docker

@somkiat

Learning

Introduction

Basic of Docker

Building containers

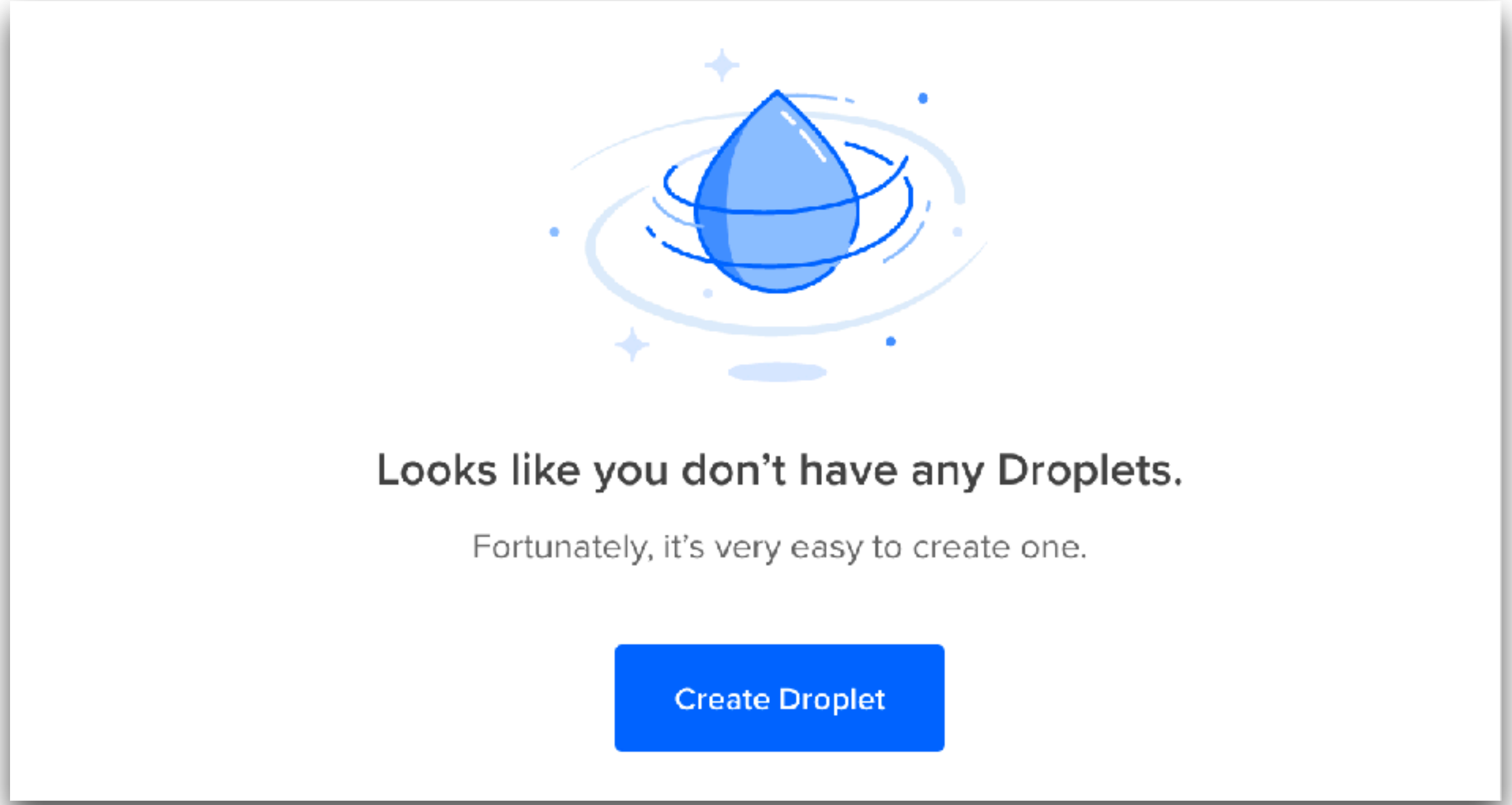
Running web apps with Docker

Docker automation

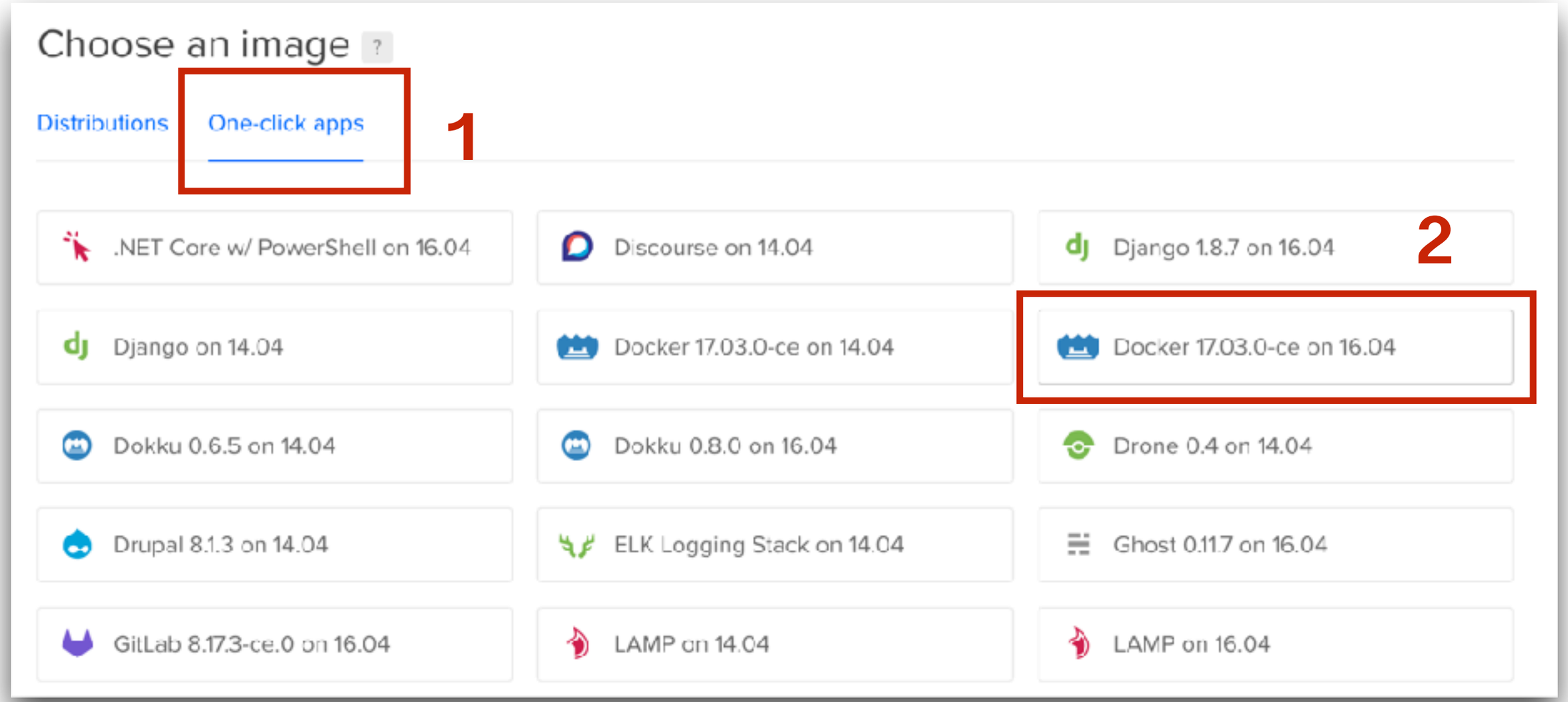
Introduction

Installation

Digital Ocean



Digital Ocean



Access via ssh

$ssh root@<ip>

Verify

$docker version

```
Client:
 Version:      17.03.0-ce
 API version:  1.26
 Go version:   go1.7.5
 Git commit:   60ccb22
 Built:        Thu Feb 23 11:02:43 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.03.0-ce
 API version:  1.26 (minimum version 1.12)
 Go version:   go1.7.5
 Git commit:   60ccb22
 Built:        Thu Feb 23 11:02:43 2017
 OS/Arch:      linux/amd64
 Experimental: false
```

Hello docker

$docker run hello-world

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
78445dd45222: Pull complete
Digest: sha256:c5515758d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7
Status: Downloaded newer image for hello-world:latest

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.
 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://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

Echo with docker

```
$docker run ubuntu echo "Hello World"
```

Basic of Docker

Docker objects

Image

Container

Dockerfile

Registry

Docker container

Container (content layer)

Image (Init layer)

Example

MySQL 8.0

Debian (jessie)

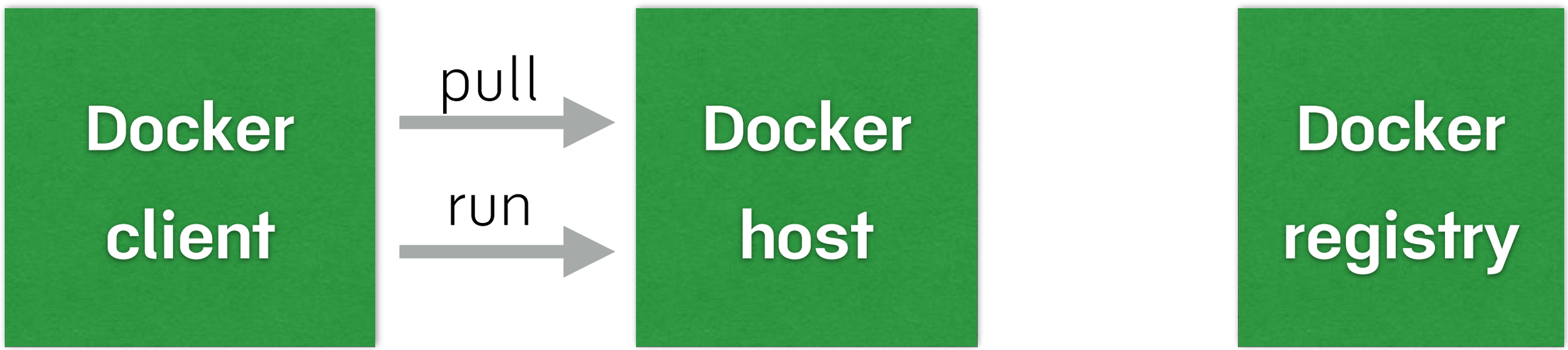
How docker works ?

Docker client

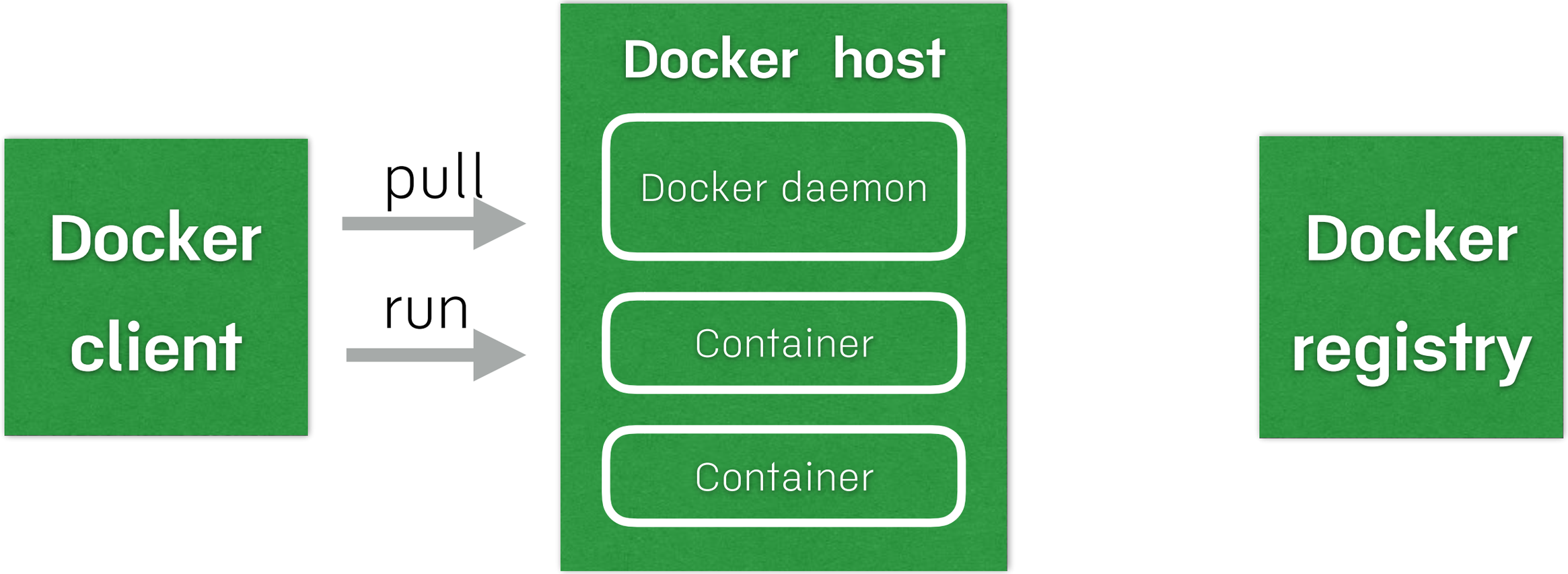
Docker host

Docker registry

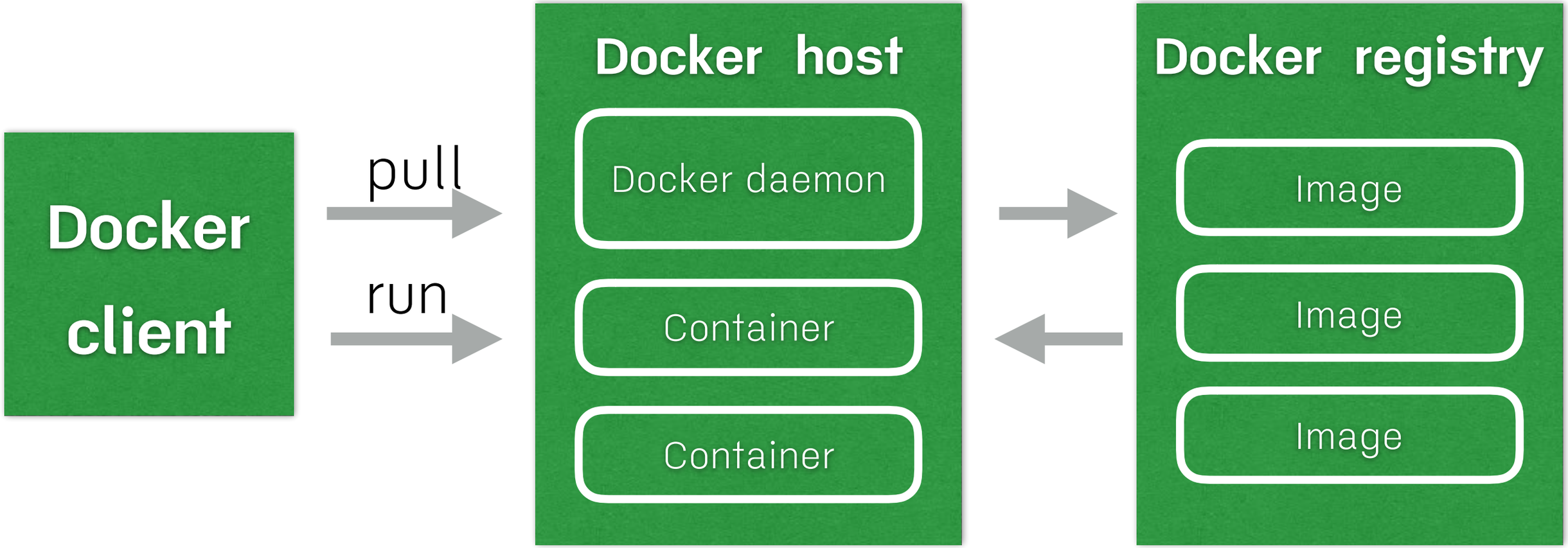
How docker works ?



How docker works ?



How docker works ?



Docker commands

List all images

$docker images

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
alpine              latest              4a415e366388        2 weeks ago         3.99 MB
ubuntu              latest              0ef2e08ed3fa        2 weeks ago         130 MB
ubuntu              xenial              0ef2e08ed3fa        2 weeks ago         130 MB
ubuntu              trusty              7c09e61e9035        2 weeks ago         188 MB
hello-world         latest              48b5124b2768        2 months ago        1.84 kB
```

Docker image command

$docker image

```
Usage:  docker image COMMAND

Manage images

Options:
      --help   Print usage

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
```

List all containers

$docker ps -a

Docker container command

$docker container

```
Usage:  docker container COMMAND

Manage containers

Options:
      --help   Print usage

Commands:
  attach      Attach to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  inspect     Display detailed information on one or more containers
  kill        Kill one or more running containers
  logs        Fetch the logs of a container
  ls          List containers
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  prune       Remove all stopped containers
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  run         Run a command in a new container
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes

Run 'docker container COMMAND --help' for more information on a command.
```

List all containers

$docker container ls -a

Remove all containers

```
$docker stop $(docker ps -a -q)
$docker rm $(docker ps -a -q)
```

Remove all stepped container

$docker container prune

Container run process

Foreground

Interactive

Background

Let’s start (old)

$docker run

Let’s start (new)

$docker container run

https://docs.docker.com/engine/reference/commandline/container_run/#options

Foreground

$docker container run nginx

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
693502eb7dfb: Pull complete
6decb850d2bc: Pull complete
c3e19f087ed6: Pull complete
Digest: sha256:52a189e49c0c797cfc5cbfe578c68c225d160fb13a42954144b29af3fe4fe335
Status: Downloaded newer image for nginx:latest
```

Background

$docker container run -d nginx

Interactive

```
$docker container run -it nginx bash
```

Delete all image

$docker container ?

Remove after exited

```
$docker container run --rm nginx
$docker container run --rm -d nginx
$docker container run --rm -it nginx bash
```

Start nginx with name

$docker container run --name **hello-nginx** -d nginx

Remove container

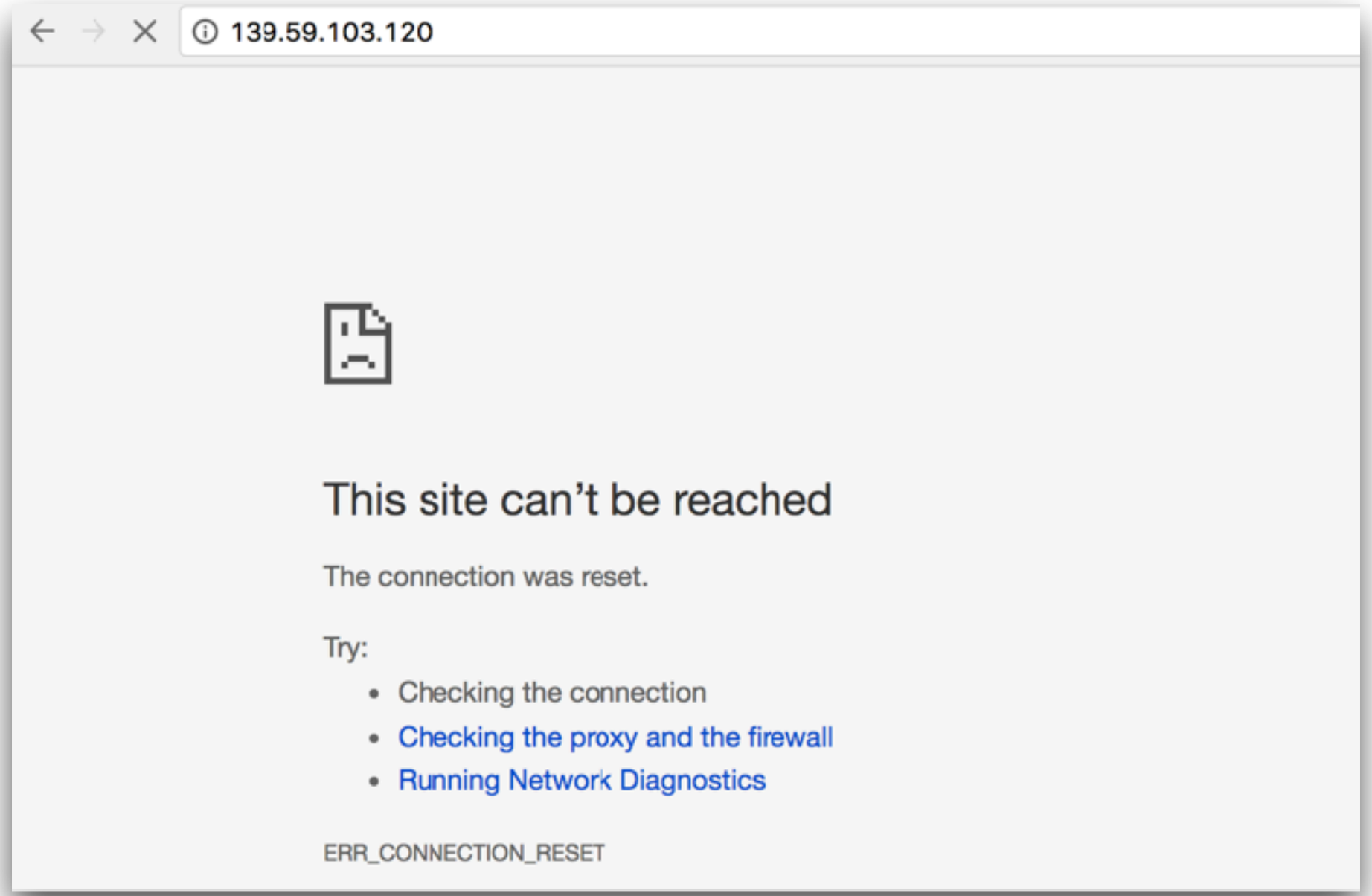
$docker container stop **hello-nginx**

$docker container rm **hello-nginx**

Start nginx again

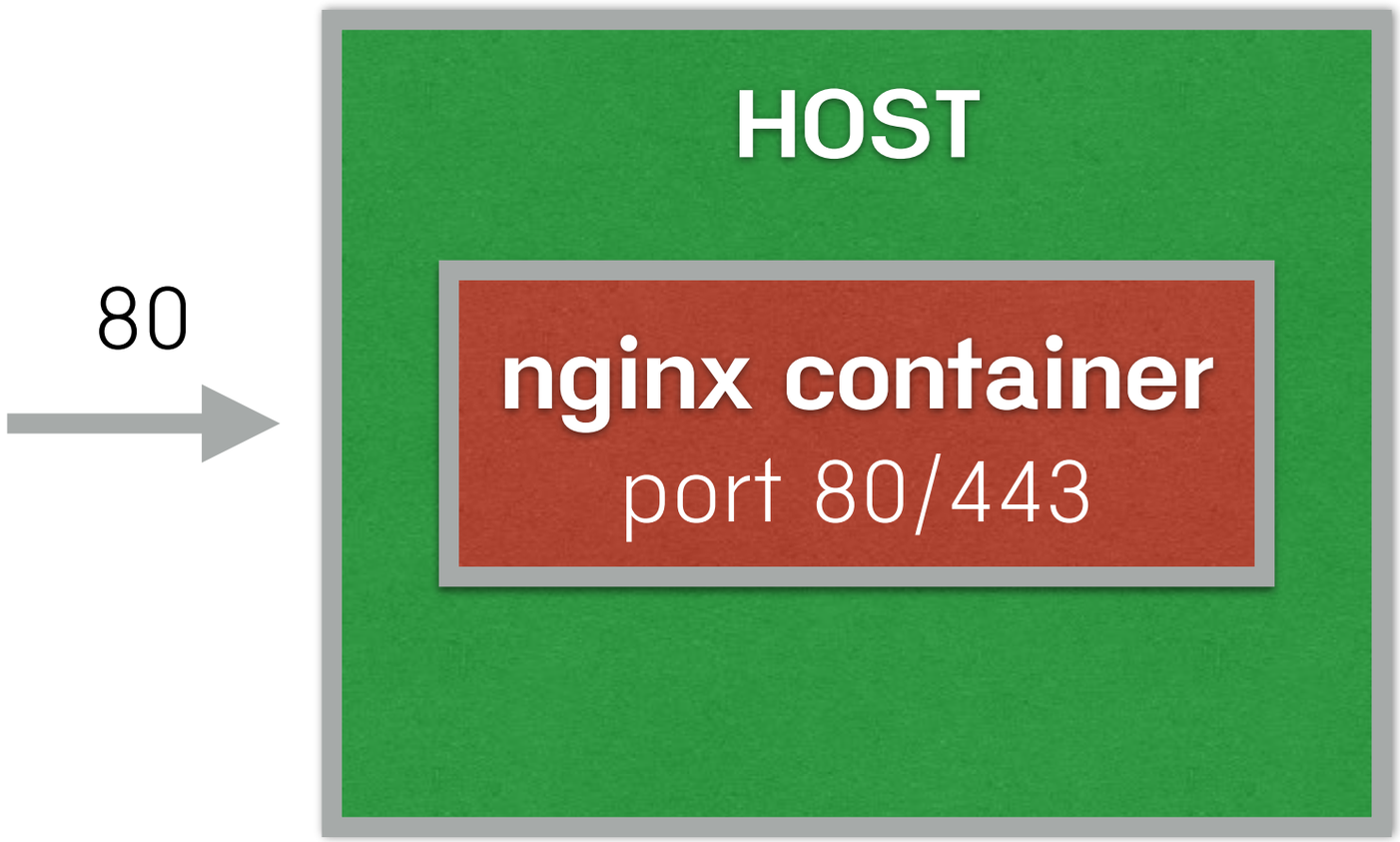
$docker container run --name **hello-nginx** -d nginx

Try to access with port 80

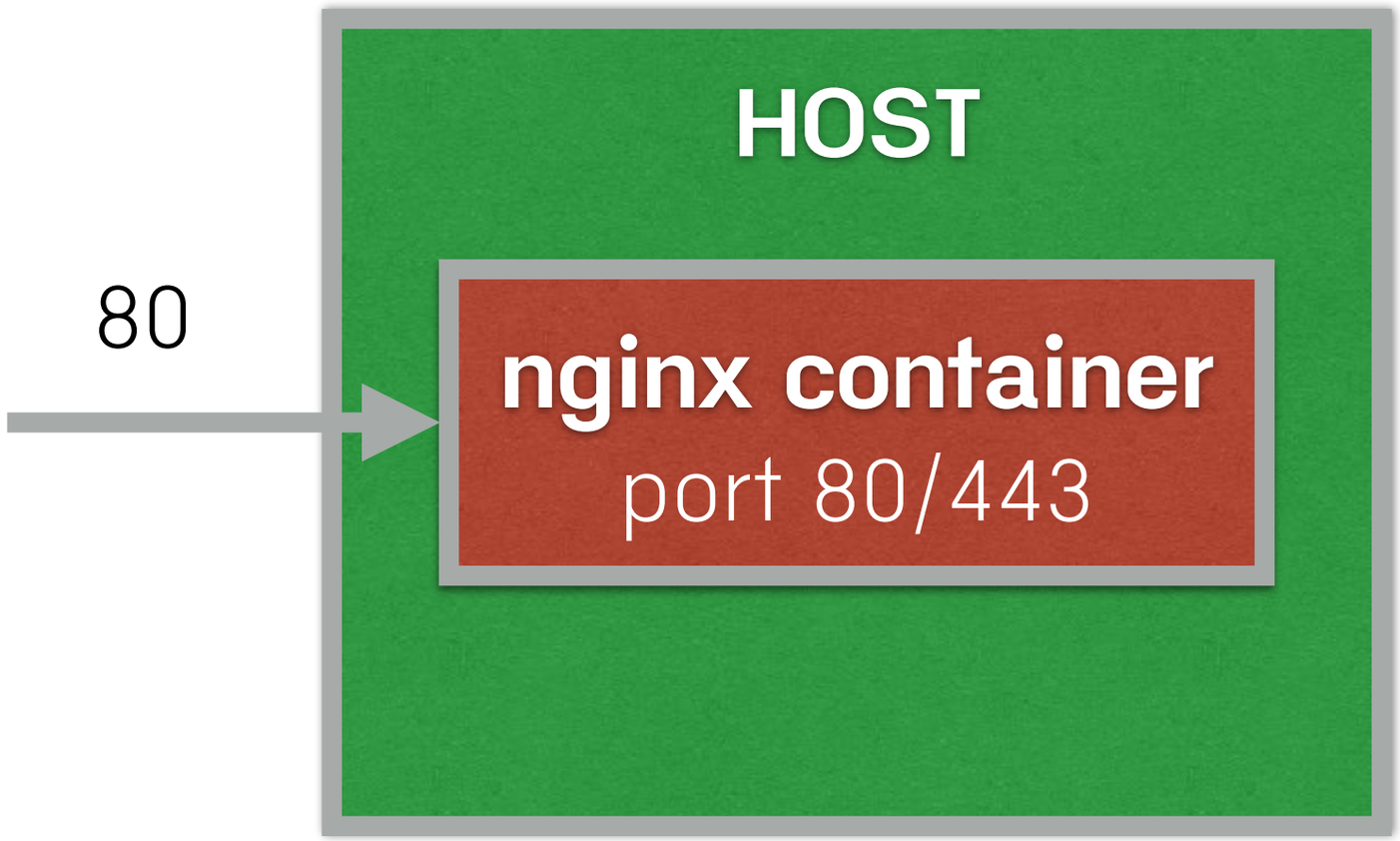


Why ?

Docker networking



Docker networking



Start with forward port

$docker container run --name **hello-nginx** -d -p 80:80 nginx

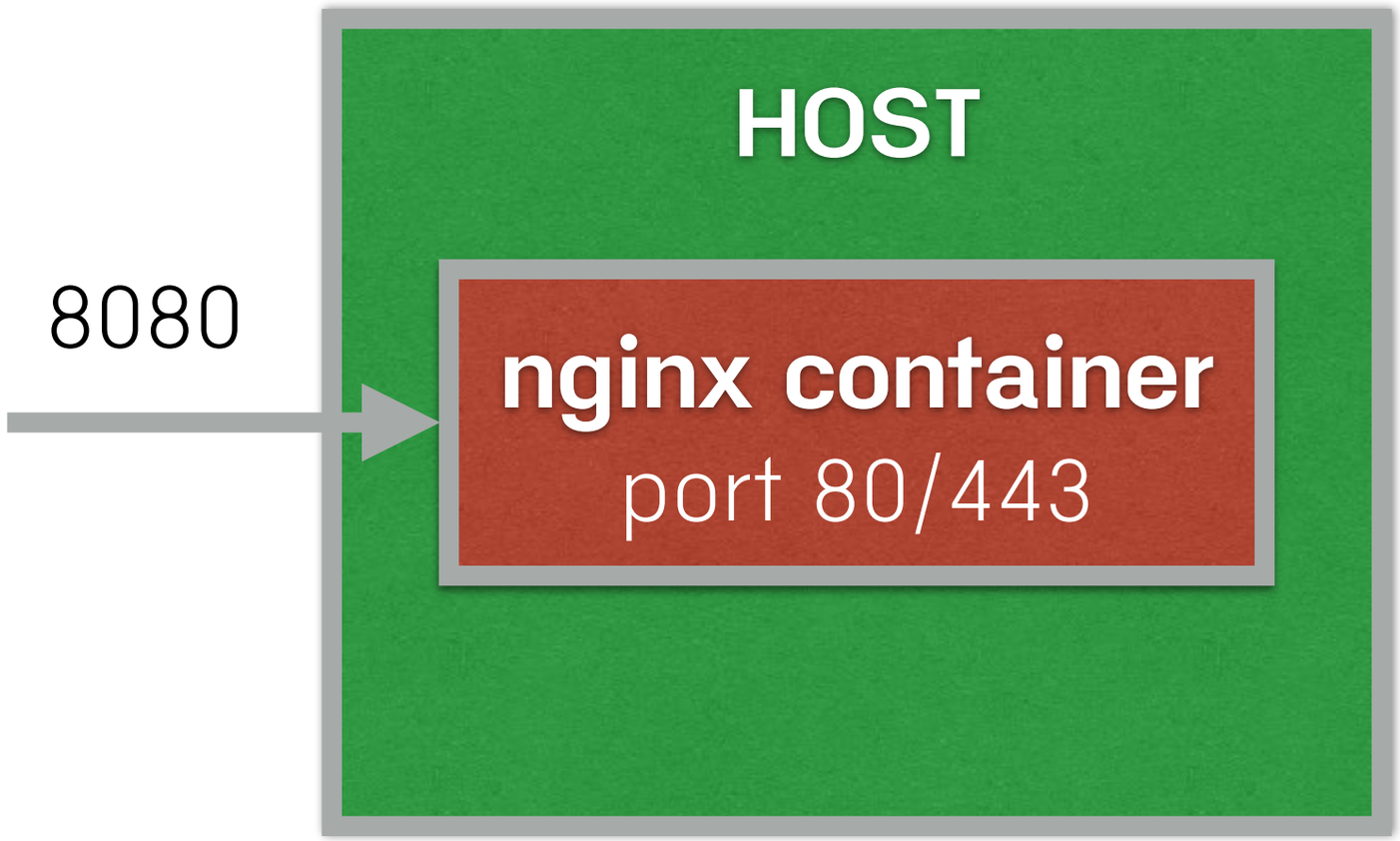
Try to access with port 80

Start with forward port

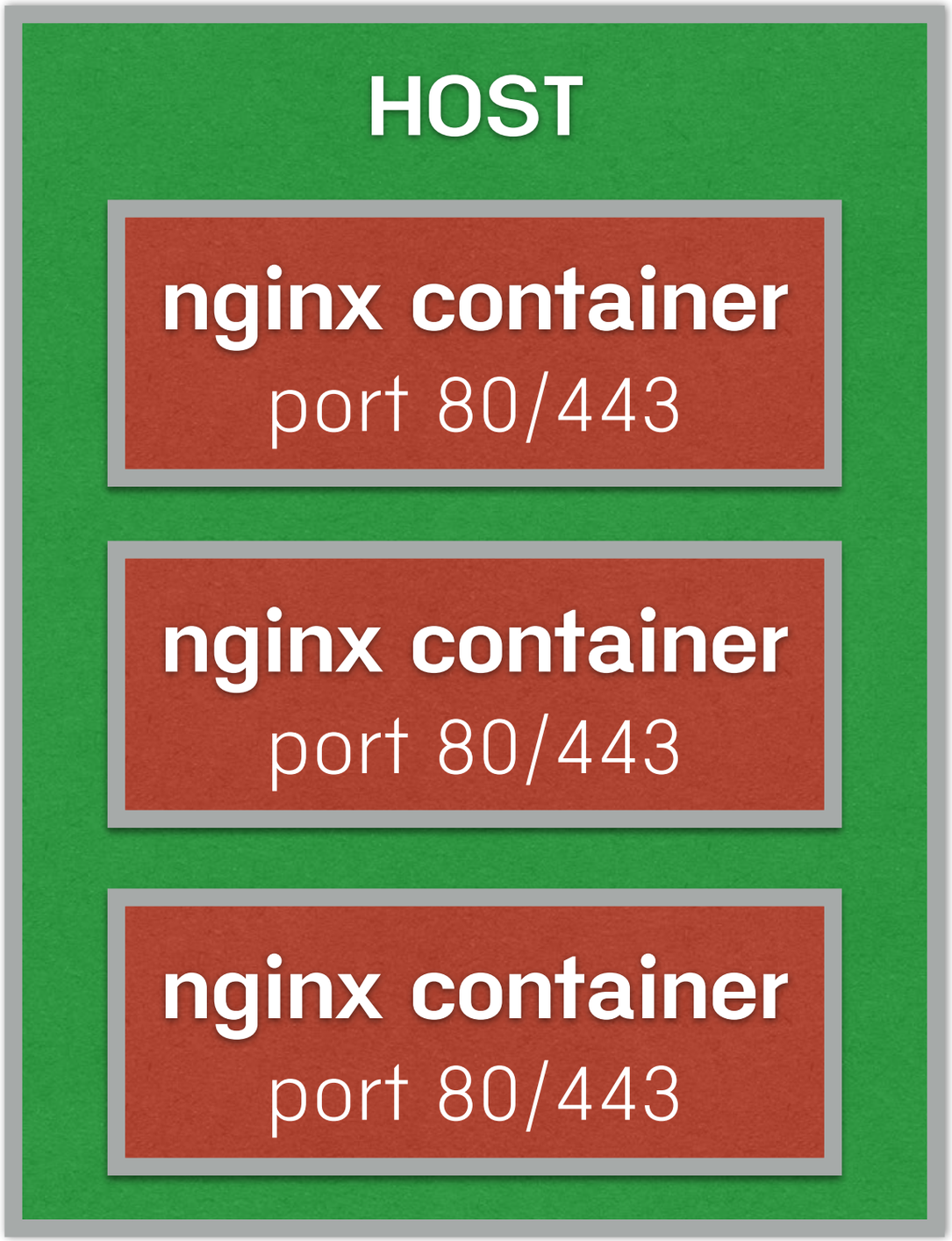
$docker container run --name **hello-nginx** -d

-p 8080:80 nginx

Docker networking



More containers



Building containers

Demo app

Docker with MySQL

Search

Explore Help Sign up Sign in

OFFICIAL REPOSITORY

mysql ☆

Last pushed: 19 days ago

Repo Info Tags

Short Description

MySQL is a widely used, open-source relational database management system (RDBMS).

Docker Pull Command

```
docker pull mysql
```

Full Description

Supported tags and respective `Dockerfile` links

- `8.0.0`, `8.0`, `8` (8.0/Dockerfile)
- `5.7.17`, `5.7`, `5`, `latest` (5.7/Dockerfile)
- `5.6.35`, `5.6` (5.6/Dockerfile)
- `5.5.54`, `5.5` (5.5/Dockerfile)

https://hub.docker.com/_/mysql/

Start MySQL container

```
$docker run  -d --name  mydb  \
  -p 3306:3306 \
  -e MYSQL_ROOT_PASSWORD=123456 \
  -e MYSQL_DATABASE=demo  \
  mysql:latest
```

https://hub.docker.com/_/mysql/

How to test ?

$mysql -uroot -p123456

How to test ?

```
$mysql -h<ip> -uroot -p123456
```

Container inspect

$docker container inspect <id/name>

```
"SandboxKey": "/var/run/docker/netns/dc2b1c8b3cea",
"SecondaryIPAddresses": null,
"SecondaryIPv6Addresses": null,
"EndpointID": "05c5f7c3d66d9afc106c16f0aa3deee6ad1a561c13da070e3c1650133f7c7f4f",
"Gateway": "172.17.0.1",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"IPAddress": "172.17.0.2",
"IPPrefixLen": 16,
"IPv6Gateway": "",
"MacAddress": "02:42:ac:11:00:02",
```

Working with spring boot

```
$git clone <url>
$cd user-service
$mvn clean package
$java -jar user-service.jar
```

https://github.com/up1/demo-service

Testing

139.59.103.120:9001/user

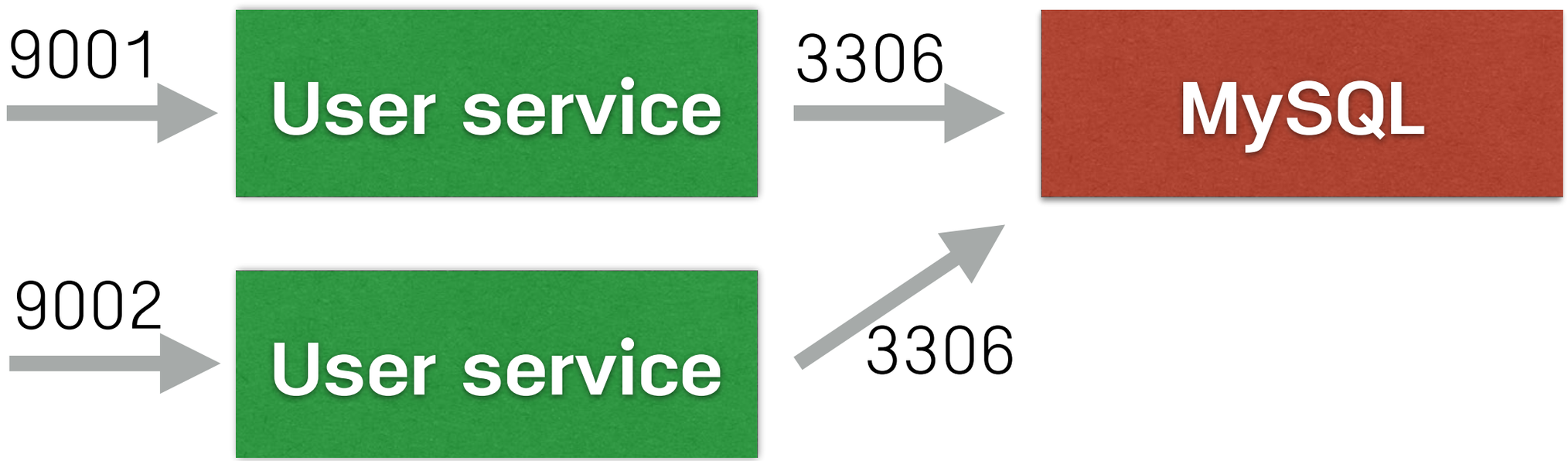
```
{
    id: 1,
    firstname: "somkiat",
    lastname: "pui"
}
```

Enable firewall

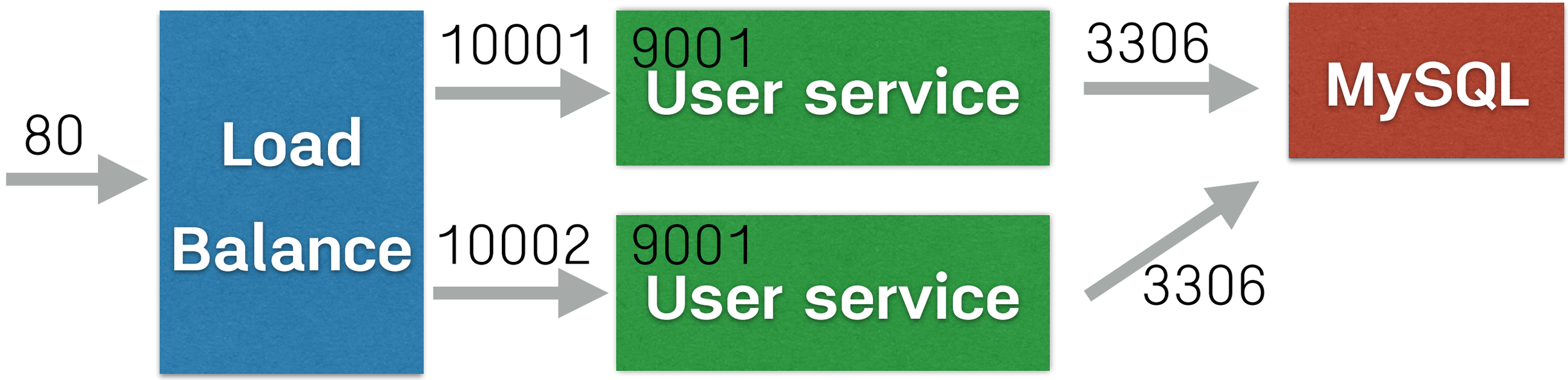
$ufw allow 9001

Next

Demo app



Demo app


80
Load
Balance
10001
9001
User service
10002
9001
User service
3306
3306
MySQL

Running web app

Resources

https://github.com/up1/demo-service