Intro to Docker

Docker Setup

- Installing Docker
  - MacOS

    https://docs.docker.com/docker-for-mac
  - Windows

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

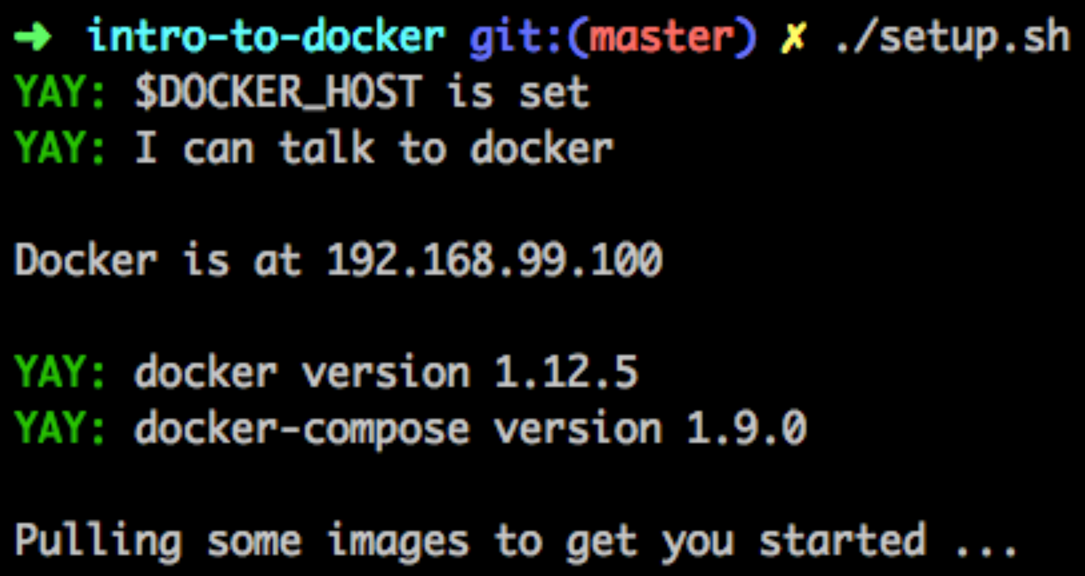
    ```
    curl -sSL https://get.docker.com/ | sh
    ```

Check Setup

- Clone the workshop repository

  git clone ssh://git@stash.reecenet.org:7999/train/intro-to-docker.git

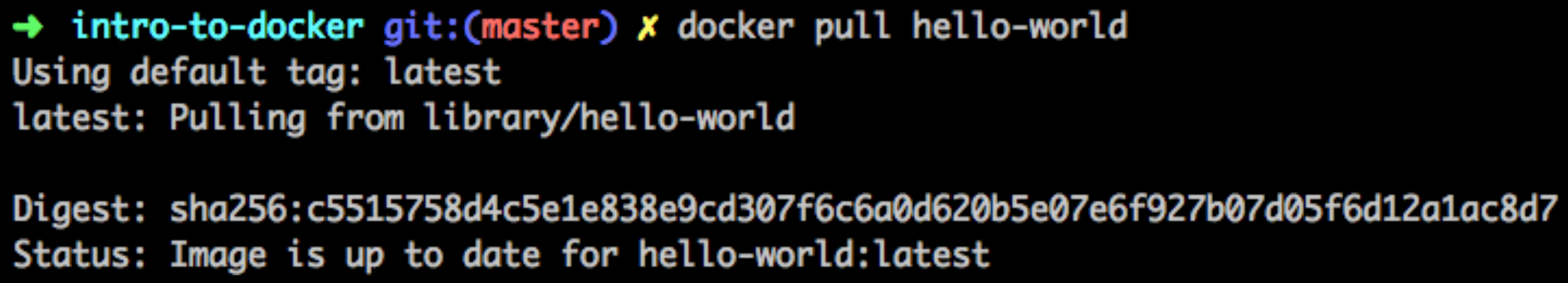
- Check your setup



```
➜ intro-to-docker git:(master) ✗ ./setup.sh
YAY: $DOCKER_HOST is set
YAY: I can talk to docker

Docker is at 192.168.99.100

YAY: docker version 1.12.5
YAY: docker-compose version 1.9.0

Pulling some images to get you started ...
```

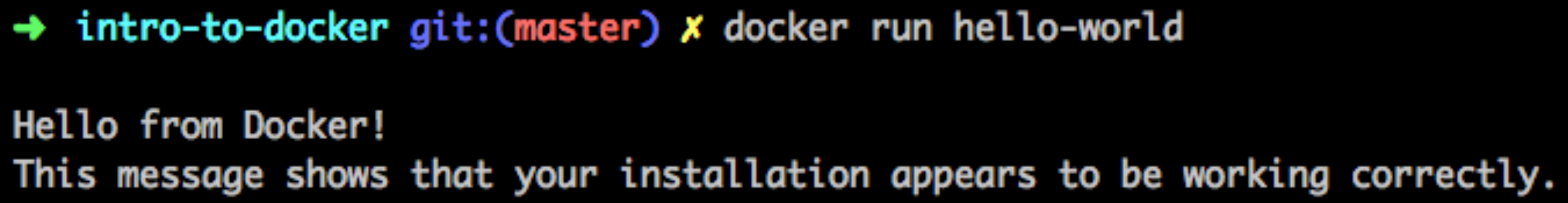
Hello World

Pull an **Image**



```
➜ intro-to-docker git:(master) ✗ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world

Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7
Status: Image is up to date for hello-world:latest
```

Run a **Container**



```
➜ intro-to-docker git:(master) ✗ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

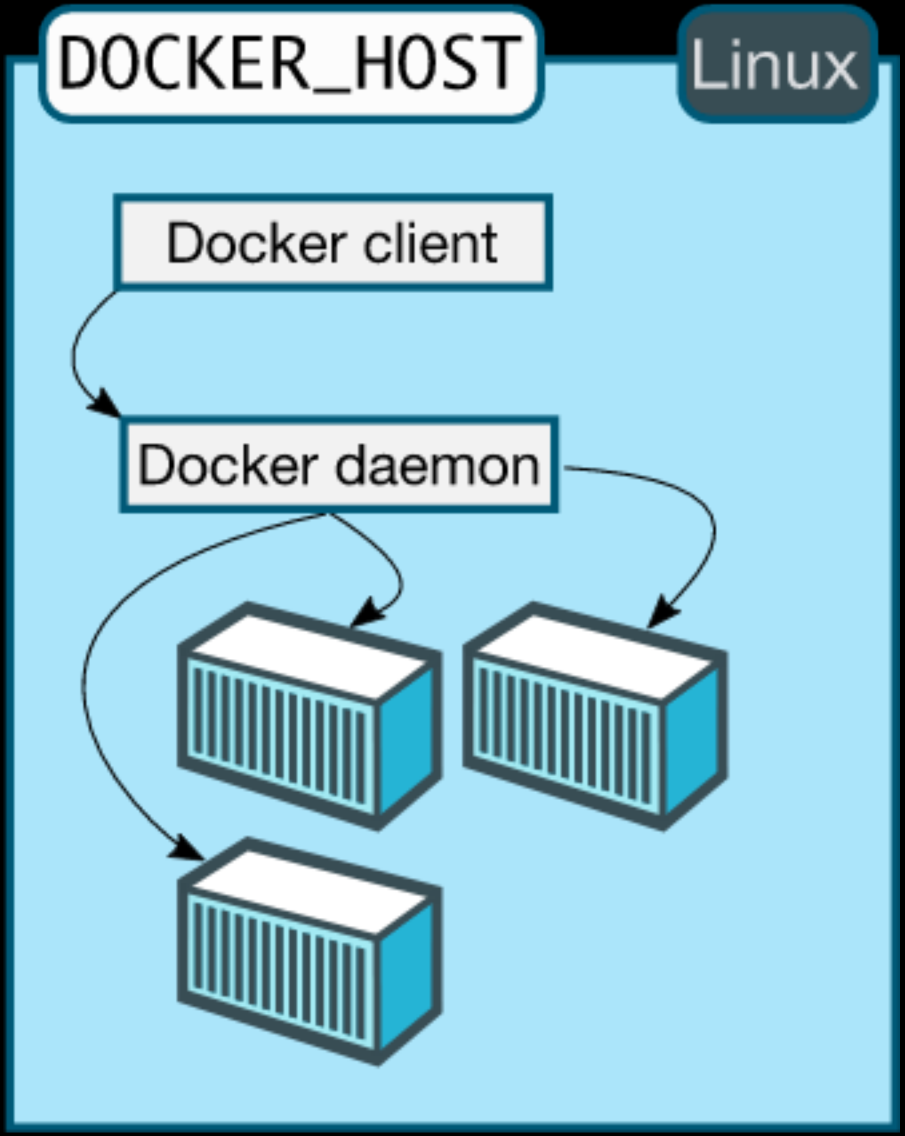
Run Command

```
$ docker run ubuntu dpkg -l
```

```
$ docker run -i -t ubuntu bash
```

- `-i = Keep STDIN open`
- `-t = Allocate a pseudo-TTY`

Docker Engine



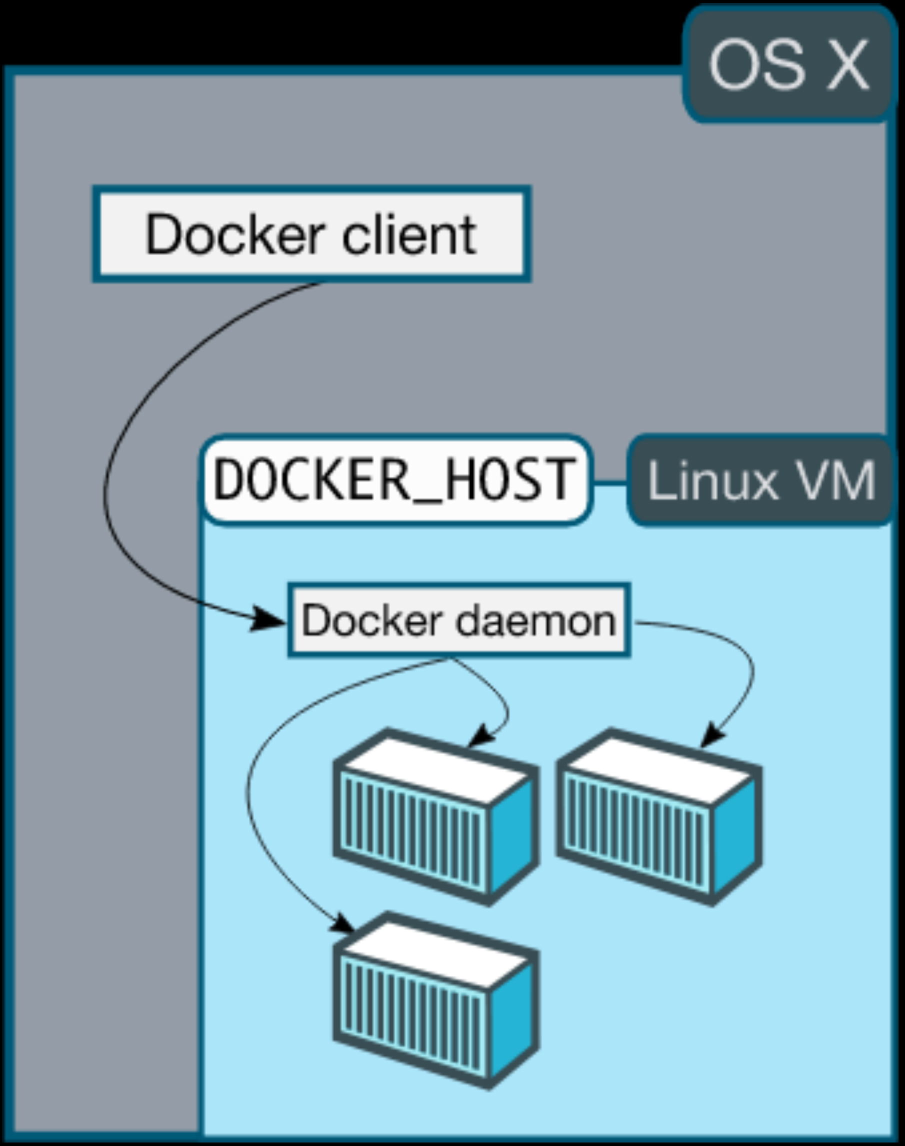
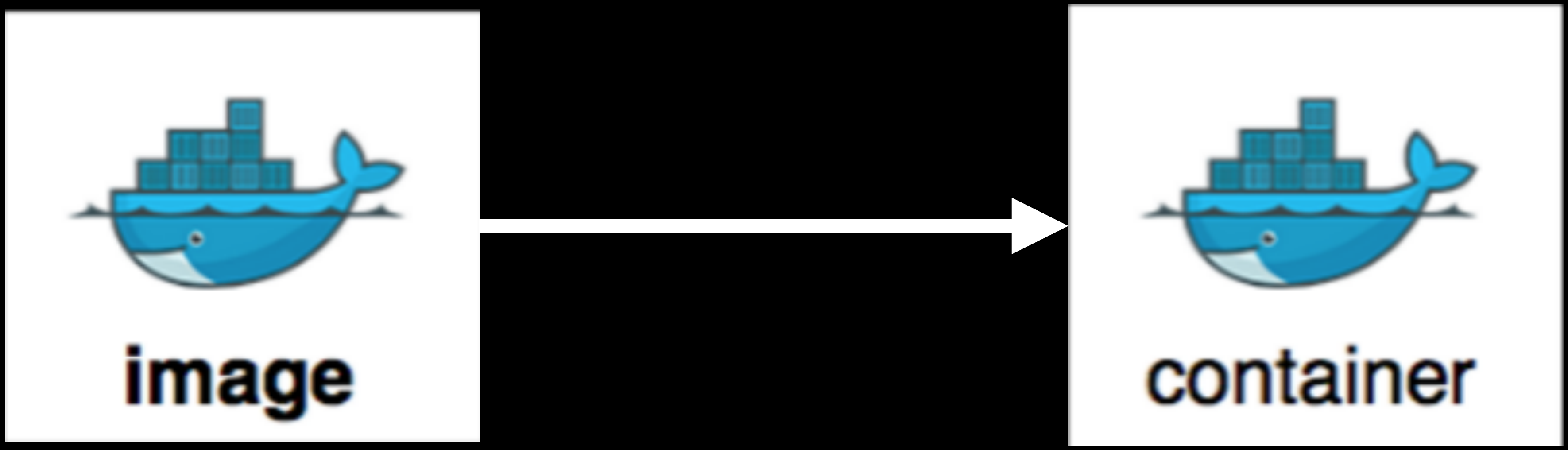


Image & Container



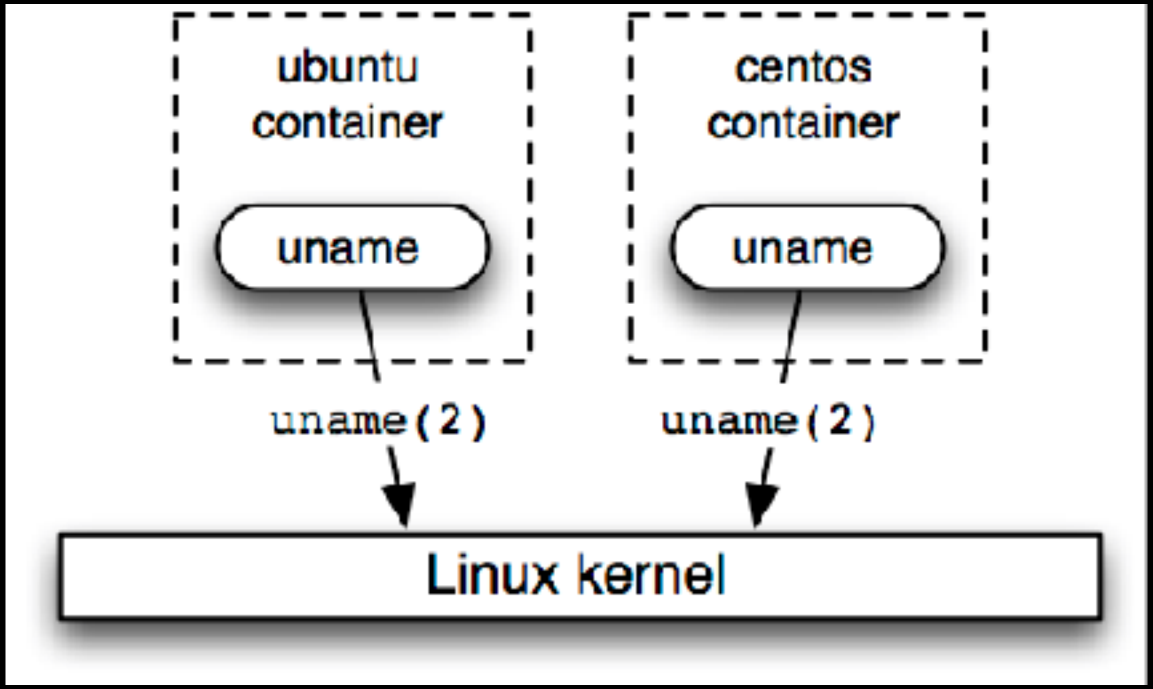
- Passive
- File system snapshot
- Like: VM Image, AMI

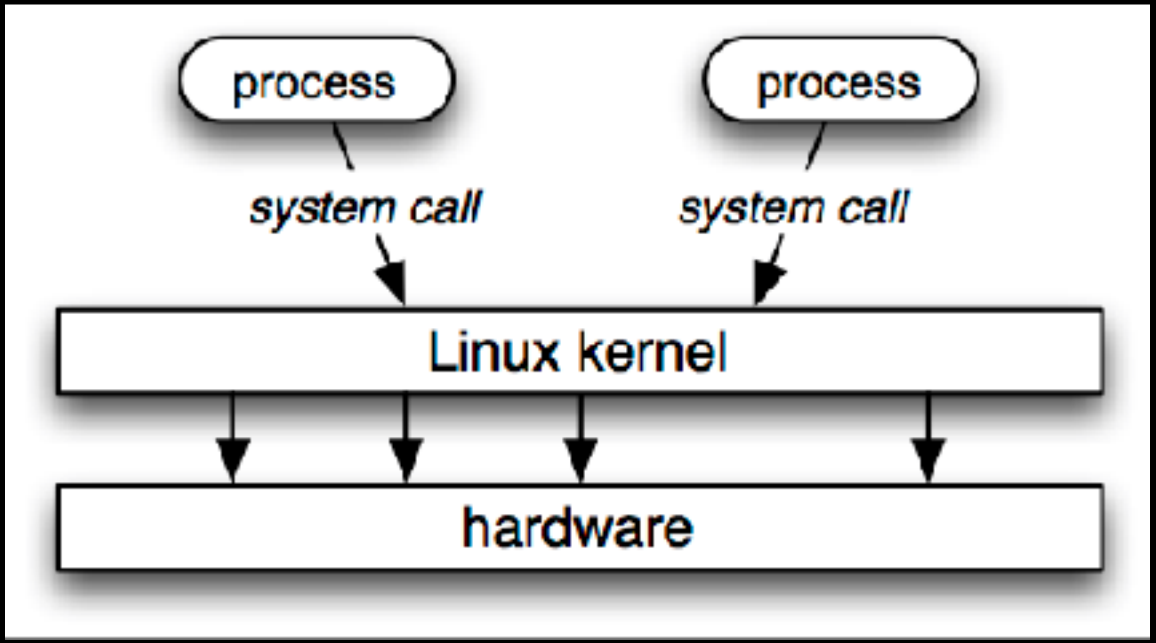
- Active
- Group of processes
- Like: Virtual Machine, EC2

Kernel

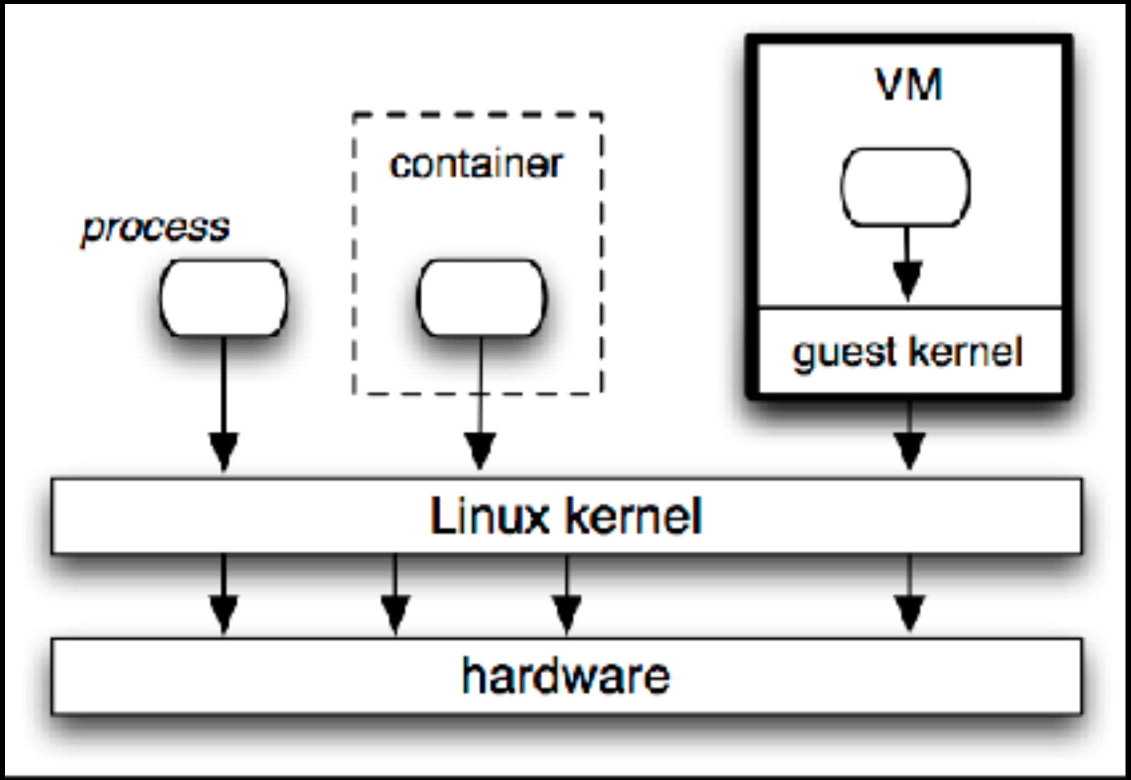
Try these two commands:

```
$ docker run ubuntu name -a
$ docker run centos name -a
```





Container VS Virtualisation



	VMs	Docker
image size	Gigabytes	Megabytes
startup time	minutes	sub-second
Linux kernel is	separate	shared
isolation is	complete	pretty good
used to encapsulate	servers	services

`Try to measure the overhead of running a command in a container`

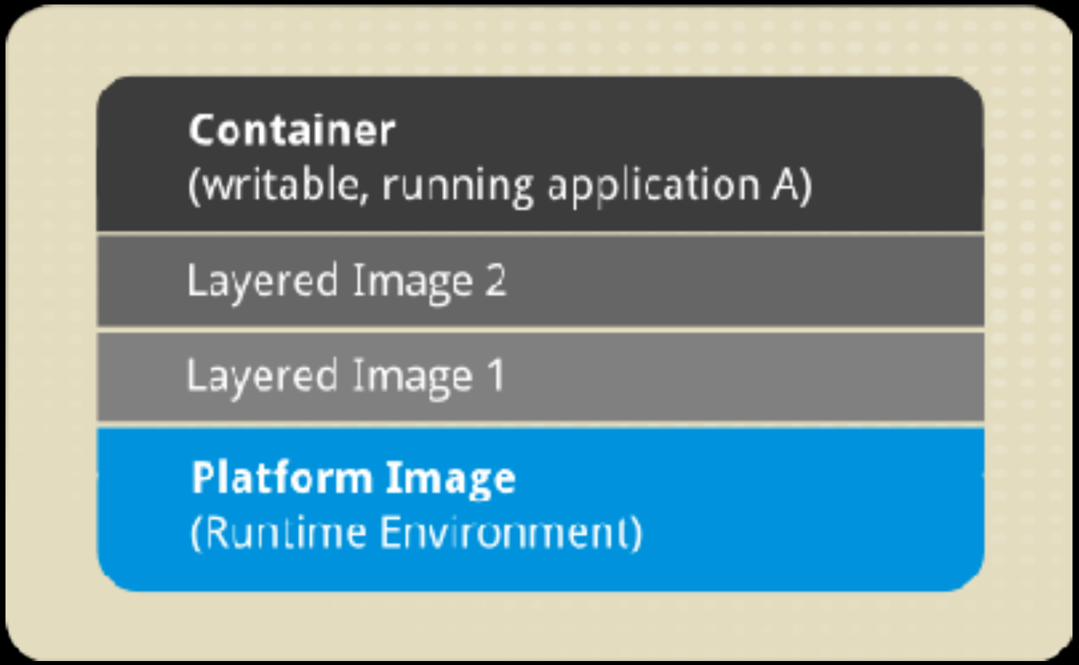
```
➜ intro-to-docker git:(master) ✗ time docker run ubuntu bash -c "time sleep 1"
```

Linux cgroups

Isolate processes by limiting access to:

- hardware resources
- other processes

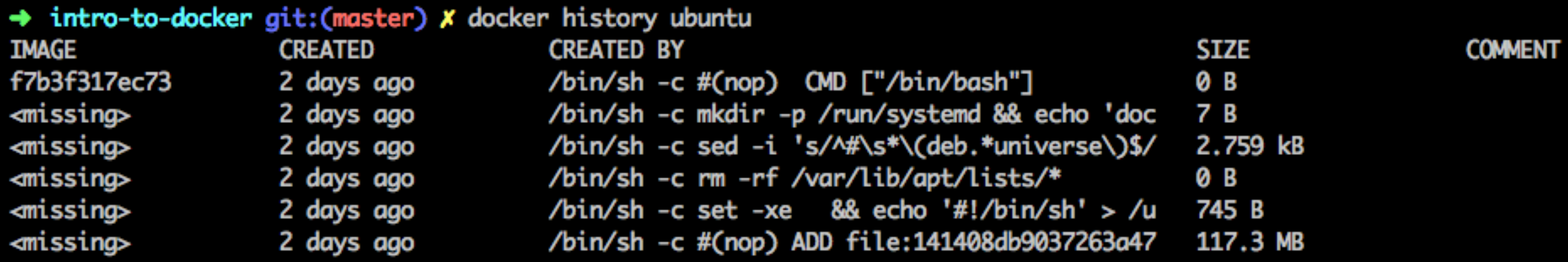
Image Layers



Union file systems

- Images are made of "layers", and can share base layers.
- Container root file system is just another(writable) layer.

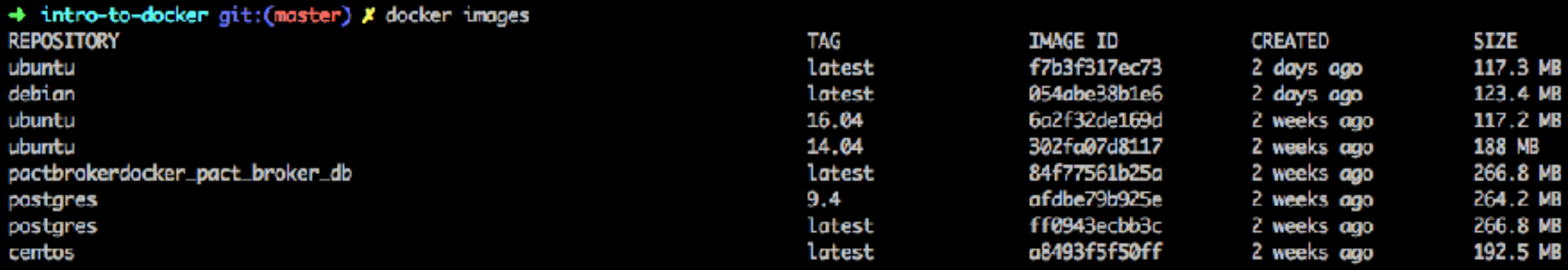
- View image layers



```
➜ intro-to-docker git:(master) ✗ docker history ubuntu
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
f7b3f317ec73        2 days ago          /bin/sh -c #(nop)  CMD ["/bin/bash"]            0 B
<missing>           2 days ago          /bin/sh -c mkdir -p /run/systemd && echo 'doc   7 B
<missing>           2 days ago          /bin/sh -c sed -i 's/^#\s*\(deb.*universe\)$/   2.759 kB
<missing>           2 days ago          /bin/sh -c rm -rf /var/lib/apt/lists/*          0 B
<missing>           2 days ago          /bin/sh -c set -xe   && echo '#!/bin/sh' > /u   745 B
<missing>           2 days ago          /bin/sh -c #(nop) ADD file:141408db9037263a47   117.3 MB
```

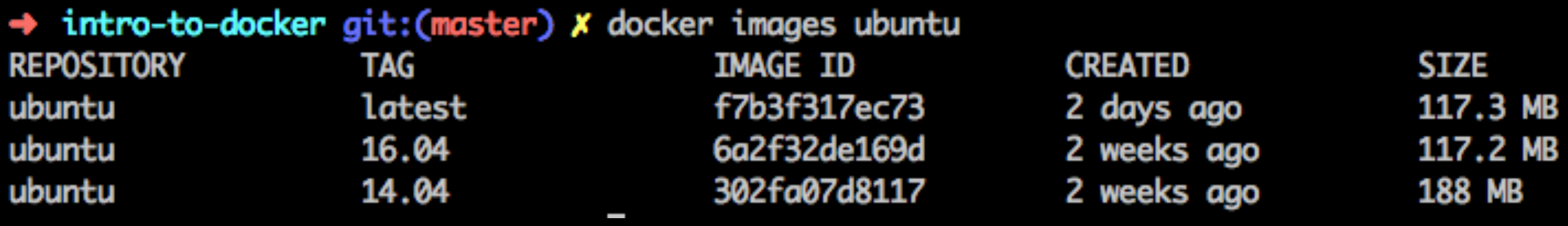
List images

- List images on your docker host



```
➜  intro-to-docker git:(master) ✗ docker images
REPOSITORY                        TAG       IMAGE ID       CREATED       SIZE
ubuntu                            latest    f7b3f317ec73   2 days ago    117.3 MB
debian                            latest    054abe38b1e6   2 days ago    123.4 MB
ubuntu                            16.04     6a2f32de169d   2 weeks ago   117.2 MB
ubuntu                            14.04     302fa07d8117   2 weeks ago   188 MB
pactbrokerdocker_pact_broker_db   latest    84f77561b25a   2 weeks ago   266.8 MB
postgres                          9.4       afdbe79b925e   2 weeks ago   264.2 MB
postgres                          latest    ff0943ecbb3c   2 weeks ago   266.8 MB
centos                            latest    a8493f5f50ff   2 weeks ago   192.5 MB
```

- List images in the ubuntu repository



```
➜  intro-to-docker git:(master) ✗ docker images ubuntu
REPOSITORY   TAG      IMAGE ID       CREATED       SIZE
ubuntu       latest   f7b3f317ec73   2 days ago    117.3 MB
ubuntu       16.04    6a2f32de169d   2 weeks ago   117.2 MB
ubuntu       14.04    302fa07d8117   2 weeks ago   188 MB
```

Image namespace & tag

[<NAMESPACE>/]<REPOSITORY>[:<TAG>]

- Image repositories can have an optional namespace, which pull image from a non-standard namespace.

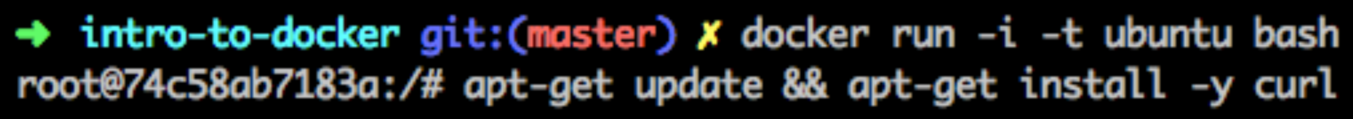
```
$ docker pull jasonxia/node-hello-world
```

- Image repositories can contain multiple images, identified by tag
  default to "latest" tag

```
$ docker run ubuntu:16.04 cat /etc/lsb-release
$ docker run ubuntu:14.04 cat /etc/lsb-release
```

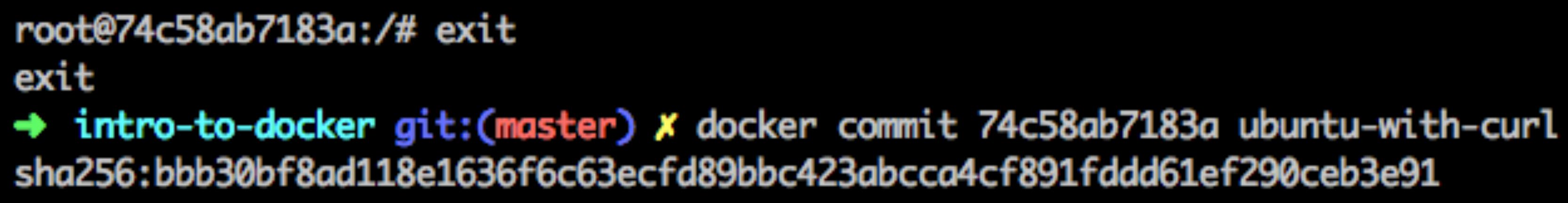
Build Image - 1

- Install curl in an ubuntu container



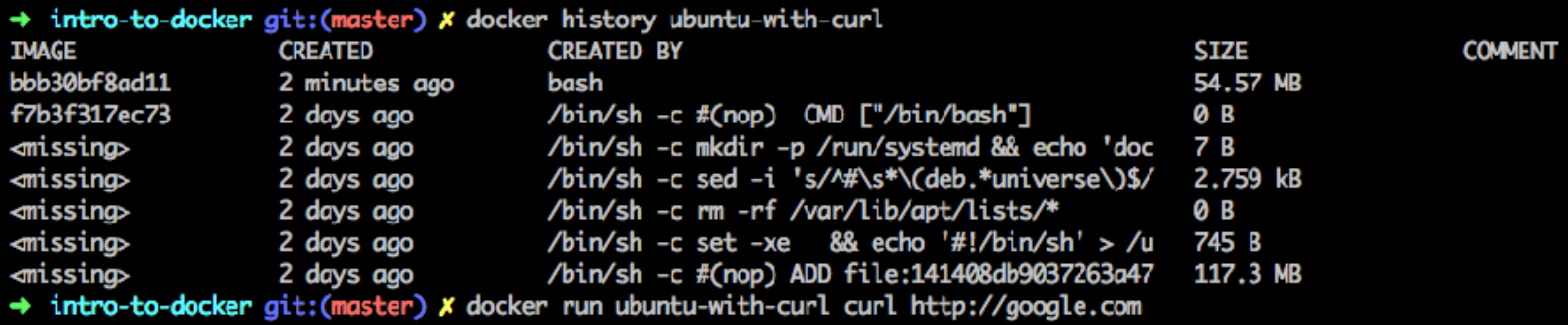
```
➜ intro-to-docker git:(master) ✗ docker run -i -t ubuntu bash
root@74c58ab7183a:/# apt-get update && apt-get install -y curl
```

- Exit and commit the container to create an image



```
root@74c58ab7183a:/# exit
exit
➜ intro-to-docker git:(master) ✗ docker commit 74c58ab7183a ubuntu-with-curl
sha256:bbb30bf8ad118e1636f6c63ecfd89bbc423abcca4cf891fddd61ef290ceb3e91
```

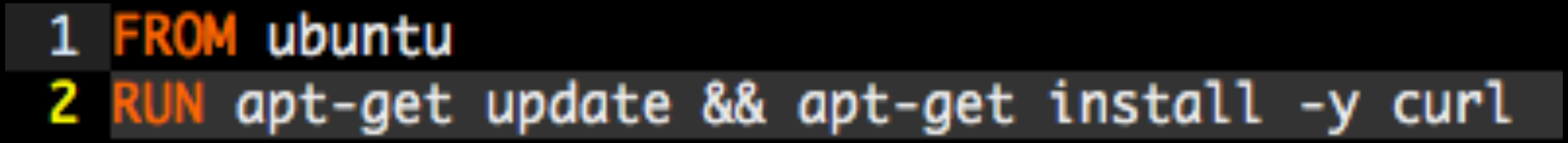
- Check out the new image



```
➜ intro-to-docker git:(master) ✗ docker history ubuntu-with-curl
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
bbb30bf8ad11        2 minutes ago       bash                                            54.57 MB
f7b3f317ec73        2 days ago          /bin/sh -c #(nop)  CMD ["/bin/bash"]            0 B
<missing>           2 days ago          /bin/sh -c mkdir -p /run/systemd && echo 'doc   7 B
<missing>           2 days ago          /bin/sh -c sed -i 's/^#\s*\(deb.*universe\)$/   2.759 kB
<missing>           2 days ago          /bin/sh -c rm -rf /var/lib/apt/lists/*          0 B
<missing>           2 days ago          /bin/sh -c set -xe   && echo '#!/bin/sh' > /u   745 B
<missing>           2 days ago          /bin/sh -c #(nop) ADD file:141408db9037263a47   117.3 MB
➜ intro-to-docker git:(master) ✗ docker run ubuntu-with-curl curl http://google.com
```

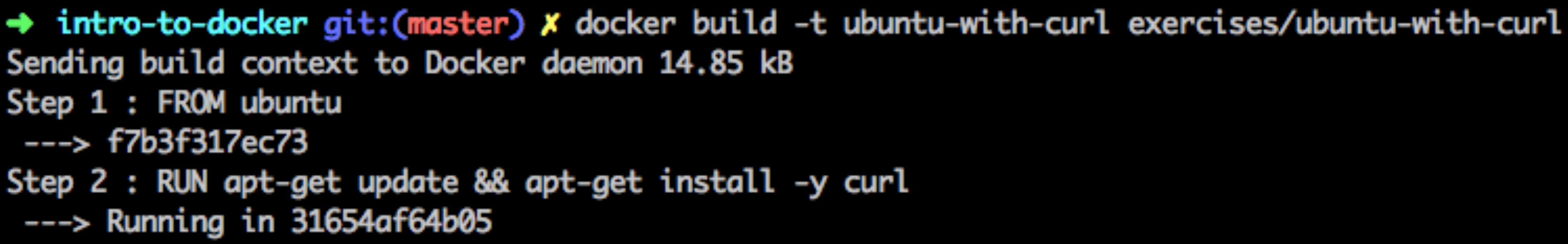
Build Image - 2

- Use a Dockerfile



```
FROM ubuntu
RUN apt-get update && apt-get install -y curl
```

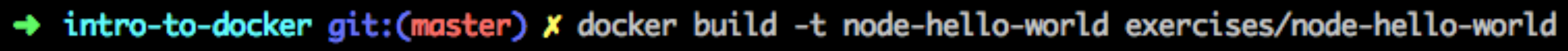
- Build an image



```
➜  intro-to-docker git:(master) ✗ docker build -t ubuntu-with-curl exercises/ubuntu-with-curl
Sending build context to Docker daemon 14.85 kB
Step 1 : FROM ubuntu
 ---> f7b3f317ec73
Step 2 : RUN apt-get update && apt-get install -y curl
 ---> Running in 31654af64b05
```

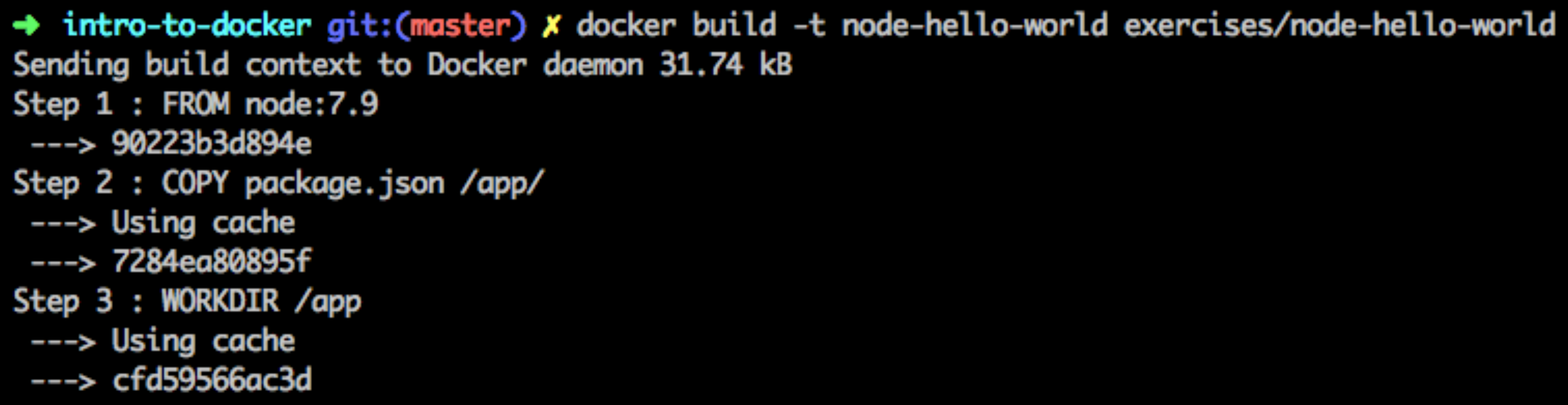
Build Cache

- Build hello world Node.js app



```
➜  intro-to-docker git:(master) ✗ docker build -t node-hello-world exercises/node-hello-world
```

- Build it again, any difference?



```
➜  intro-to-docker git:(master) ✗ docker build -t node-hello-world exercises/node-hello-world
Sending build context to Docker daemon 31.74 kB
Step 1 : FROM node:7.9
 ---> 90223b3d894e
Step 2 : COPY package.json /app/
 ---> Using cache
 ---> 7284ea80895f
Step 3 : WORKDIR /app
 ---> Using cache
 ---> cfd59566ac3d
```

- Make a change to index.js, then build again

Publish Image

- Sign up account at https://hub.docker.com

- Authenticate to Docker Hub

```
➜  intro-to-docker git:(master) ✗ docker login
```

- Tag an image into your namespace

```
➜  intro-to-docker git:(master) ✗ docer tag node-hello-world <NAMESPACE>/node-hello-world
```

- Push the image

```
➜  intro-to-docker git:(master) ✗ docer push <NAMESPACE>/node-hello-world
```

- Pull an image someone else pushed

Image Summary

Image-name pattern:

```
[<REGISTRY>/][<NAMESPACE>/]<REPOSITORY>[:<TAG>][@<DIGEST>]
```

- `<REGISTRY>` defaults to `docker.io` ("official" registry)
- `<NAMESPACE>` defaults to `library`
- `<TAG>` defaults to `latest`
- `<DIGEST>` is a SHA-256 checksum of the image manifest

These are all equivalent:

```
$ docker run hello-world
$ docker run hello-world:latest
$ docker run library/hello-world
$ docker run docker.io/library/hello-world:latest
$ docker run hello-world@sha256:48b5124b2768d2b917edcb640435044a97967……
```

Container

- Run a container as a daemon (in the background)

```
$ docker run -d hello-world
```

- Run a container with a name

```
$ docker run -d —name <your-name> hello-world
```

- List the running containers

```
$ docker ps
```

- list ALL the containers

```
$ docker ps -a
```

- Delete a container

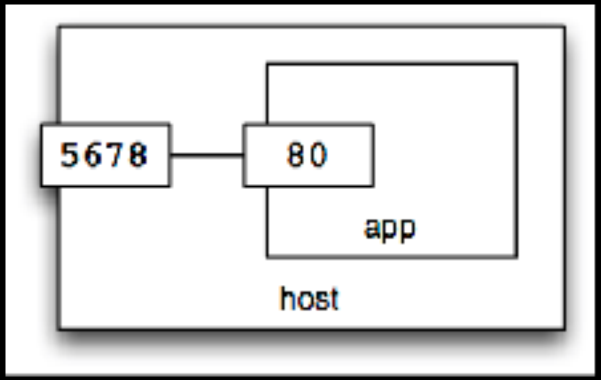
```
$ docker rm <ID>
```

- Delete a RUNNING container

```
$ docker rm -f <ID>
```

Port Mapping

```
$ docker run -d —name app -p 5678:80 node-hello-world
```

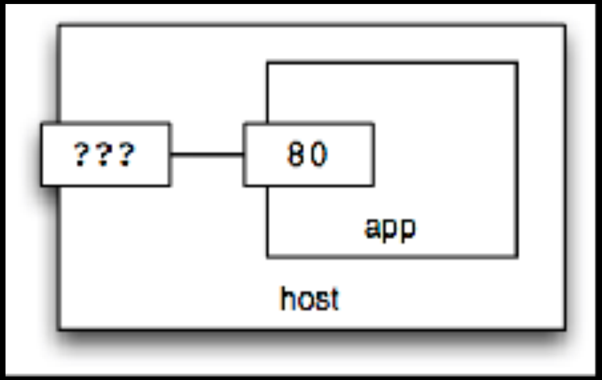


Docker will choose on if you do not specify a host port

```
$ docker run -d —name app -p 80 node-hello-world
```

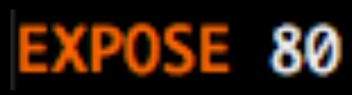
You can use docker port to find it out

```
$ docker port app 80
```



EXPOSE Port

You can specify in Dockerfile that a port should be exposed.



```
EXPOSE 80
```

But, you still need to tell Docker to expose it, with -P

```
$ docker run -d —name app -P node-hello-world
$ docker port app 80
```

Docker Logs

Output the logs

```
$ docker logs app
```

- With timestamps

  ```
  $ docker logs —timestamps app
  ```

- Real time

  ```
  $ docker logs —follow —timestamps app
  ```

Set ENV variables

The application depends on ENV variable(in index.js)

```
// Constants
var PORT = (process.env.PORT || "80");
var MESSAGE = (process.env.MESSAGE || "Hello World.");

```

You can set it in Dockerfile

```
ENV PORT 80
```

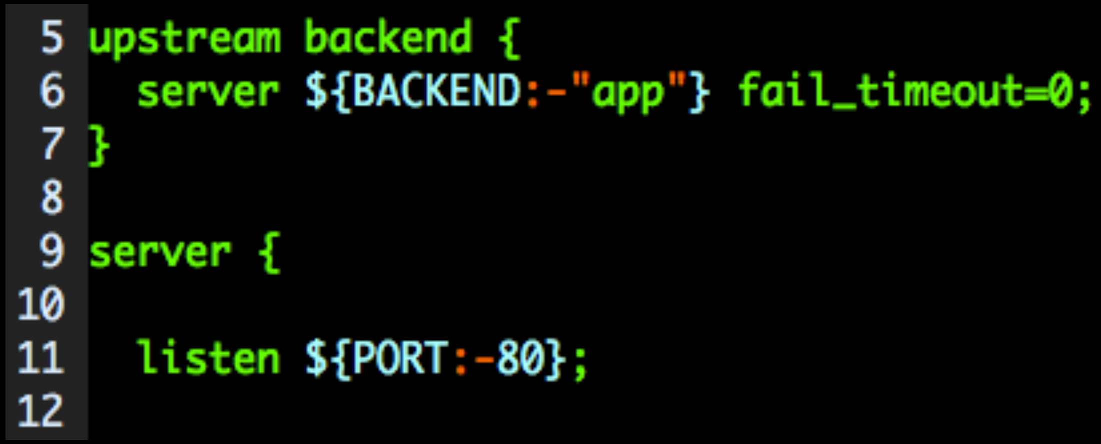
You can also set via command

```
$ docker run -d —name app -p 5678:80 -e MESSAGE=‘Good day!’ node-hello-world
```

Network

Pull nginx-proxy image - listen on port 80, forward requests to a host called “app”

```
$ docker pull jasonxia/nginx-proxy
```



```
upstream backend {
  server ${BACKEND:-"app"} fail_timeout=0;
}

server {

  listen ${PORT:-80};

```

Inter-Container Networking

Create a “network”

```
$ docker network create shared
$ docker network ls
```

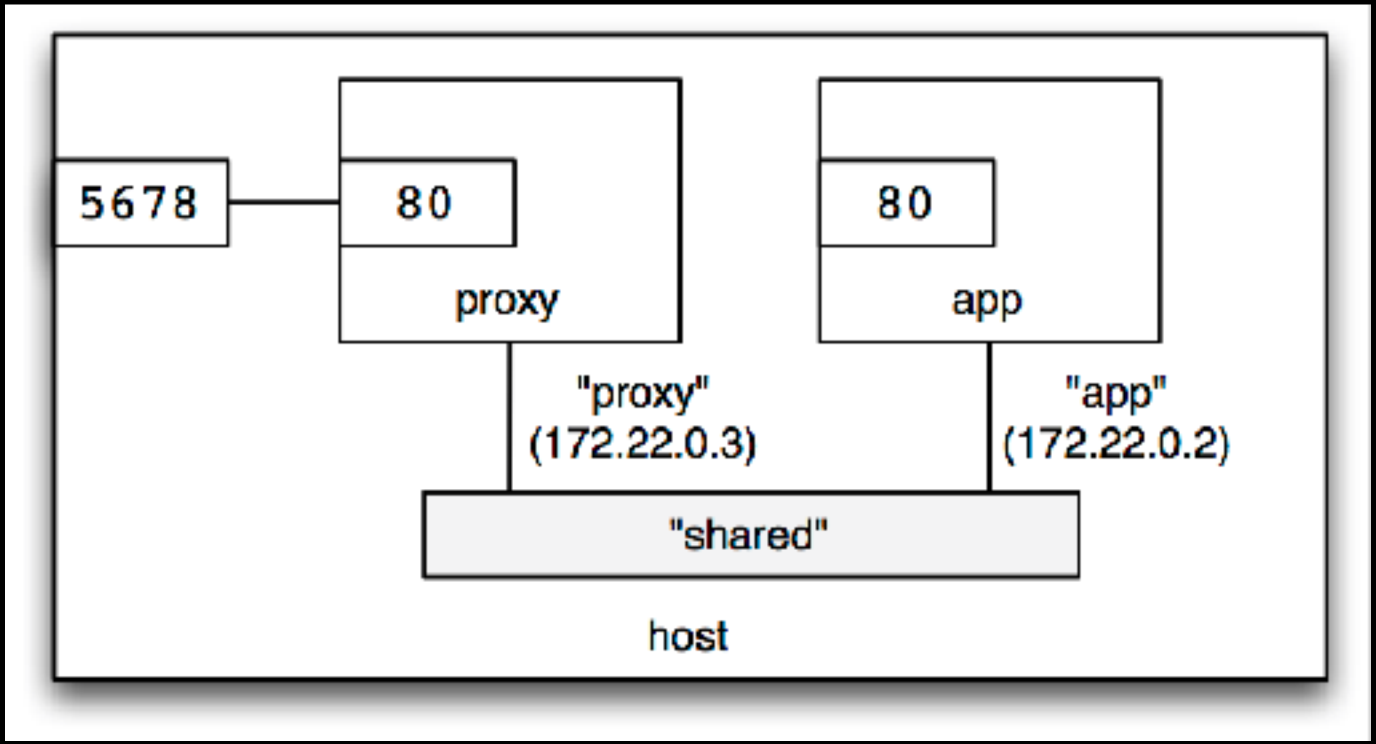
Attach some containers

```
$ docker rm -f app proxy
$ docker run -d —name app —net shared jasonxia/node-hello-world
$ docker run -d —name proxy —net shared -p 5678:80 jasonxia/nginx-proxy
```

Docker Network

Explore the network

```
$ docker network inspect shared
```

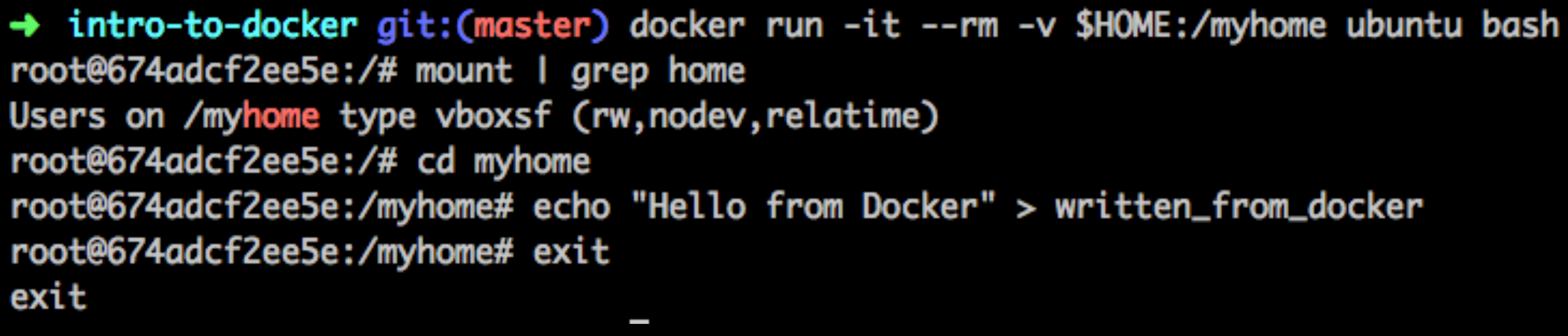


Inspect network

```
$ docker run -it —rm —net shared busybox
/ # nslookup app
```

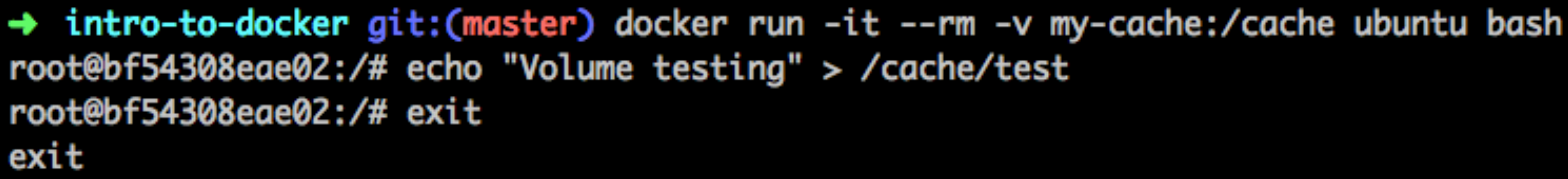
Volumes

Mount your home directory from the “host”:



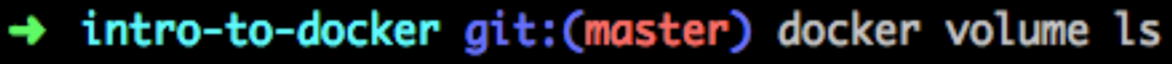
```
➜  intro-to-docker git:(master) docker run -it --rm -v $HOME:/myhome ubuntu bash
root@674adcf2ee5e:/# mount | grep home
Users on /myhome type vboxsf (rw,nodev,relatime)
root@674adcf2ee5e:/# cd myhome
root@674adcf2ee5e:/myhome# echo "Hello from Docker" > written_from_docker
root@674adcf2ee5e:/myhome# exit
exit
```

Named cache volume:

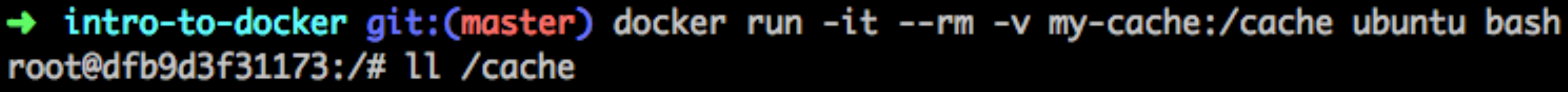


```
➜  intro-to-docker git:(master) docker run -it --rm -v my-cache:/cache ubuntu bash
root@bf54308eae02:/# echo "Volume testing" > /cache/test
root@bf54308eae02:/# exit
exit
```

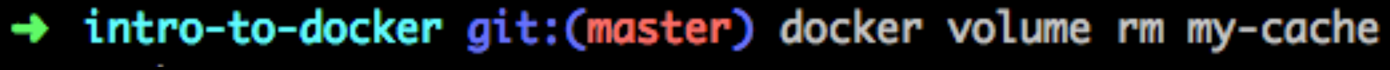
Check the volume created and use it again:



```
➜  intro-to-docker git:(master) docker volume ls
```



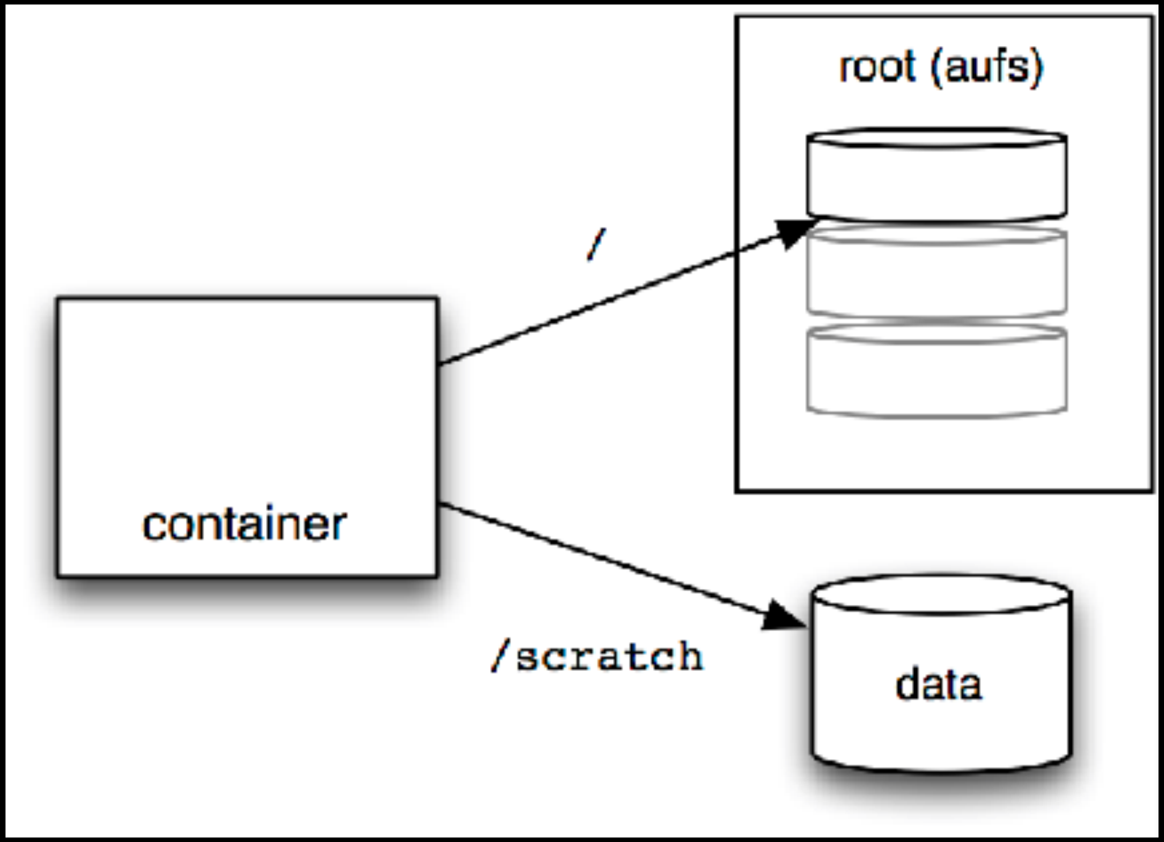
```
➜  intro-to-docker git:(master) docker run -it --rm -v my-cache:/cache ubuntu bash
root@dfb9d3f31173:/# ll /cache
```



```
➜  intro-to-docker git:(master) docker volume rm my-cache
```

Anonymous Volume

```
➜  intro-to-docker git:(master) docker run -it --rm --read-only -v /scratch ubuntu
root@39392d04693b:/# echo "Testing" > /tmp/test
bash: /tmp/test: Read-only file system
root@39392d04693b:/# echo TESTING > /scratch/test
root@39392d04693b:/# exit
```



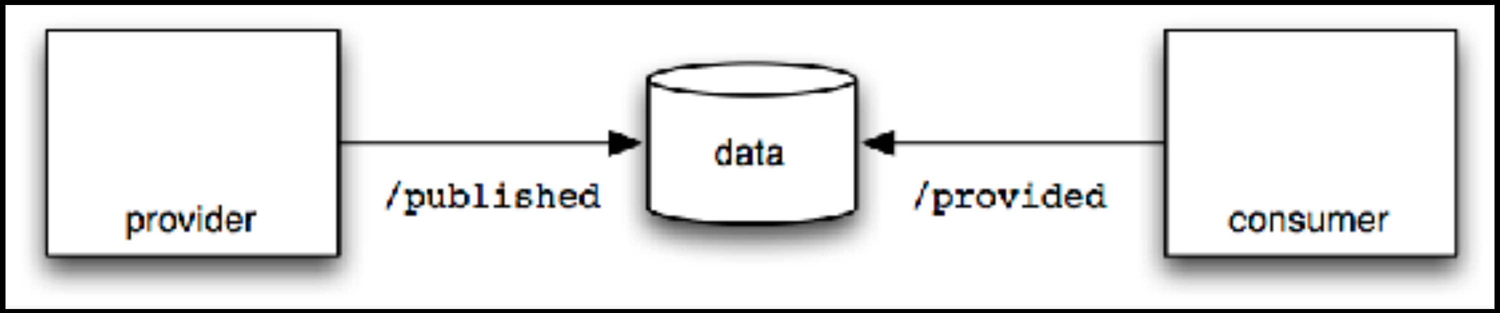
Share Volume

Run a container with a volume:

```
➜  intro-to-docker git:(master) docker run -it --rm --name provider -v data:/published ubuntu
root@5ef3252de64c:/# echo Hello > /published/message
root@5ef3252de64c:/# exit
```

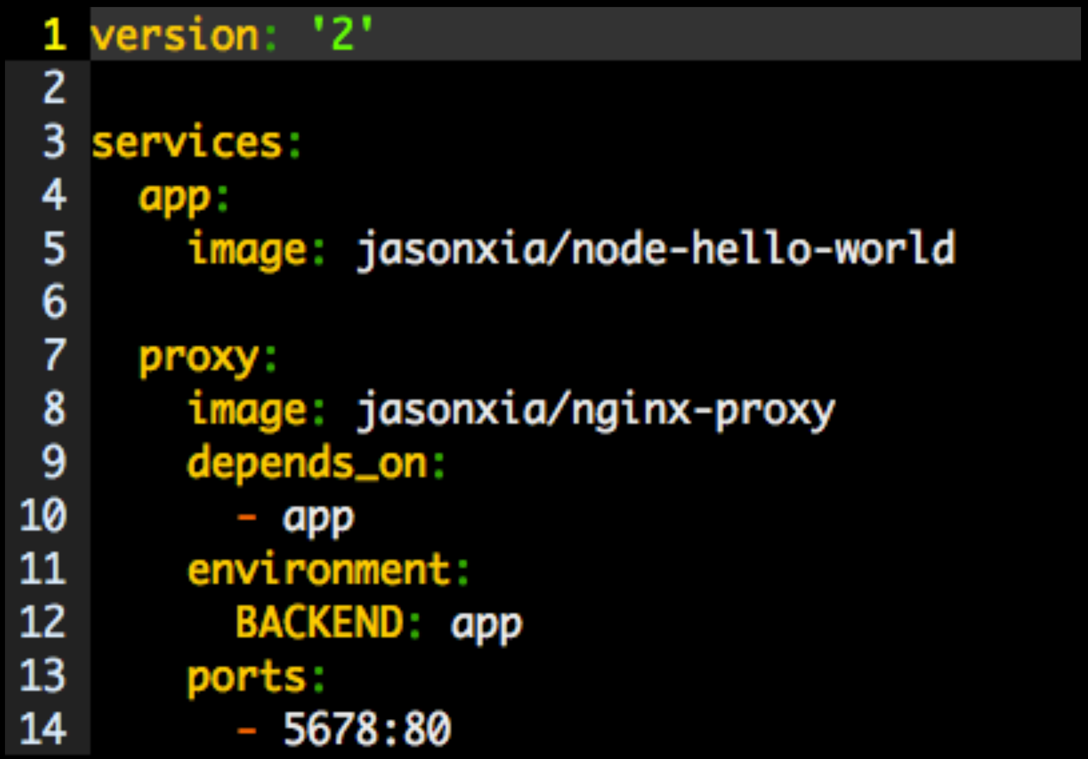
Use the volume from a different container:

```
➜  intro-to-docker git:(master) docker run -it --rm --name consumer -v data:/provided:ro ubuntu
root@b7db03ecbc43:/# cat /provided/message
Hello
root@b7db03ecbc43:/# exit
```

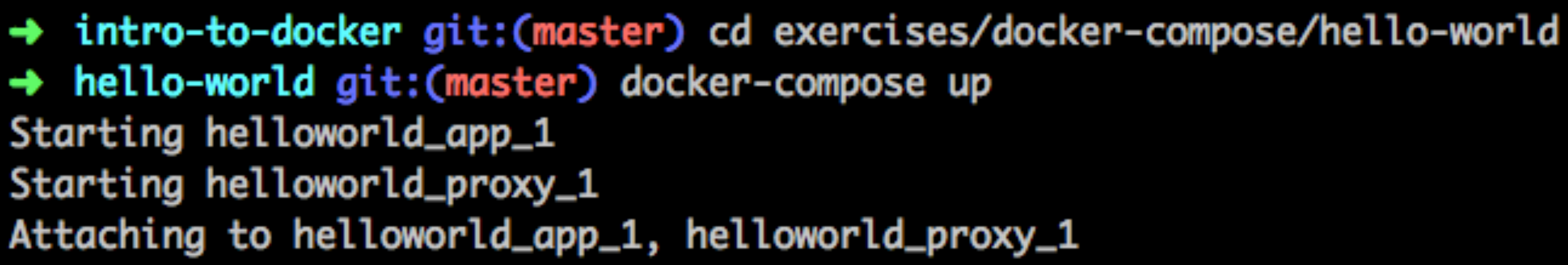


Docker Compose

Hello World app with Nignx proxy example:

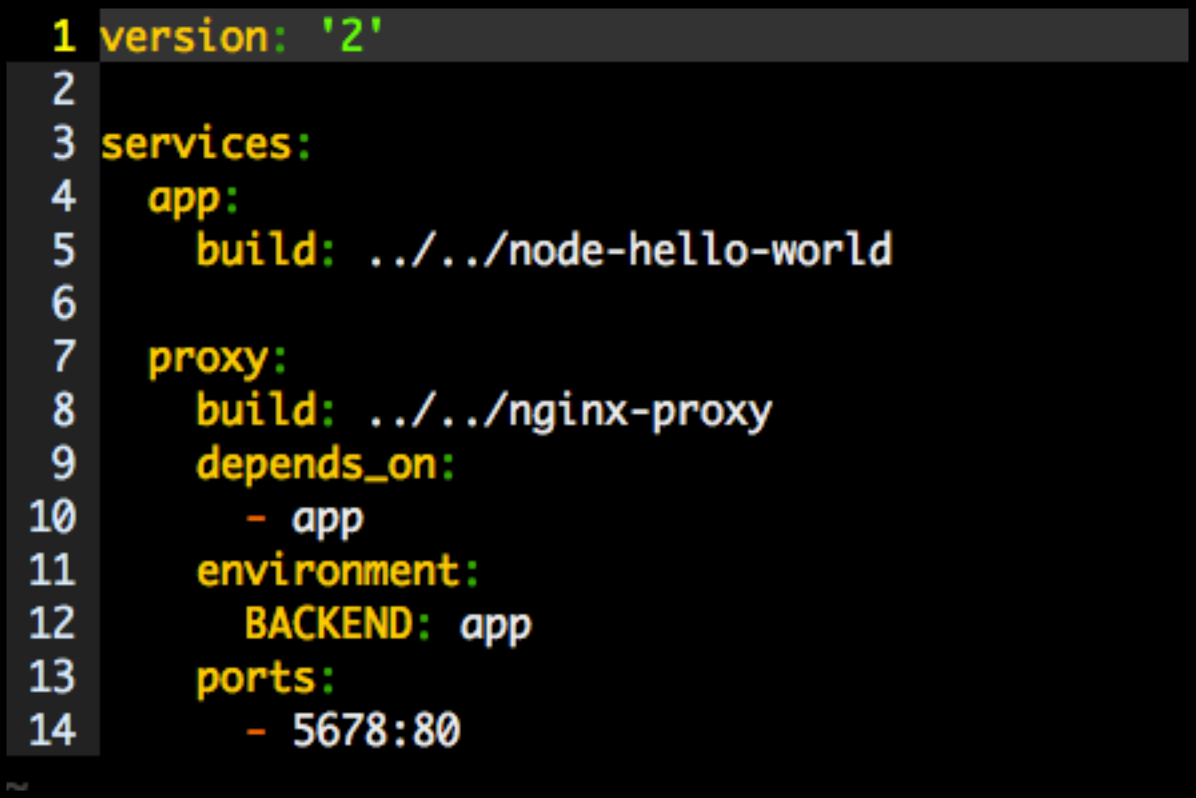


```yaml
version: '2'

services:
  app:
    image: jasonxia/node-hello-world

  proxy:
    image: jasonxia/nginx-proxy
    depends_on:
      - app
    environment:
      BACKEND: app
    ports:
      - 5678:80
```

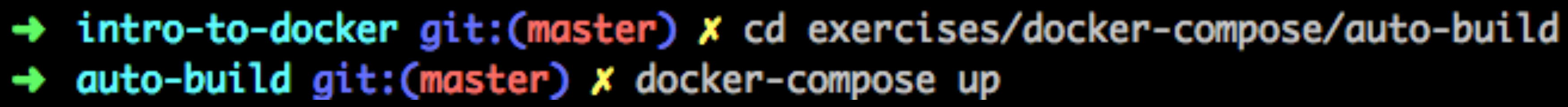


```
➜  intro-to-docker git:(master) cd exercises/docker-compose/hello-world
➜  hello-world git:(master) docker-compose up
Starting helloworld_app_1
Starting helloworld_proxy_1
Attaching to helloworld_app_1, helloworld_proxy_1
```

Docker Compose - Autobuild

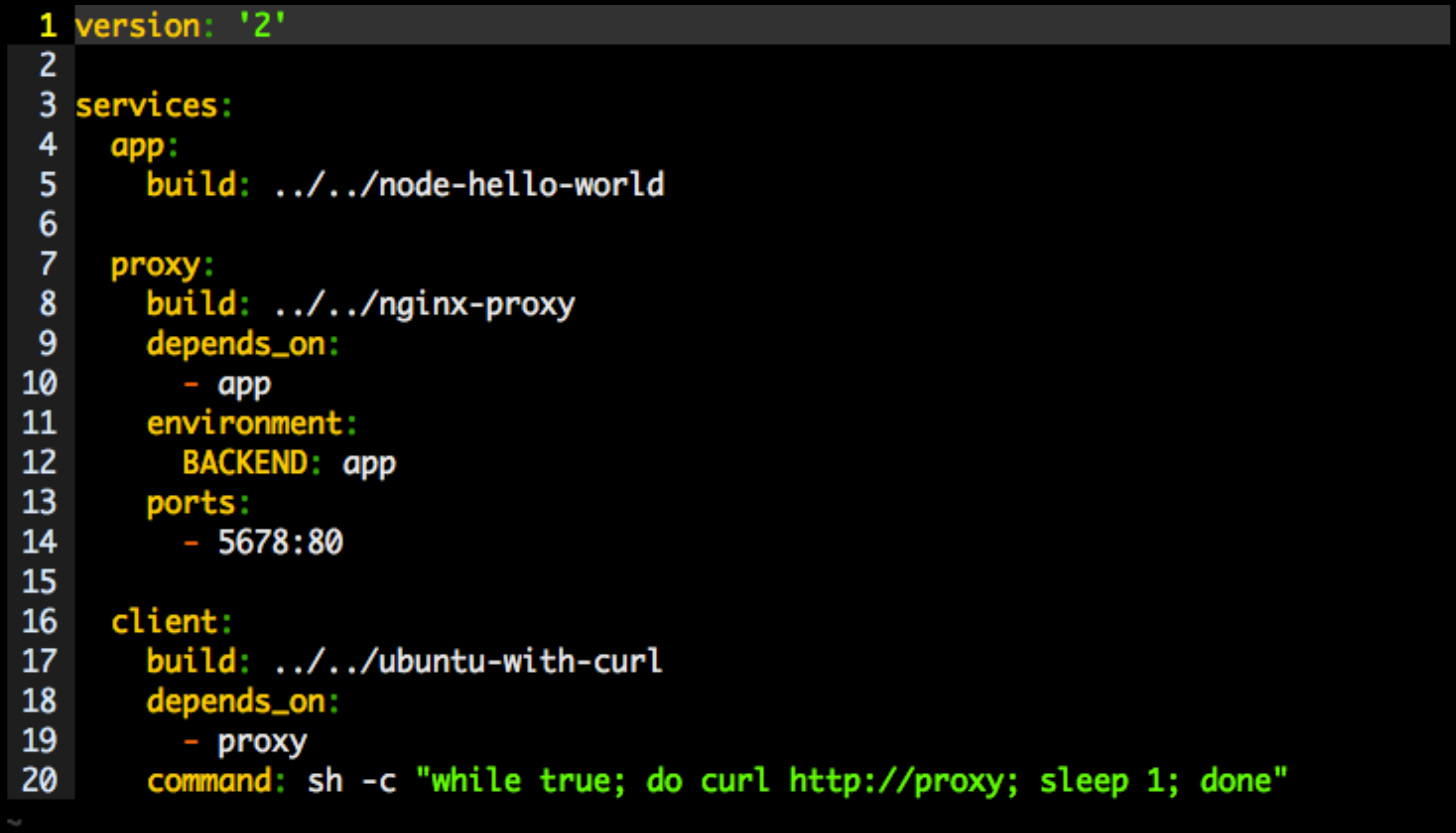


```yaml
version: '2'

services:
  app:
    build: ../../node-hello-world

  proxy:
    build: ../../nginx-proxy
    depends_on:
      - app
    environment:
      BACKEND: app
    ports:
      - 5678:80
```



```
➜  intro-to-docker git:(master) ✗ cd exercises/docker-compose/auto-build
➜  auto-build git:(master) ✗ docker-compose up
```

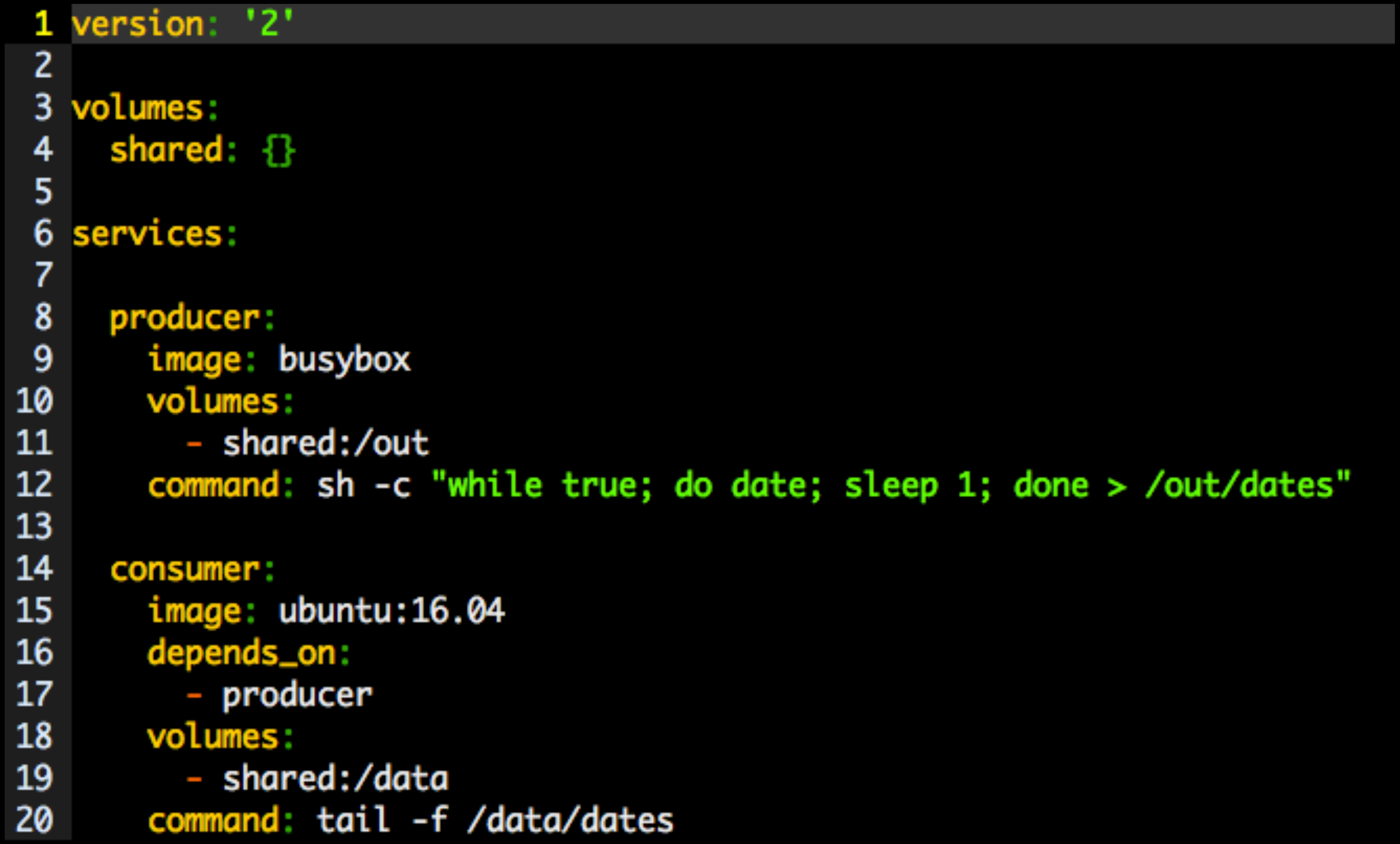
Docker Compose - with Client



```yaml
version: '2'

services:
  app:
    build: ../../node-hello-world

  proxy:
    build: ../../nginx-proxy
    depends_on:
      - app
    environment:
      BACKEND: app
    ports:
      - 5678:80

  client:
    build: ../../ubuntu-with-curl
    depends_on:
      - proxy
    command: sh -c "while true; do curl http://proxy; sleep 1; done"
```

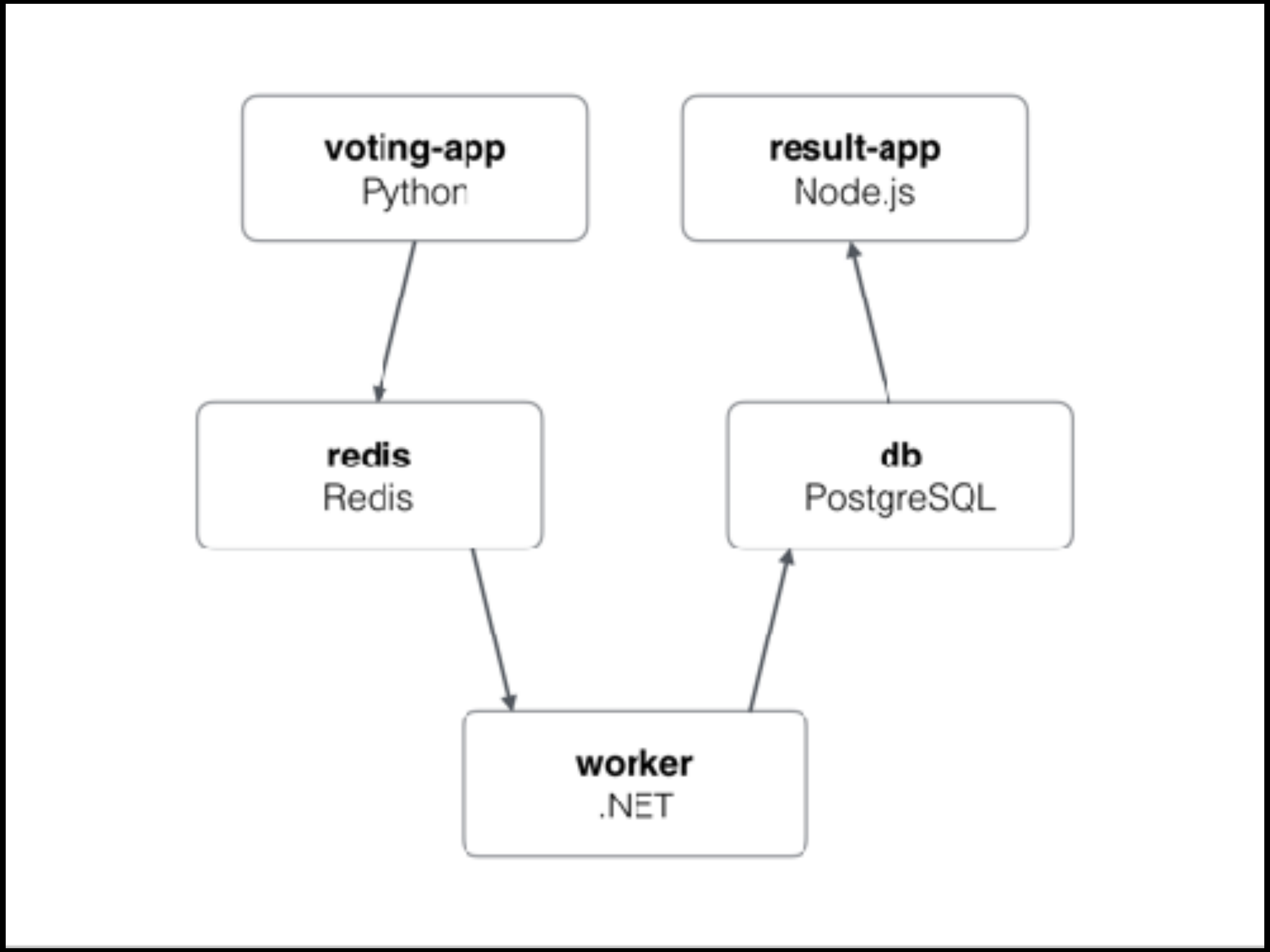
```
$ docker-compose run —rm client

$ docker-compose run —rm client bash
```

Docker Compose - Share Volume



```yaml
version: '2'

volumes:
  shared: {}

services:

  producer:
    image: busybox
    volumes:
      - shared:/out
    command: sh -c "while true; do date; sleep 1; done > /out/dates"

  consumer:
    image: ubuntu:16.04
    depends_on:
      - producer
    volumes:
      - shared:/data
    command: tail -f /data/dates
```

Example



```
$ cd exercises/example-voting-app
$ docker-compose up
```

Thanks