Docker intro

What did you install?

```
$ docker version
```

Concepts and commands

```
$ docker --help | less
```

{client, engine, registry, image, container, volume, service, network, swarm, node, secret, …}

{images, build, push, pull, run, exec, inspect, ps, stop, rm, rmi, …}

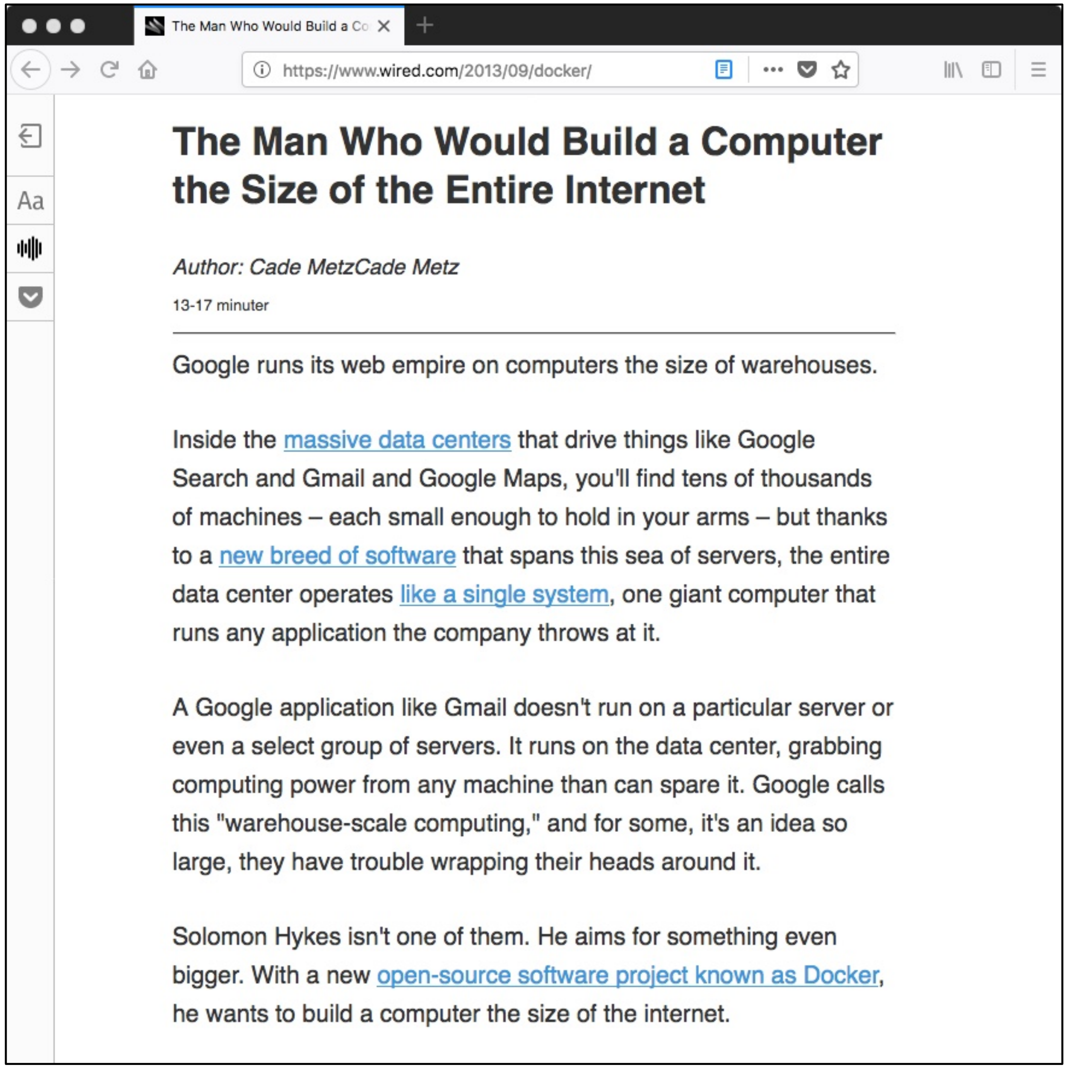
What’s the purpose of Docker containers?

To run portable applications.

Anywhere.

Portable applications

Old news


The Man Who Would Build a Co
https://www.wired.com/2013/09/docker/
Aa

The Man Who Would Build a Computer the Size of the Entire Internet

Author: Cade MetzCade Metz

13-17 minuter

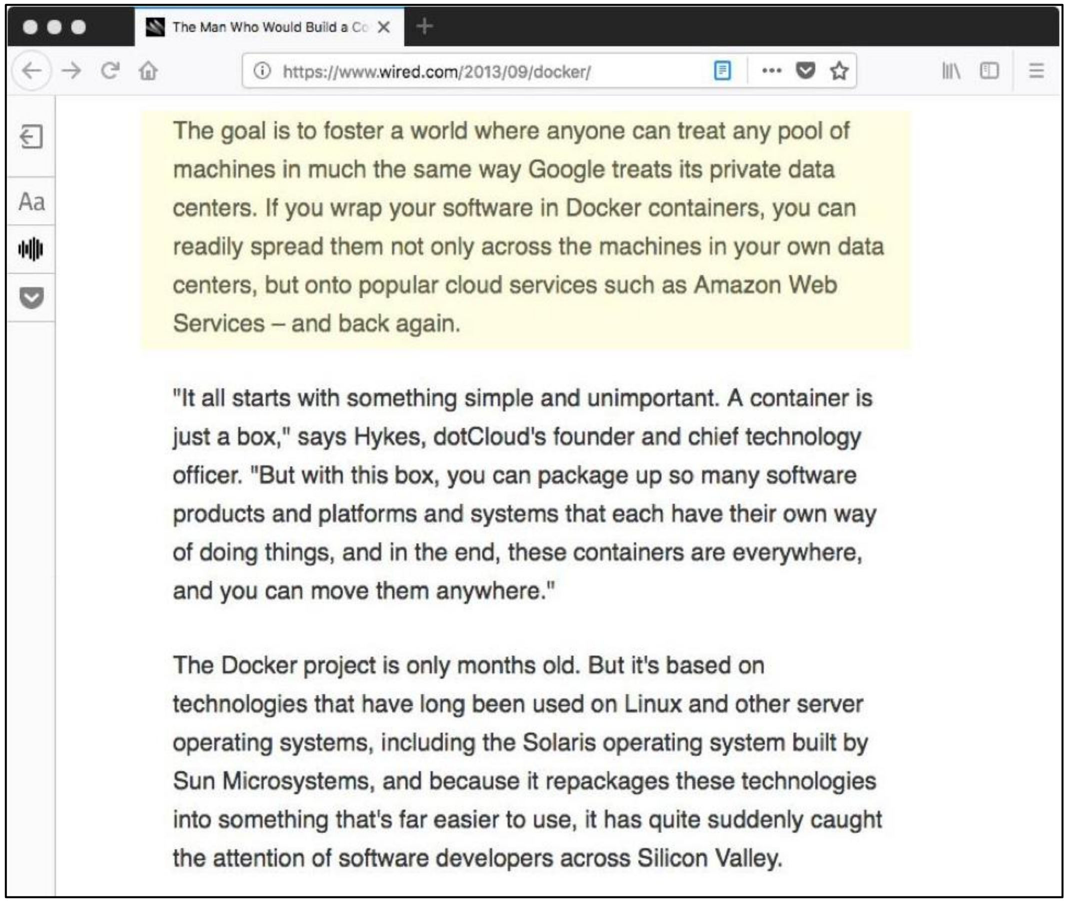
Google runs its web empire on computers the size of warehouses.

Inside the massive data centers that drive things like Google Search and Gmail and Google Maps, you'll find tens of thousands of machines – each small enough to hold in your arms – but thanks to a new breed of software that spans this sea of servers, the entire data center operates like a single system, one giant computer that runs any application the company throws at it.

A Google application like Gmail doesn't run on a particular server or even a select group of servers. It runs on the data center, grabbing computing power from any machine than can spare it. Google calls this "warehouse-scale computing," and for some, it's an idea so large, they have trouble wrapping their heads around it.

Solomon Hykes isn't one of them. He aims for something even bigger. With a new open-source software project known as Docker, he wants to build a computer the size of the internet.

• • •


The Man Who Would Build a Co
https://www.wired.com/2013/09/docker/
Aa

The goal is to foster a world where anyone can treat any pool of machines in much the same way Google treats its private data centers. If you wrap your software in Docker containers, you can readily spread them not only across the machines in your own data centers, but onto popular cloud services such as Amazon Web Services – and back again.

"It all starts with something simple and unimportant. A container is just a box," says Hykes, dotCloud's founder and chief technology officer. "But with this box, you can package up so many software products and platforms and systems that each have their own way of doing things, and in the end, these containers are everywhere, and you can move them anywhere."

The Docker project is only months old. But it's based on technologies that have long been used on Linux and other server operating systems, including the Solaris operating system built by Sun Microsystems, and because it repackages these technologies into something that's far easier to use, it has quite suddenly caught the attention of software developers across Silicon Valley.

What are Docker containers?

Linux applications,

running as Linux processes,
sharing a Linux kernel,
being constrained by namespaces and cgroups,
launched from union(-mounted) file systems,
that also hosts any and all dependencies,

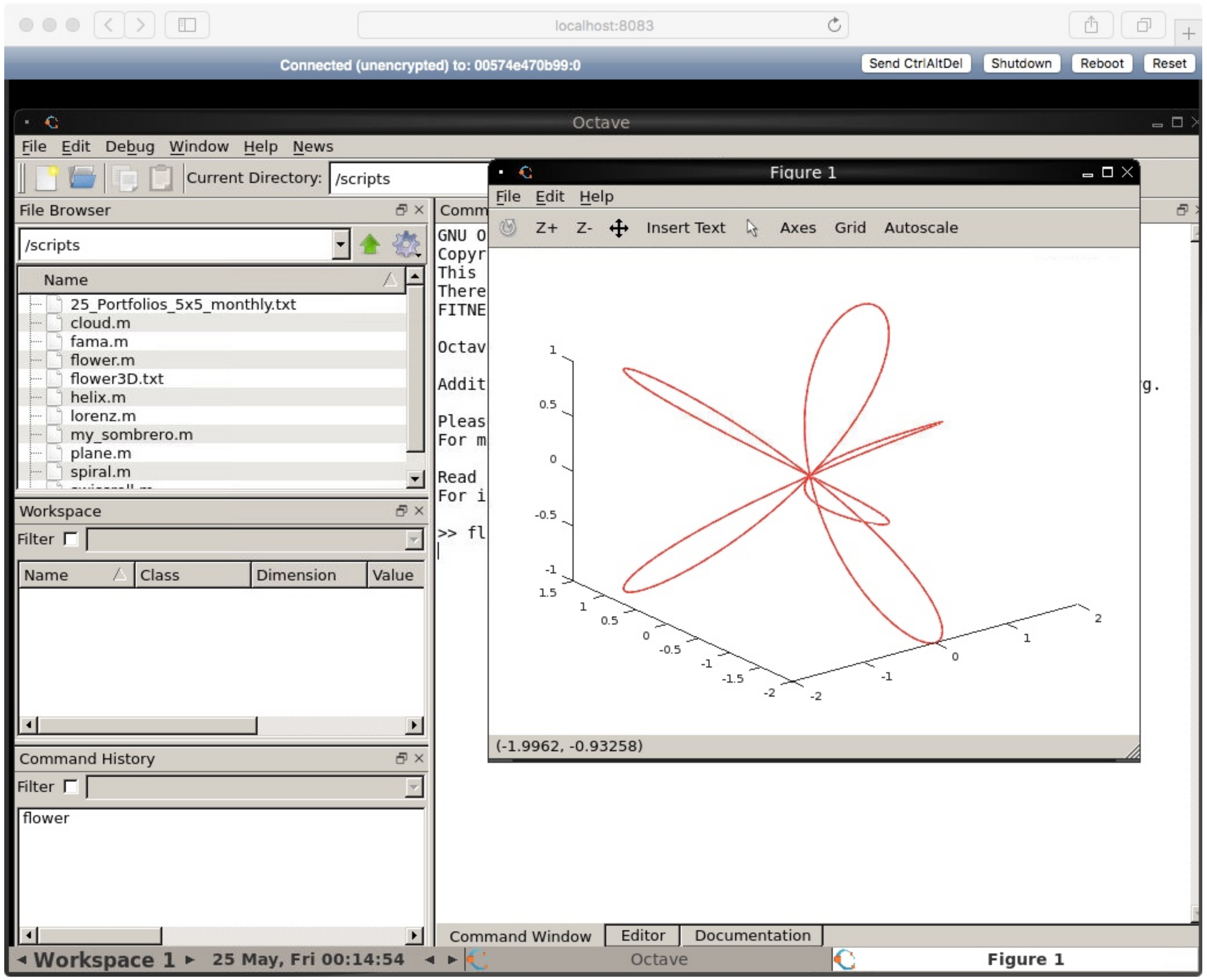
plus some more stuff …

Processes, from mounted file systems hosting dependencies

First contact

```
$ docker run -p 8083:8083 -it epflsti/octave-x11-novnc-
docker:latest


$ open http://localhost:8083
```

Portable applications


localhost:8083
Connected (unencrypted) to: 00574e470b99:0
Send CtrlAltDel
Shutdown
Reboot
Reset
Octave
File Edit Debug Window Help News
Current Directory: /scripts
File Browser
/scripts
Name
Workspace
Filter
Name Class Dimension Value
Command History
Filter
Figure 1
File Edit Help
Z+ Z- Insert Text Axes Grid Autoscale
Command Window Editor Documentation
Workspace 1
25 May, Fri 00:14:54
Octave
Figure 1

25_Portfolios_5x5_monthly.txt
cloud.m
fama.m
flower.m
flower3D.txt
helix.m
lorenz.m
my_sombrero.m
plane.m
spiral.m

flower

```
GNU O
Copyr
This
There
FITNE

Octav

Addit

Pleas
For m

Read
For i

>> fl
```

(-1.9962, -0.93258)

Baby steps

```
$ open https://hub.docker.com/r/rogerlarsson/git

$ docker pull rogerlarsson/git

$ docker run rogerlarsson/git echo hello world
```

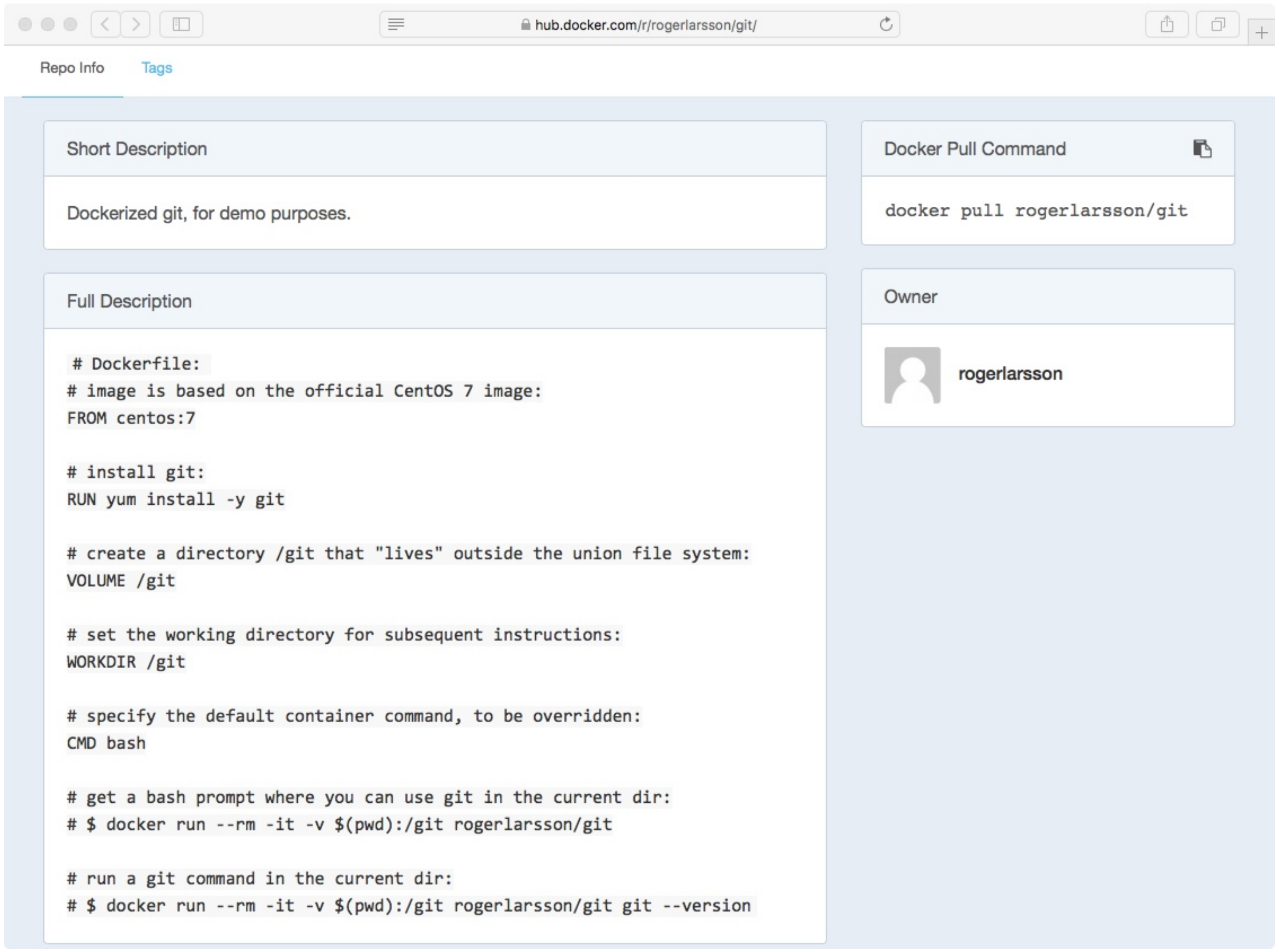
Portable git

```
$ docker run --rm -it -v $(pwd):/git rogerlarsson/git

[root@f4a25b34ddbe git]# pwd
[root@f4a25b34ddbe git]# git clone https://github.com/rogerlarsson/layersofcake.git
[root@f4a25b34ddbe git]# cd layersofcake/
[root@f4a25b34ddbe layersofcake]# git log

[root@f4a25b34ddbe layersofcake]# w
 [root@f4a25b34ddbe layersofcake]# id
[root@f4a25b34ddbe layersofcake]# ps faux

[root@f4a25b34ddbe layersofcake]# ls -la /
[root@f4a25b34ddbe layersofcake]# cat /etc/centos-release
```

Images and containers

```
$ docker image ls
$ docker images


$ docker container ls -a
$ docker ps -a
```

Recipes for cake



Short Description

Dockerized git, for demo purposes.

Docker Pull Command

```
docker pull rogerlarsson/git
```

Owner

rogerlarsson

Full Description

```
# Dockerfile:
# image is based on the official CentOS 7 image:
FROM centos:7

# install git:
RUN yum install -y git

# create a directory /git that "lives" outside the union file system:
VOLUME /git

# set the working directory for subsequent instructions:
WORKDIR /git

# specify the default container command, to be overridden:
CMD bash

# get a bash prompt where you can use git in the current dir:
# $ docker run --rm -it -v $(pwd):/git rogerlarsson/git

# run a git command in the current dir:
# $ docker run --rm -it -v $(pwd):/git rogerlarsson/git git --version
```

Stacks of read-only file system layers

```
$ docker images
$ docker history rogerlarsson/git:latest
```

Union file systems

Multiple file systems overlaid, appearing as a single file system

Containers add a read/write file system layer on top of the read-only image layers

Creating an image

```
$ nano clock.py
```

```
# clock.py

from datetime import datetime
from time import sleep

while True:
     print(datetime.now())
     sleep(3)
```

```
$ nano Dockerfile
```

```
# Dockerfile
FROM python:3-alpine

# install nano:
RUN apk add --no-cache nano

# create an /app directory:
RUN mkdir -vp /app

# copy the application code:
COPY clock.py /app

# set the working directory:
WORKDIR /app

# specify the container command:
CMD python clock.py
```

```
$ docker build -t clock_app .
```

Creating a container

```
$ docker images

$ docker run -it clock_app
$ # leave container using ^P^Q

$ docker ps −a
$ docker logs --follow 2380
$ docker attach 2380

$ docker exec -it 2380 sh
```

Swarms

```
$ docker swarm init

$ docker node ls
```

Stacks

```
$ nano clocks.yml
```

```
# clocks.yml
# docker stack deploy -c clocks.yml lotsofclocks

version: "3"

services:
  clock:
    image: clock_app:latest
    deploy:
      replicas: 10
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
```

Deployment

```
$ docker stack deploy -c clocks.yml lotsofclocks

$ docker stack ls

$ docker stack ps lotsofclocks

$ docker service ls
```

Scaling

```
$ nano clocks.yml    # replicas: 20

$ docker stack deploy -c clocks.yml lotsofclocks

$ docker stack ps lotsofclocks

$ docker service ls

$ docker stack rm lotsofclocks
```

Well done!

```
docker version
docker --help | less
docker run -p 8083:8083 -it epflsti/octave-x11-novnc-docker:latest
docker pull rogerlarsson/git
docker run rogerlarsson/git echo hello world
docker run --rm -it -v $(pwd):/git rogerlarsson/git
docker images
docker ps -a
docker history rogerlarsson/git:latest
docker build -t clock_app .
docker run -it clock_app
docker logs -f 7d39
docker attach 7d39
docker exec -it 7d39 sh
docker swarm init
docker node ls
docker stack deploy -c clocks.yml lotsofclocks
docker stack ls
docker stack ps lotsofclocks
docker service ls
docker stack rm lotsofclocks
# docker swarm leave --force
```