



Docker Containers

Marko Ambrož,
Žiga Hudolin
Ministrstvo za javno upravo
DIREKTORAT ZA INFORMATIKO

“Vendor lock in” with DOCKER ???

DOCKER CLIENT

DOCKER ENGINE

DOCKER SWARM

DOCKER COMPOSE

DOCKER MACHINE

DOCKER HUB

DOCKER REGISTRY

DOCKER CLOUD

DOCKER UCP
Universal Control Plane

DOCKER TOOLBOX

<http://blog.dennybritz.com/2015/10/01/a-brief-guide-to-the-docker-ecosystem/>

Supported operating systems

General purpose operating systems:

Amazon EC2
Arch Linux
Azure Microsoft
CentOS
CRUX Linux
Debian
Fedora
FrugalWare
Gentoo
Google Cloud
Joyent
Mac OS X
Oracle Linux
Rackspace Cloud
Red Hat Enterprise Linux
SoftLayer IBM
openSUSE and SUSE Linux
Ubuntu
Windows 20+

Container optimized operating systems:

RedHat Project Atomic
Boot2Docker
CoreOS
OSv
VMWare PhotonOS
RancherOS
Ubuntu Core 7+

Dockerization of applications

GITHUB
(GitLab)

→ DOCKER BUILD →

DOCKERHUB
(Docker Registry)

↑ DOCKERFILE

```
FROM ubuntu
RUN apt-get update
RUN apt-get -y install git build-essential
RUN apt-get -y install nodejs npm
RUN apt-get install -y git-core zip
RUN apt-get clean
WORKDIR /opt/coder/
RUN git clone github: googlecoder.git .
RUN ls -la && pwd
WORKDIR /opt/coder/coder-apps
RUN ./install_common.sh ../coder-base
WORKDIR /opt/coder/coder-base
RUN npm install
COPY src/config.js /opt/coder/coder-base/
\ config.js#
#EXPOSE 80 443
EXPOSE 8180 8181
CMD ["nodejs", "server.js"]
```

↓ DOCKER PULL ↓

**Carina, Sloppy.io,
Tutum, StackDock,**
...

DOCKER RUN →

Are Docker Containers cloud neutral?

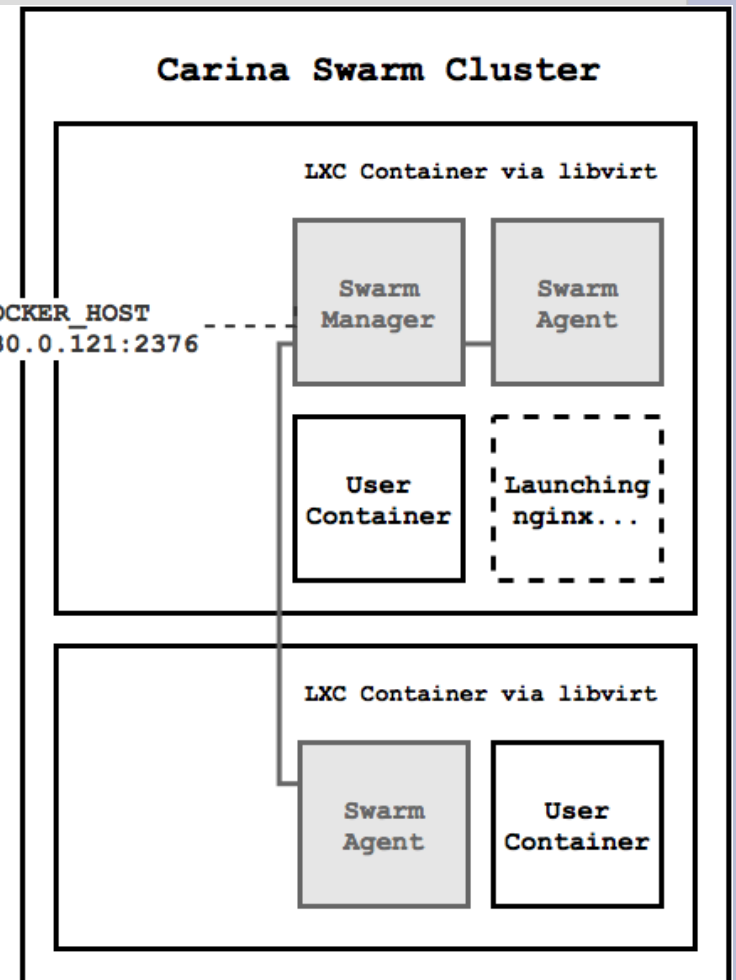
IaaS (Infrastructure as a Service)

e.g.: DigitalOcean droplets: Ubuntu, Fedora, Debian, CoreOS, Centos

CaaS (Container as a Service)

e.g.: RackSpace: CARINA

```
$ eval "$( carina env mycluster )"
$ echo $DOCKER_HOST
tcp://104.130.0.121:2376
$ docker run nginx
```



If the demo fails you can swim with:

**GoogleCoder in a Container at
DigitalOcean:**

<https://188.166.34.228:7443>

Password: Slnog32016



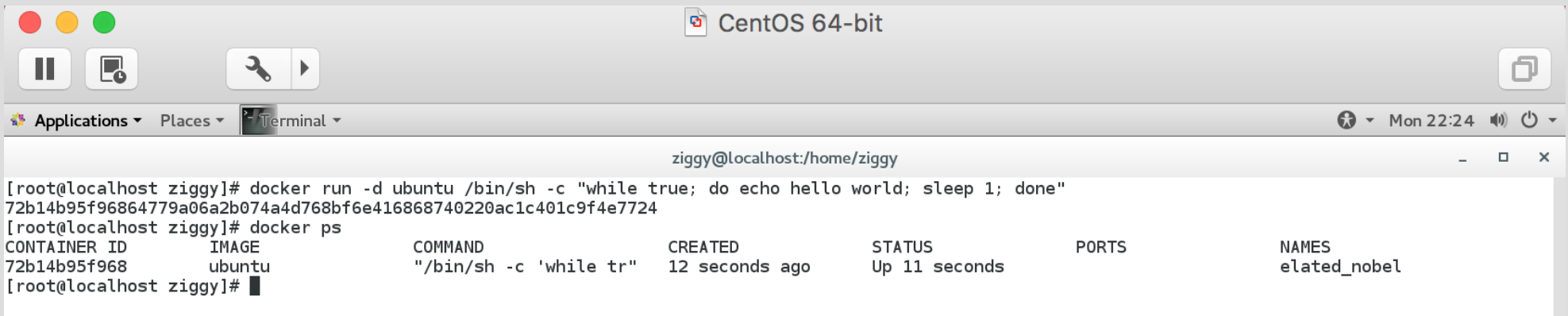
Demonstration

Example 1

- **Goal:** Create a container running as a deamon and a command that print hello world on screen every second
- **In this example we will use**
 - `docker run`; runs the container
 - `-d`; flag runs the container in the background
 - `ubuntu`; is the image you would like to run
 - `docker logs`; looks inside the container
- **Finally we specify a command to run:**
 - `/bin/sh -c "while true; do echo hello world; sleep 1; done"`

Example 1: docker run

- `docker run -d ubuntu /bin/sh -c "while true; do echo hello world; sleep 1; done"`



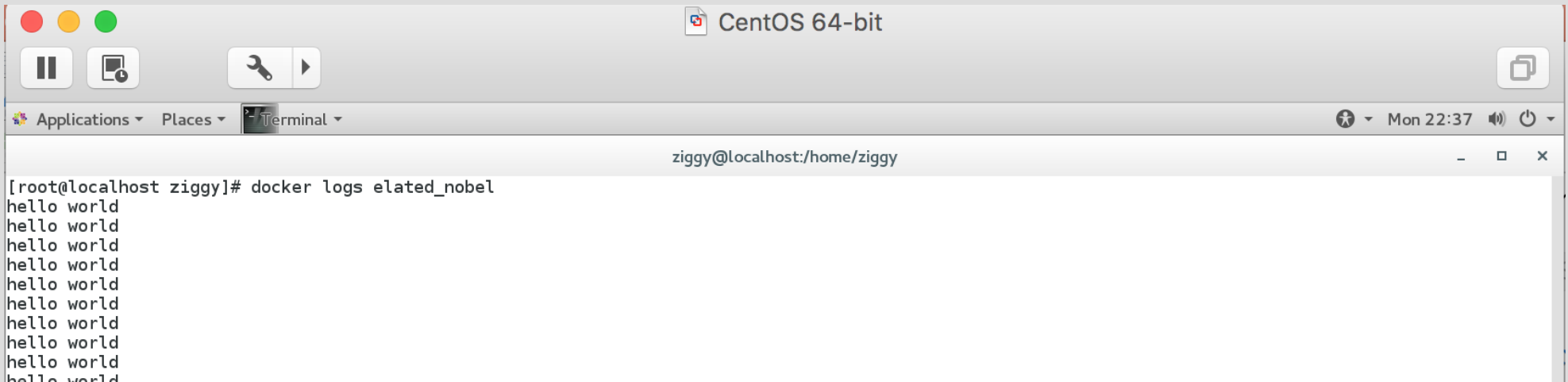
The screenshot shows a terminal window titled "CentOS 64-bit" with the following content:

```
ziggy@localhost:/home/ziggy
[root@localhost ziggy]# docker run -d ubuntu /bin/sh -c "while true; do echo hello world; sleep 1; done"
72b14b95f96864779a06a2b074a4d768bf6e416868740220ac1c401c9f4e7724
[root@localhost ziggy]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
72b14b95f968      ubuntu             "/bin/sh -c 'while tr"  12 seconds ago     Up 11 seconds      eLATED_NOBEL      elated_nobel
[root@localhost ziggy]#
```

- This long string “72b14b95f96864779a06a2b...” is called a container ID
- `docker ps` command queries the Docker daemon for information about all the containers.

Example 1: docker logs

- To see what is going on inside the container we use the docker logs command

A screenshot of a terminal window titled "CentOS 64-bit". The terminal shows the command `docker logs elated_nobel` being executed, resulting in ten lines of "hello world" output. The terminal prompt is `[root@localhost ziggy]#`. The window title bar includes standard Linux window controls and system information like "Mon 22:37".

```
[root@localhost ziggy]# docker logs elated_nobel
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
```

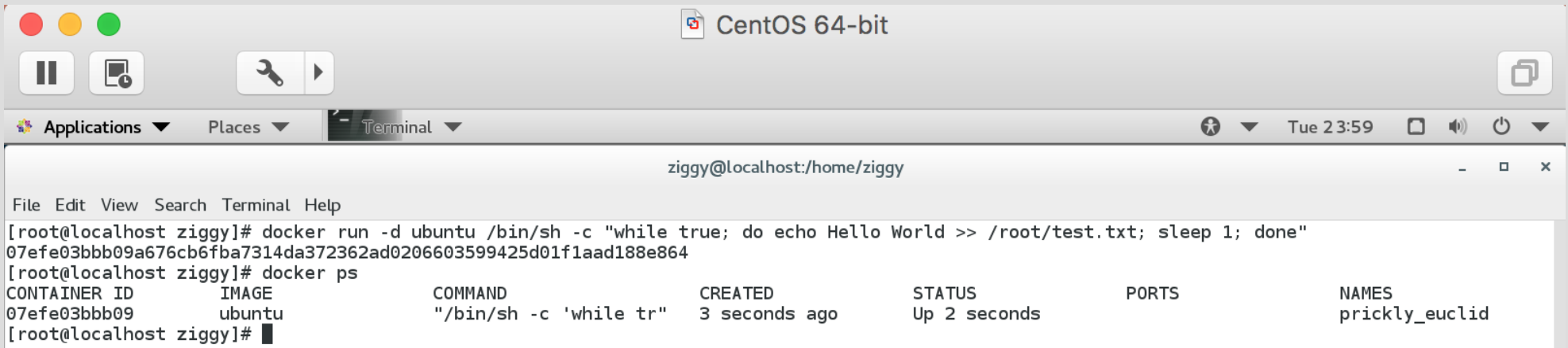
- docker stop command tells Docker to politely stop the running container and returns the name of the container it stopped in our case "elated_nobel"

Example 2

- Goal: Create a container running as a deamon and a command that print Hello World every second and write it in a specific file
- In this example we will use
 - docker run; runs the container
 - -d; flag runs the container in the background
 - ubuntu; is the image you would like to run
 - docker exec; runs a new bash session in a running container
- Finally we specify a command to run an write result in a file

Example 2: docker run

- `docker run -d ubuntu /bin/sh -c "while true; do echo Hello World >> /root/test.txt; sleep 1; done"`



The screenshot shows a terminal window titled "CentOS 64-bit" with the prompt `ziggy@localhost:/home/ziggy`. The user has executed the command `docker run -d ubuntu /bin/sh -c "while true; do echo Hello World >> /root/test.txt; sleep 1; done"`, which returned the long container ID `07efe03bbb09a676cb6fba7314da372362ad0206603599425d01f1aad188e864`. Subsequently, the user ran `docker ps`, resulting in the following table:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
07efe03bbb09	ubuntu	"/bin/sh -c 'while tr"	3 seconds ago	Up 2 seconds		prickly_euclid

- This long string “07efe03bbb09a676cb6fba73...” is called a container ID
- `docker ps` command queries the Docker daemon for information about all the containers.

Example 2: export TERM

- We set TERM environment variable in running container

The image shows two terminal windows from a CentOS 64-bit environment. The top window shows the process of starting a container and setting the TERM variable. The bottom window shows the output of the 'top' command inside the container.

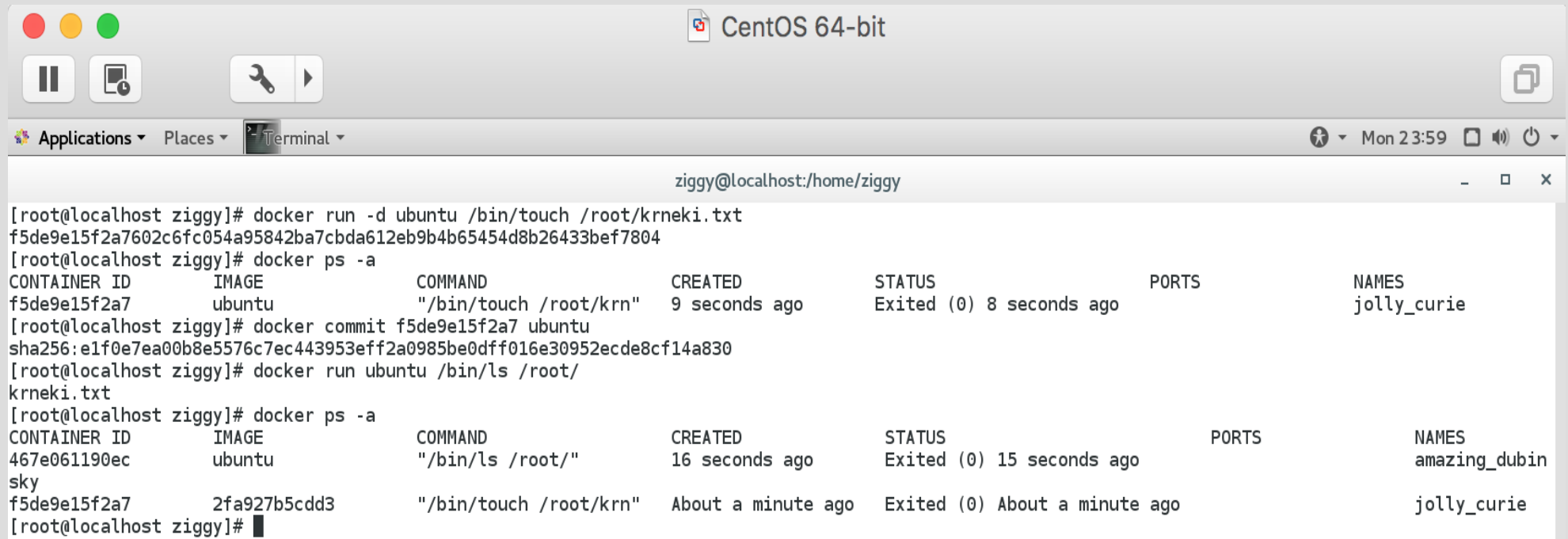
```
File Edit View Search Terminal Help
[root@localhost ziggy]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS       NAMES
07efe03bbb09   ubuntu   "/bin/sh -c 'while tr"   11 minutes ago Up 11 minutes   prickly_euclid
[root@localhost ziggy]# docker exec -it 07efe03bbb09 bash
root@07efe03bbb09:/# top
TERM environment variable not set.
root@07efe03bbb09:/# export TERM=linux
root@07efe03bbb09:/# top
```

```
File Edit View Search Terminal Help
top - 22:12:49 up 1:07, 0 users, load average: 0.00, 0.04, 0.15
Tasks:  4 total,  1 running,  3 sleeping,  0 stopped,  0 zombie
%Cpu(s):  0.8 us,  0.3 sy,  0.0 ni, 98.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 1868688 total,  338224 free,  954212 used,  576252 buff/cache
KiB Swap: 2097148 total,  2065252 free,  31896 used,  641876 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
   1 root       20   0   4500    688   588  S   0.0   0.0   0:00.22  sh
  727 root       20   0  18256   2012  1504  S   0.0   0.1   0:00.00  bash
  866 root       20   0  36652  1708  1264  R   0.0   0.1   0:00.00  top
  884 root       20   0   4372    376   304  S   0.0   0.0   0:00.00  sleep
```

Example 3: docker commit

- Goal: update a container created from an image with a specific file and commit the result to an image
- In this example we will use
 - docker commit; create a new image from a container's changes



```
CentOS 64-bit
Applications ▾ Places ▾ Terminal ▾
ziggy@localhost:/home/ziggy

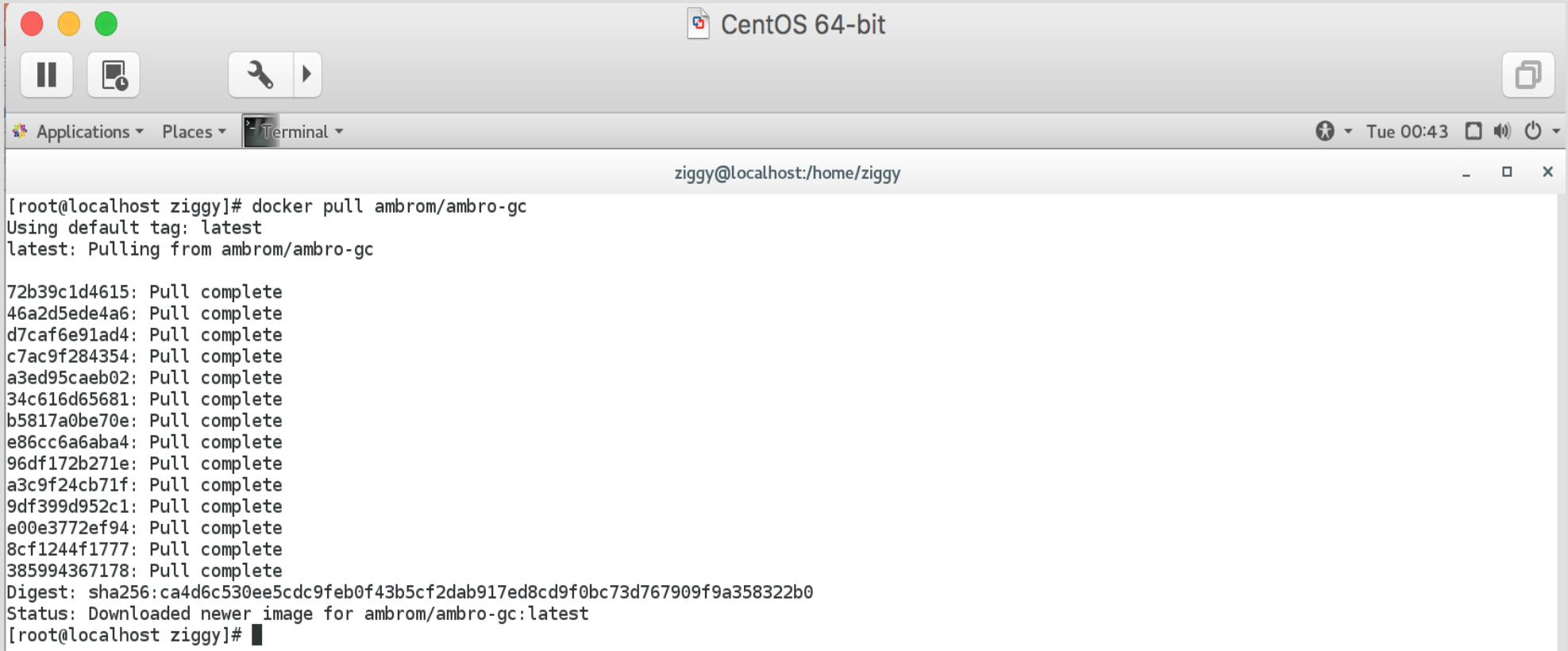
[root@localhost ziggy]# docker run -d ubuntu /bin/touch /root/krneki.txt
f5de9e15f2a7602c6fc054a95842ba7cbda612eb9b4b65454d8b26433bef7804
[root@localhost ziggy]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
f5de9e15f2a7        ubuntu             "/bin/touch /root/krn"  9 seconds ago      Exited (0) 8 seconds ago           jolly_curie
[root@localhost ziggy]# docker commit f5de9e15f2a7 ubuntu
sha256:e1f0e7ea00b8e5576c7ec443953eff2a0985be0dff016e30952ecde8cf14a830
[root@localhost ziggy]# docker run ubuntu /bin/ls /root/
krneki.txt
[root@localhost ziggy]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
467e061190ec        ubuntu             "/bin/ls /root/"    16 seconds ago      Exited (0) 15 seconds ago           amazing_dubin
sky
f5de9e15f2a7        2fa927b5cdd3       "/bin/touch /root/krn"  About a minute ago  Exited (0) About a minute ago           jolly_curie
[root@localhost ziggy]#
```

Example 4

- Goal: Install application Google Coder
- In this example we will use
 - docker pull; pull an image or a repository from the registry
 - docker images; list images
- Finally we check our application in browser

Example 4: docker pull

- We will use image from ambrom/ambro-gc repository on dockerhub

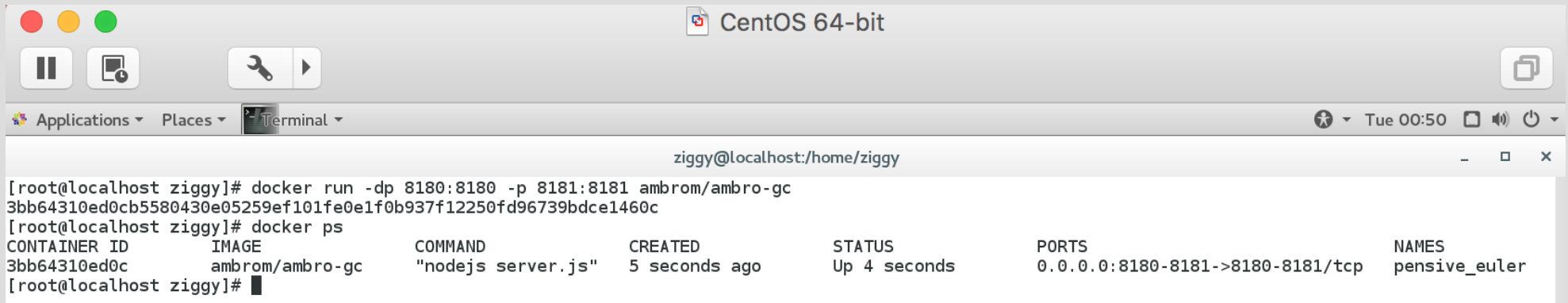
A screenshot of a terminal window titled "CentOS 64-bit" showing the execution of the command "docker pull ambrom/ambro-gc". The terminal output displays the progress of pulling the image, including a list of 16 layers and their SHA256 hashes, followed by the digest and status of the pull operation.

```
[root@localhost ziggy]# docker pull ambrom/ambro-gc
Using default tag: latest
latest: Pulling from ambrom/ambro-gc

72b39c1d4615: Pull complete
46a2d5ede4a6: Pull complete
d7caf6e91ad4: Pull complete
c7ac9f284354: Pull complete
a3ed95caeb02: Pull complete
34c616d65681: Pull complete
b5817a0be70e: Pull complete
e86cc6a6aba4: Pull complete
96df172b271e: Pull complete
a3c9f24cb71f: Pull complete
9df399d952c1: Pull complete
e00e3772ef94: Pull complete
8cf1244f1777: Pull complete
385994367178: Pull complete
Digest: sha256:ca4d6c530ee5cdc9feb0f43b5cf2dab917ed8cd9f0bc73d767909f9a358322b0
Status: Downloaded newer image for ambrom/ambro-gc:latest
[root@localhost ziggy]#
```

Example 4: docker run

- With command `docker run` we will run our application but we have to specify ports “8180 and 8181”



```
CentOS 64-bit
Applications Places Terminal
Tue 00:50
ziggy@localhost:/home/ziggy
[root@localhost ziggy]# docker run -dp 8180:8180 -p 8181:8181 ambrom/ambro-gc
3bb64310ed0cb5580430e05259ef101fe0e1f0b937f12250fd96739bdce1460c
[root@localhost ziggy]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
3bb64310ed0c        ambrom/ambro-gc    "nodejs server.js" 5 seconds ago      Up 4 seconds       0.0.0.0:8180-8181->8180-8181/tcp    pensive_euler
[root@localhost ziggy]#
```

- We will open the browser and check if application is alive and responsive

Primer 4: test aplikacije

➤ <https://localhost:8181>

Coder - Mozilla Firefox

Coder

<https://localhost:8181/app/coder?firstuse> Search

Update your Coder's settings here.

You can create new apps by clicking this green '+' button.

Each colored block is an app. Try clicking the pink 'Hello Coder' app to get started.

Getting Started

Check out the quick tips above. For a more detailed tour, go to goo.gl/coder. When you're ready to dive in, just click 'Got It'.

Got It