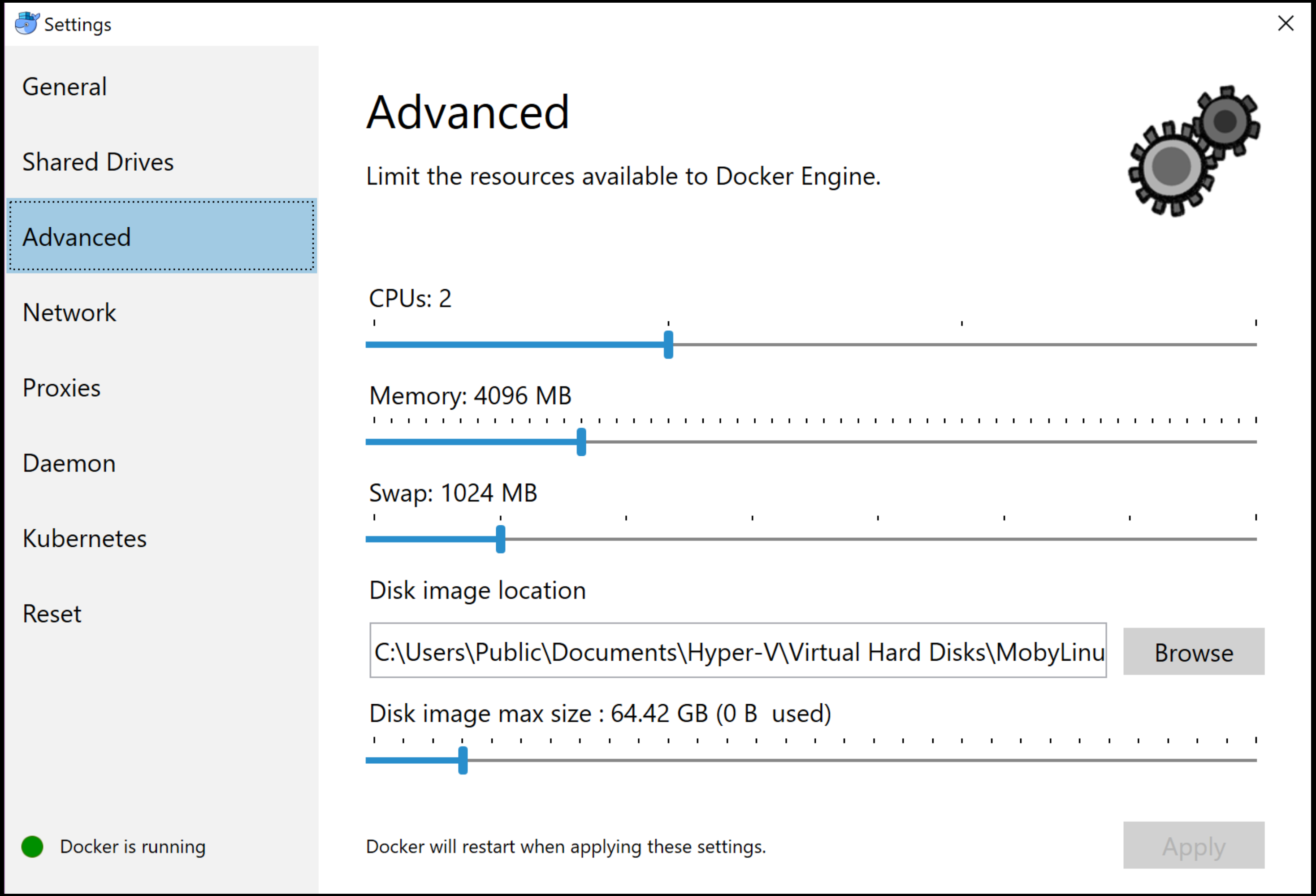
Install Docker for Windows

Estimated reading time: 4 minutes

Docker for Windows is the Community Edition (CE) of Docker for Microsoft Windows. To download Docker for Windows, head to Docker Store.

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

Configure settings


Settings

General
Shared Drives
Advanced
Network
Proxies
Daemon
Kubernetes
Reset

● Docker is running

Advanced

Limit the resources available to Docker Engine.

CPUs: 2

Memory: 4096 MB

Swap: 1024 MB

Disk image location

C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks\MobyLinu

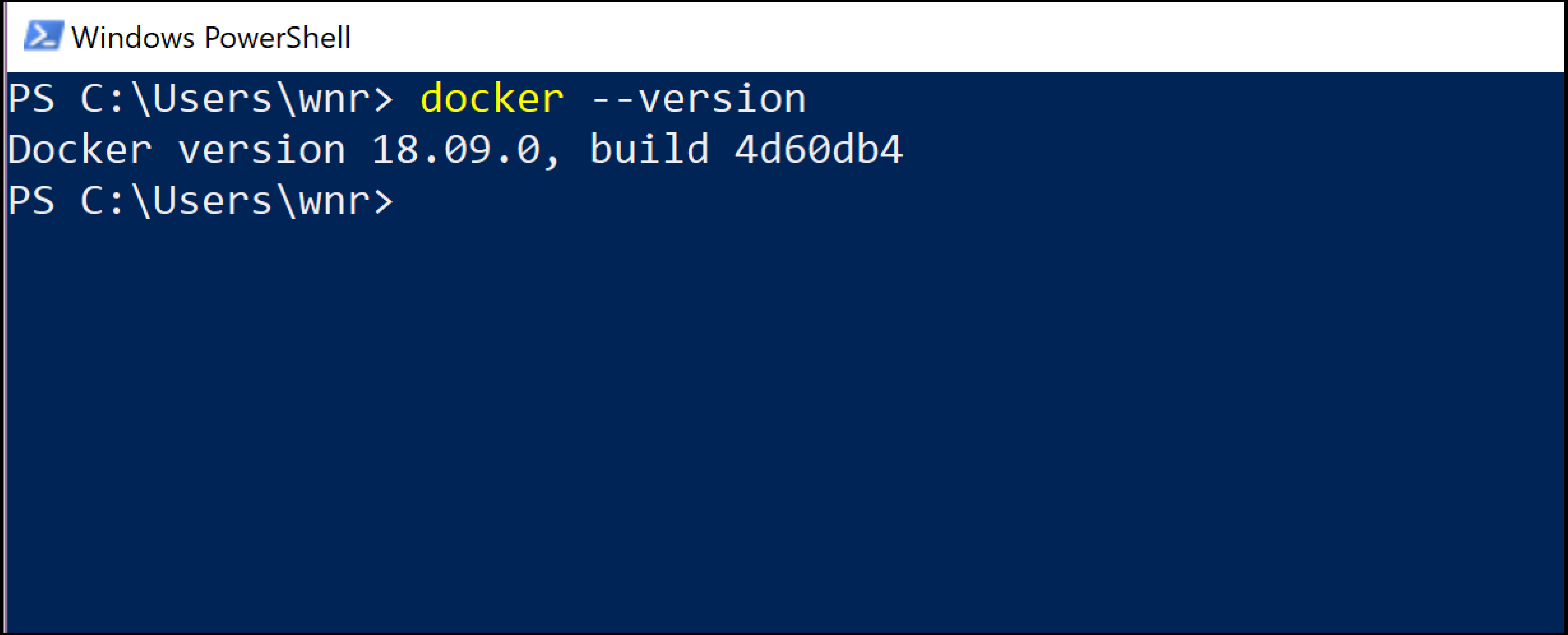
Browse

Disk image max size : 64.42 GB (0 B used)

Docker will restart when applying these settings.

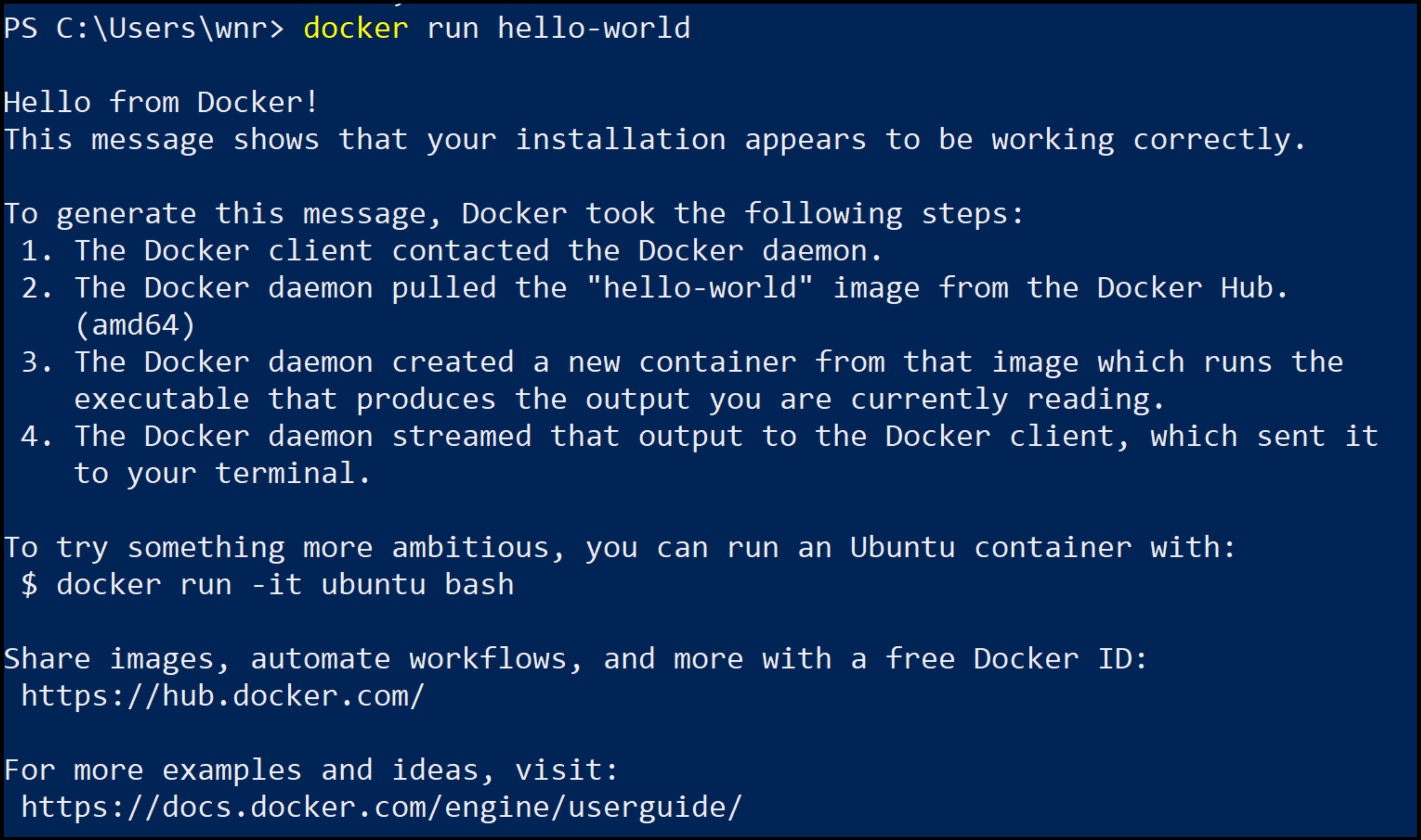
Apply

Open a command (or Power) shell

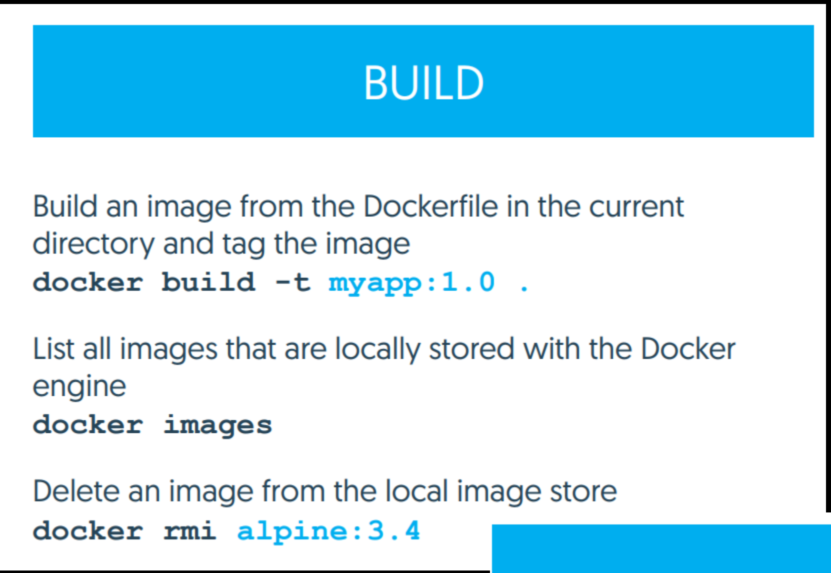


```
PS C:\Users\wnr> docker --version
Docker version 18.09.0, build 4d60db4
PS C:\Users\wnr>
```

Test it



```
PS C:\Users\wnr> docker run hello-world

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/engine/userguide/
```

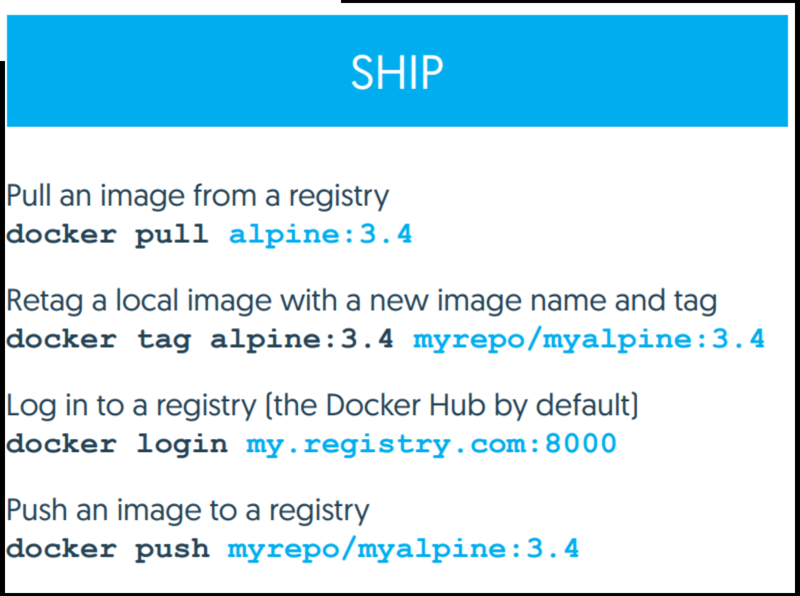


BUILD

Build an image from the Dockerfile in the current directory and tag the image
```
docker build -t myapp:1.0 .
```

List all images that are locally stored with the Docker engine
```
docker images
```

Delete an image from the local image store
```
docker rmi alpine:3.4
```



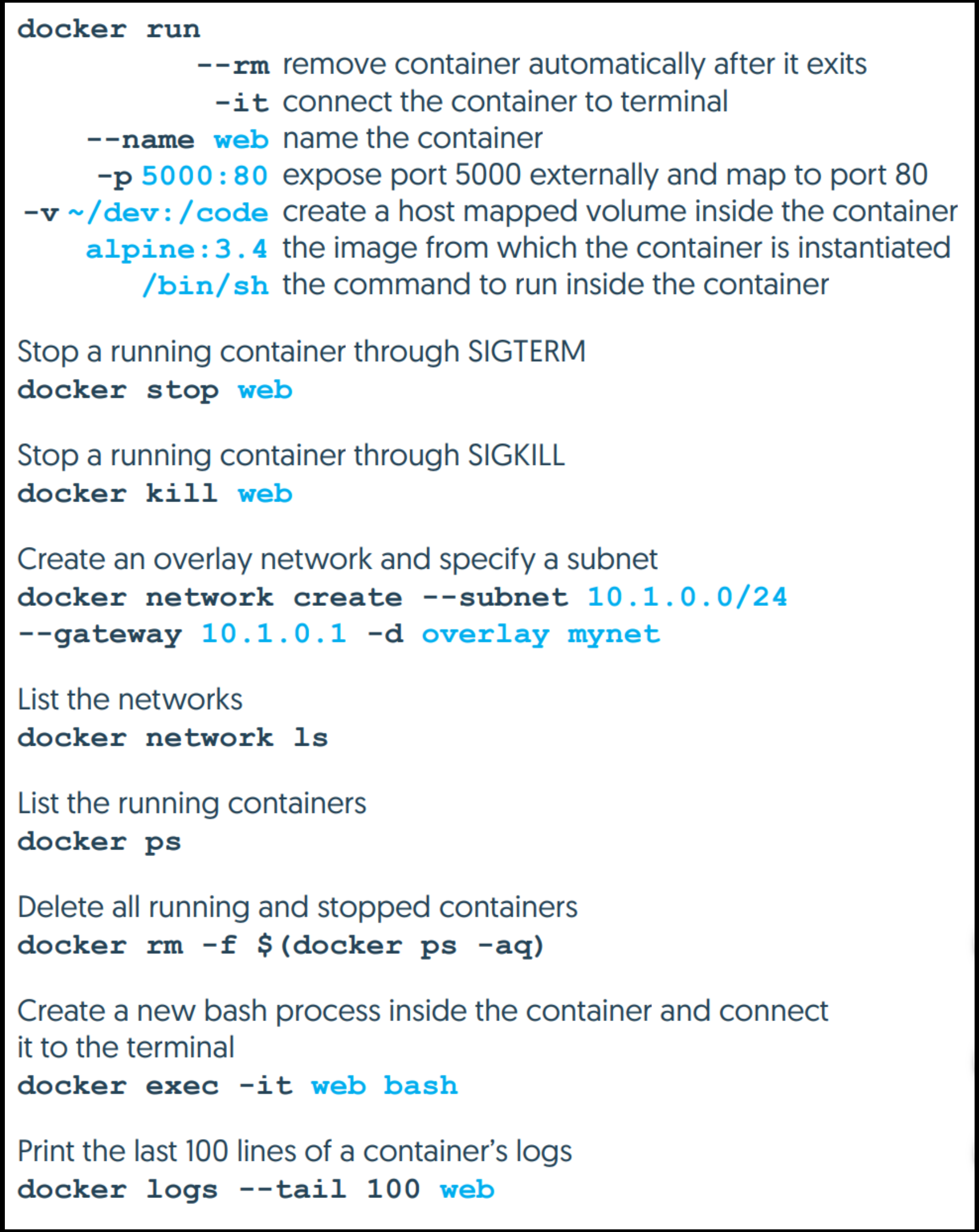
SHIP

Pull an image from a registry
```
docker pull alpine:3.4
```

Retag a local image with a new image name and tag
```
docker tag alpine:3.4 myrepo/myalpine:3.4
```

Log in to a registry (the Docker Hub by default)
```
docker login my.registry.com:8000
```

Push an image to a registry
```
docker push myrepo/myalpine:3.4
```



```
docker run
              --rm remove container automatically after it exits
               -it connect the container to terminal
        --name web name the container
         -p 5000:80 expose port 5000 externally and map to port 80
    -v ~/dev:/code create a host mapped volume inside the container
        alpine:3.4 the image from which the container is instantiated
           /bin/sh the command to run inside the container
```

Stop a running container through SIGTERM
```
docker stop web
```

Stop a running container through SIGKILL
```
docker kill web
```

Create an overlay network and specify a subnet
```
docker network create --subnet 10.1.0.0/24
--gateway 10.1.0.1 -d overlay mynet
```

List the networks
```
docker network ls
```

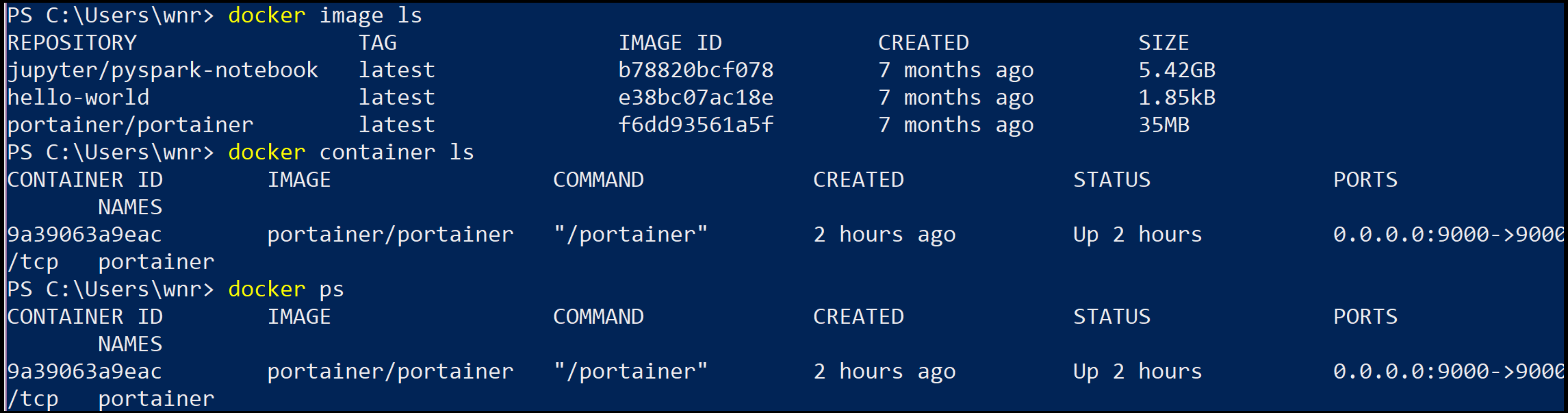
List the running containers
```
docker ps
```

Delete all running and stopped containers
```
docker rm -f $(docker ps -aq)
```

Create a new bash process inside the container and connect it to the terminal
```
docker exec -it web bash
```

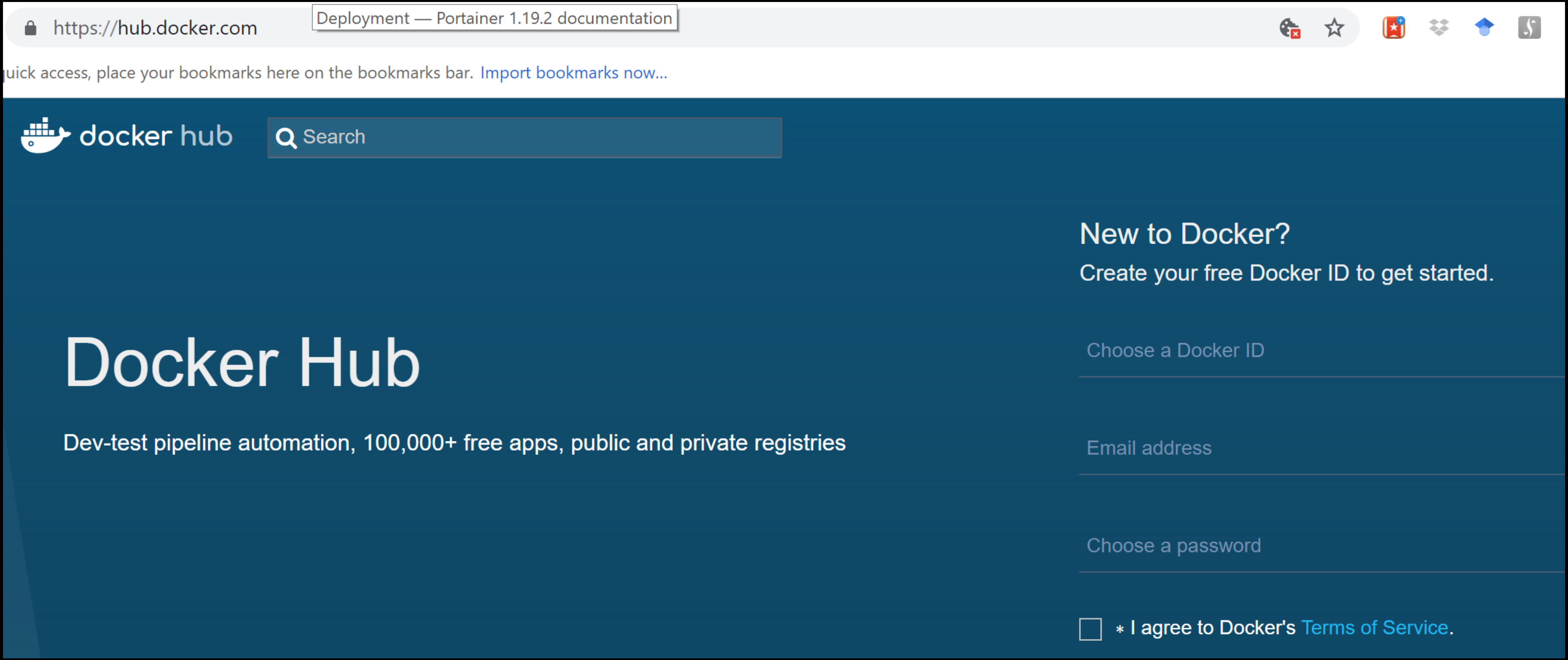
Print the last 100 lines of a container's logs
```
docker logs --tail 100 web
```

List docker images & containers



```
PS C:\Users\wnr> docker image ls
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
jupyter/pyspark-notebook  latest              b78820bcf078        7 months ago        5.42GB
hello-world               latest              e38bc07ac18e        7 months ago        1.85kB
portainer/portainer       latest              f6dd93561a5f        7 months ago        35MB
PS C:\Users\wnr> docker container ls
CONTAINER ID        IMAGE                 COMMAND             CREATED             STATUS              PORTS
       NAMES
9a39063a9eac        portainer/portainer   "/portainer"        2 hours ago         Up 2 hours          0.0.0.0:9000->900
/tcp   portainer
PS C:\Users\wnr> docker ps
CONTAINER ID        IMAGE                 COMMAND             CREATED             STATUS              PORTS
       NAMES
9a39063a9eac        portainer/portainer   "/portainer"        2 hours ago         Up 2 hours          0.0.0.0:9000->900
/tcp   portainer
```

Get a container



https://hub.docker.com

Deployment — Portainer 1.19.2 documentation

uick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

docker hub

Search

Docker Hub

Dev-test pipeline automation, 100,000+ free apps, public and private registries

New to Docker?

Create your free Docker ID to get started.

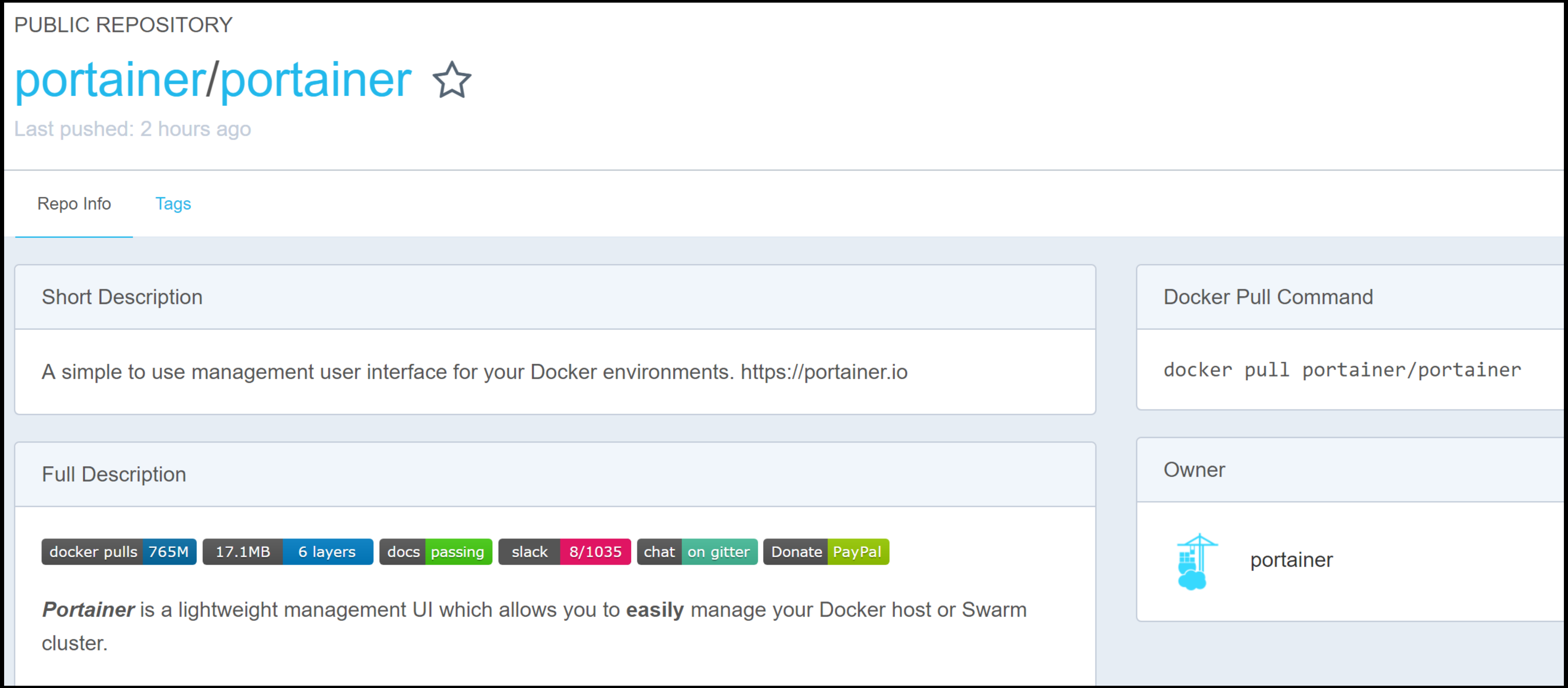
Choose a Docker ID

Email address

Choose a password

☐ * I agree to Docker's Terms of Service.

Pull portainer



PUBLIC REPOSITORY

portainer/portainer ☆

Last pushed: 2 hours ago

Repo Info Tags

Short Description

A simple to use management user interface for your Docker environments. https://portainer.io

Docker Pull Command

```
docker pull portainer/portainer
```

Full Description

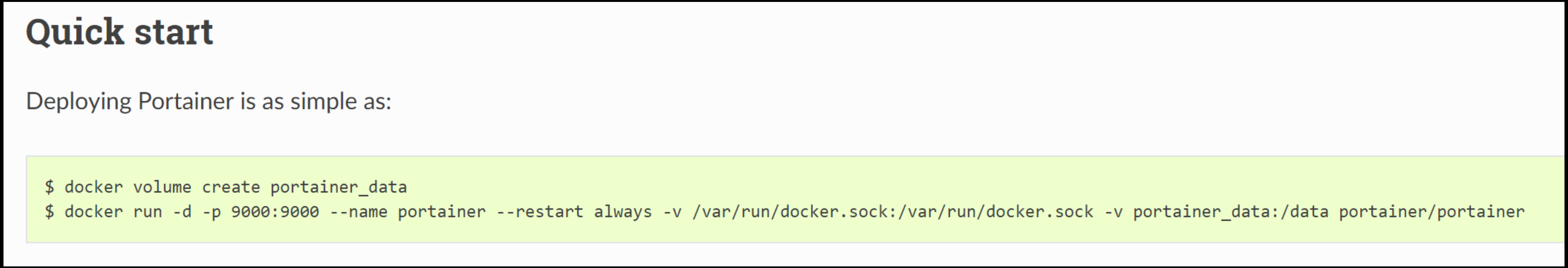
docker pulls 765M | 17.1MB 6 layers | docs passing | slack 8/1035 | chat on gitter | Donate PayPal

Portainer is a lightweight management UI which allows you to **easily** manage your Docker host or Swarm cluster.

Owner

portainer

Run portainer

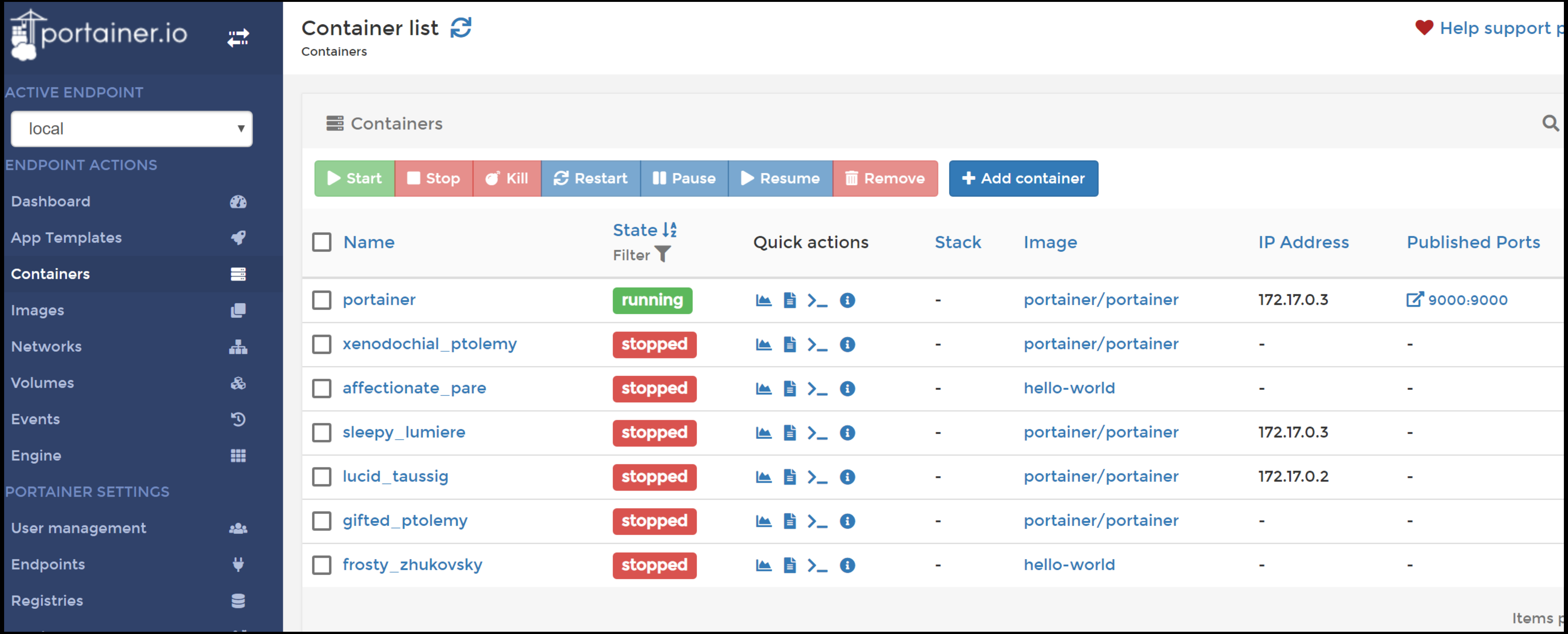


Quick start

Deploying Portainer is as simple as:

```
$ docker volume create portainer_data
$ docker run -d -p 9000:9000 --name portainer --restart always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer
```

View your containers in portainer



Container list
Containers

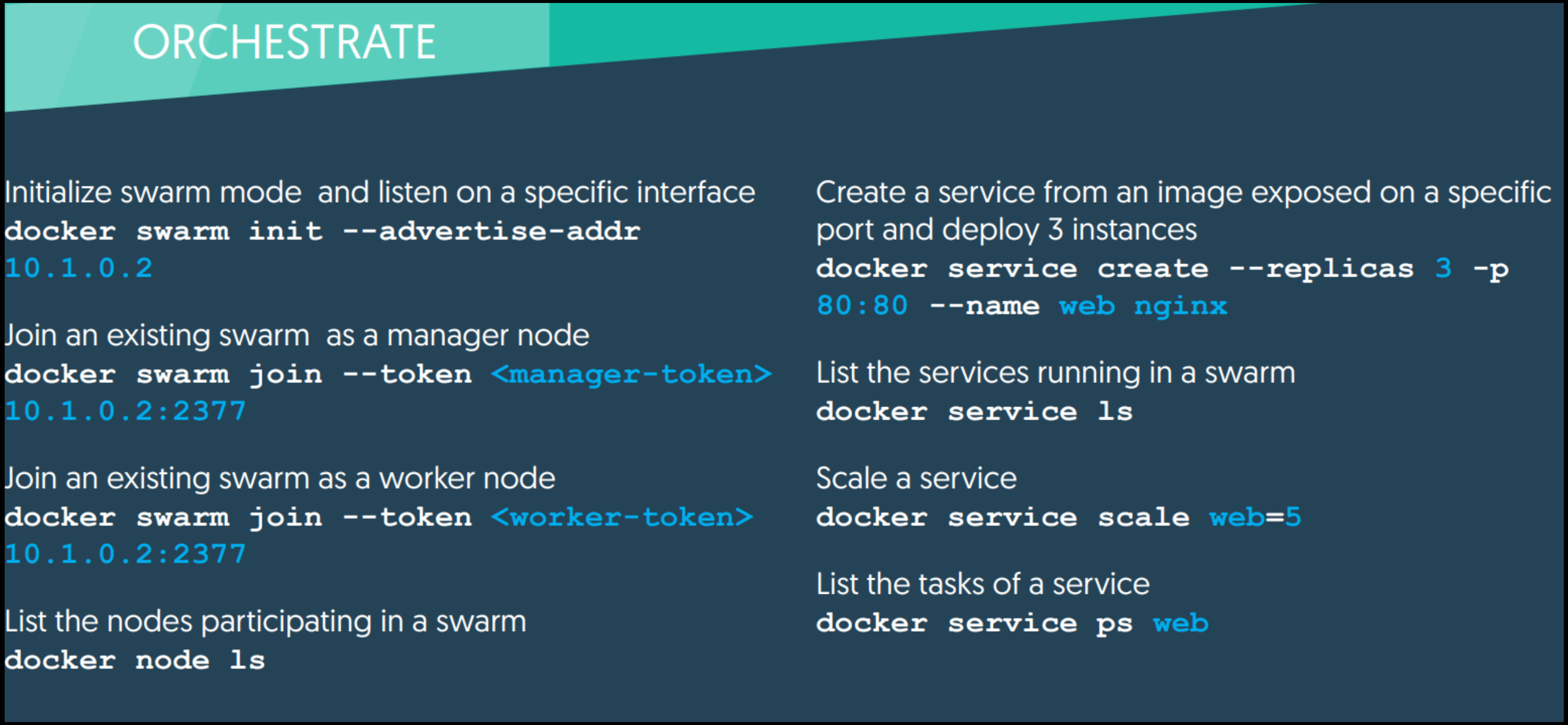
Containers

Name	State Filter	Quick actions	Stack	Image	IP Address	Published Ports
portainer	running		-	portainer/portainer	172.17.0.3	9000:9000
xenodochial_ptolemy	stopped		-	portainer/portainer	-	-
affectionate_pare	stopped		-	hello-world	-	-
sleepy_lumiere	stopped		-	portainer/portainer	172.17.0.3	-
lucid_taussig	stopped		-	portainer/portainer	172.17.0.2	-
gifted_ptolemy	stopped		-	portainer/portainer	-	-
frosty_zhukovsky	stopped		-	hello-world	-	-

After you run, delete the container

- Image defines the program
- Container is dynamically created to run program
  - When done, you can delete the container

Eventually run multiple containers



ORCHESTRATE

Initialize swarm mode and listen on a specific interface

```
docker swarm init --advertise-addr
10.1.0.2
```

Join an existing swarm as a manager node

```
docker swarm join --token <manager-token>
10.1.0.2:2377
```

Join an existing swarm as a worker node

```
docker swarm join --token <worker-token>
10.1.0.2:2377
```

List the nodes participating in a swarm

```
docker node ls
```

Create a service from an image exposed on a specific port and deploy 3 instances

```
docker service create --replicas 3 -p
80:80 --name web nginx
```

List the services running in a swarm

```
docker service ls
```

Scale a service

```
docker service scale web=5
```

List the tasks of a service

```
docker service ps web
```