Full Stack for Front-End 

Jem Young

@jemyoung

Senior Software Engineer

FRONT END HAPPY HOUR

Slides

jemyoung.com/fsfe

Part 1

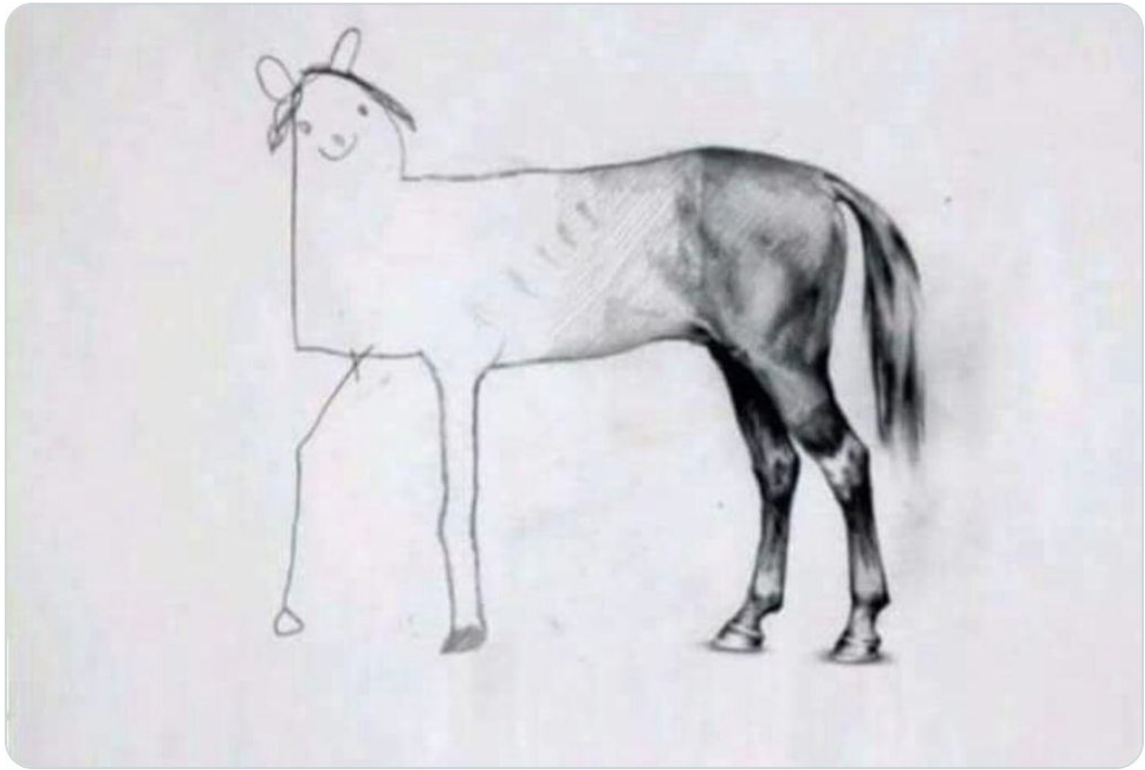
getting started

- “Full Stack”
- Command line
- Shells
- How the internet works
- VIM

What is “Full Stack?”

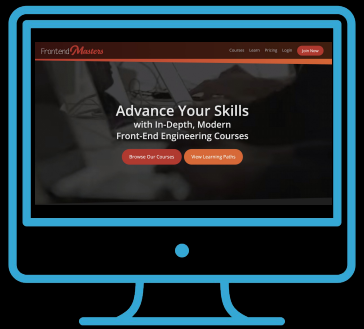
Brian Holt
@holtbt

“Fullstack” developer.



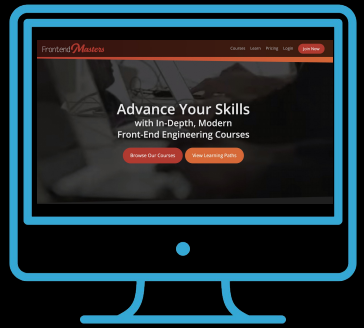
Frontend

Backend



magic?

Frontend



cars

televisions

browsers

desktop apps

appliances

mobile devices

Backend

APIs

data analytics

security

reliability

databases

platform

What is “Full Stack?”

Someone who can build an application from start to finish

Full Stack Engineer

Why Full Stack?

Command Line

Command Line

- Fast
- Consistent
- Easier to automate

Why the command line?

Exercise

Command Line

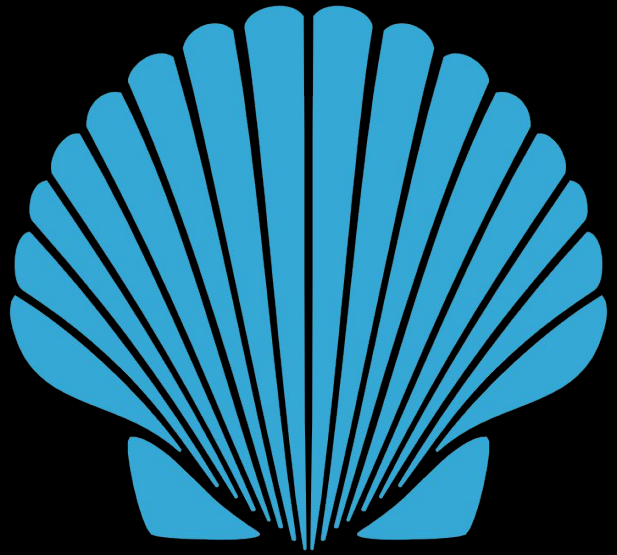
Command Line

- **cd** - change directory
- **ls** - list directory contents
- **pwd** - print working directory
- **mkdir** - make directory
- **rmdir** - remove directory

- **cat** - show file contents
- **man** - command manual
- **less** - show file contents by page
- **rm** - remove file
- **echo** - repeat input

Shells

Shells



Shells

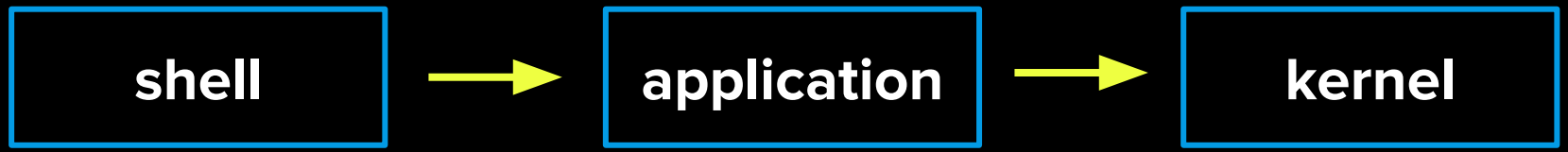
Command interpreter to interface with system

`Shell`

Runs shell applications

`Terminal`

Shells



Show current shell

```
$ echo $0
```

How does the internet work?

How does the internet work?

A system of globally interconnected devices

`Internet`

How does the internet work?

Private internet

`Intranet`

How does the internet work?

Internet Protocol

`IP`

A label assigned to an internet connected device

`IP Address`

How does the internet work?

8.8.8.8

IPv4

2001:4860:4860:8888

IPv6

How does the internet work?

Transmission Control Protocol

TCP

User Datagram Protocol

UDP

Exercise

Ping

```
$ ping twitter.com
```

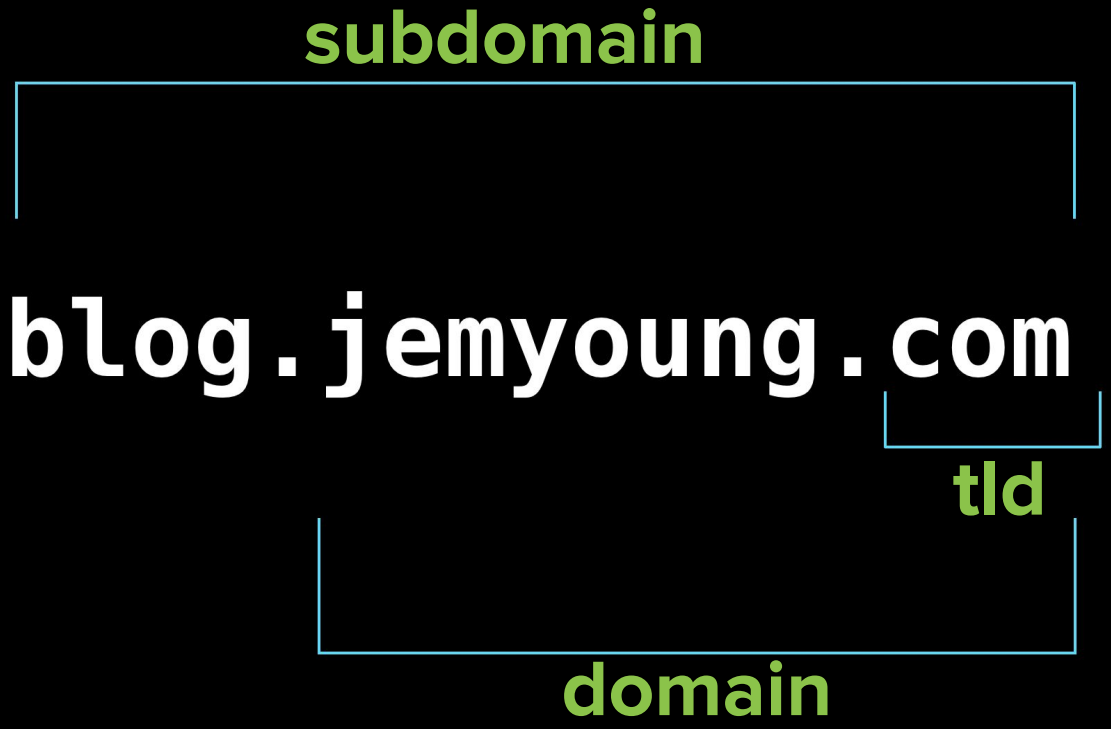
```
$ ping twitter.com
PING twitter.com (104.244.42.193): 56 data bytes
64 bytes from 104.244.42.193: icmp_seq=0 ttl=57 time=10.650 ms
64 bytes from 104.244.42.193: icmp_seq=1 ttl=57 time=16.752 ms
64 bytes from 104.244.42.193: icmp_seq=2 ttl=57 time=39.176 ms
64 bytes from 104.244.42.193: icmp_seq=3 ttl=57 time=11.582 ms
^C
--- twitter.com ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 10.650/19.540/39.176/11.573 ms
```

How does the internet work?

Domain Name System

DNS

How does the internet work?



How does the internet work?

jemyoung.com

====>

23.23.185.61

How does the internet work?

Map domains to IP addresses

`Nameservers`

Exercise

Trace Route

Traceroute manual

```
$ man traceroute
```

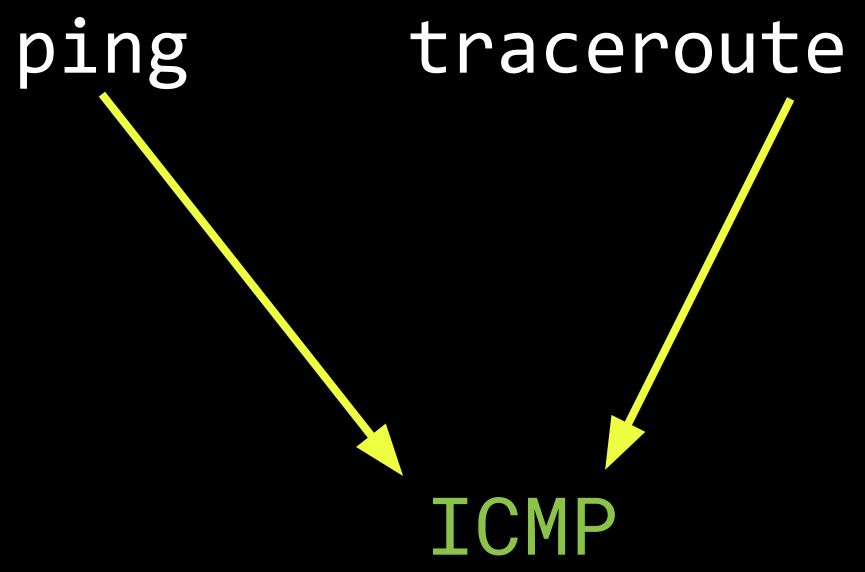
Run traceroute

```
$ traceroute google.com
```

```
$ traceroute google.com
traceroute to google.com (172.217.164.110), 64 hops max, 52 byte packets
 1  gateway (172.22.216.1)  3.586 ms  6.615 ms  6.322 ms
 2  69.53.241.253 (69.53.241.253)  6.514 ms
    69.53.241.245 (69.53.241.245)  9.761 ms  9.872 ms
 3  69.53.254.249 (69.53.254.249)  4.700 ms  10.539 ms
    69.53.254.253 (69.53.254.253)  3.938 ms
 4  23.246.0.210 (23.246.0.210)  3.492 ms  71.771 ms
    23.246.0.208 (23.246.0.208)  4.796 ms
 5  23.246.0.23 (23.246.0.23)  10.490 ms
    23.246.0.21 (23.246.0.21)  10.696 ms  9.592 ms
 6  4.7.18.29 (4.7.18.29)  3.681 ms  4.621 ms  5.335 ms
 7  ae-2-7.edge1.sanjose3.level3.net (4.69.209.169)  4.229 ms
    ae-1-8.edge1.sanjose3.level3.net (4.69.209.165)  6.249 ms
    108.170.242.225 (108.170.242.225)  5.223 ms
 8  209.85.252.251 (209.85.252.251)  6.190 ms
    72.14.223.91 (72.14.223.91)  5.705 ms
    209.85.252.251 (209.85.252.251)  6.324 ms
 9  108.170.242.225 (108.170.242.225)  7.904 ms
    sfo03s18-in-f14.1e100.net (172.217.164.110)  5.659 ms
    108.170.243.1 (108.170.243.1)  7.450 ms
```

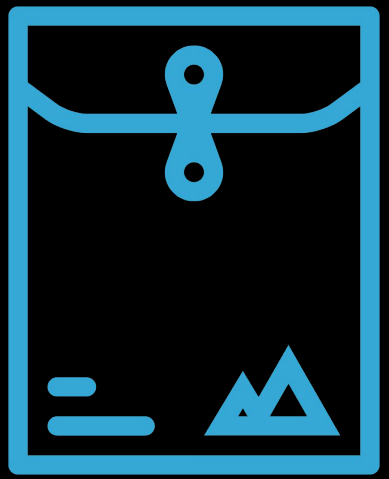
```
$ traceroute frontendmasters.com
```

How does the internet work?



Internet Control
Message Protocol

How does the internet work?



Packet

(Again) How does the internet work?


domain
IP
TCP
?

VIM

VIM

Vi Improved

VIM

VIM

insert mode	Text editing
command mode	Primary mode
last line mode	Searching, saving, exiting

VIM

insert mode	`i`
command mode	`ESC`
last line mode	`:`

HOW TO QUIT VIM

ESC :q!

VIM

https://linuxmoz.com/vi-commands-cheat-sheet/

Vi Page Down

Ctrl+F – Vi page down – Moves forward a page **Ctrl+D** – Moves forward half a page

Vi Page Up

Ctrl+B – Vi page up – Moves back a page **Ctrl+U** – Moves backward a half-page

Vi Delete Line

dd – Vi delete line, regardless of the cursors position on the line

D – Deletes all text from the cursor position to the end of the line

dL – Deletes all text from the cursor position to the end of the screen

dG – Deletes all text from the cursor to the EOF

d^ – Deletes all text from the beginning of the line to the cursor

Vi Save & Exit

:q – Vi exit – this will close Vi

:wq – Vi save & exit

: x – Vi exit, and prompts it you want to save on exit.

Shift+ZZ – Alternative way to save and exit Vi

:q! – Exits vi and discards and changes you made

:wq! – Vi Save and exit if you are root and do not have the write bit set for the file you are attempting to write.

http://vim.wikia.com/wiki/Copy,_cut_and_paste

Exercise

VIM

```
$ man vi

$ vi
```

Exercise

Creating and
editing in VIM

```
$ vi temp
```

Part 2

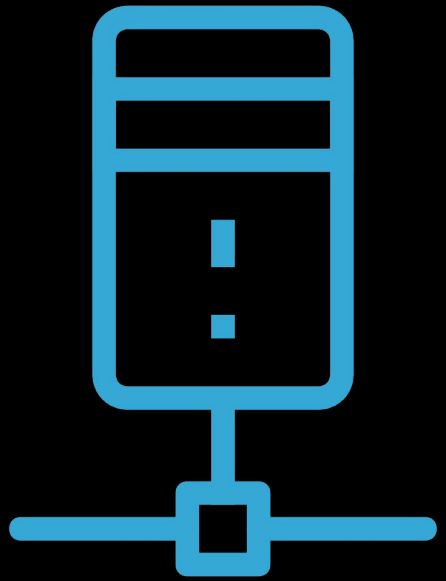
servers ftw

- Servers
- The cloud
- VPS
- SSH
- Buying a domain

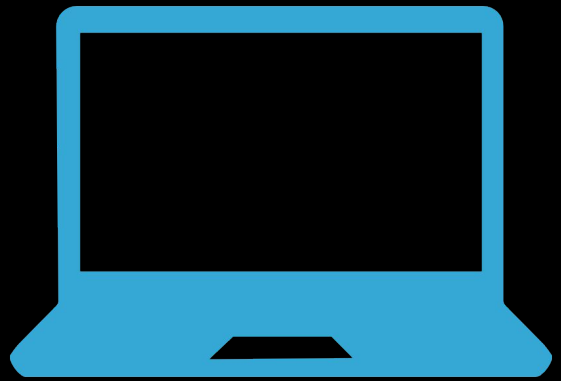
Servers

Servers

What does a server do?



Servers



Exercise

Let’s make a server!

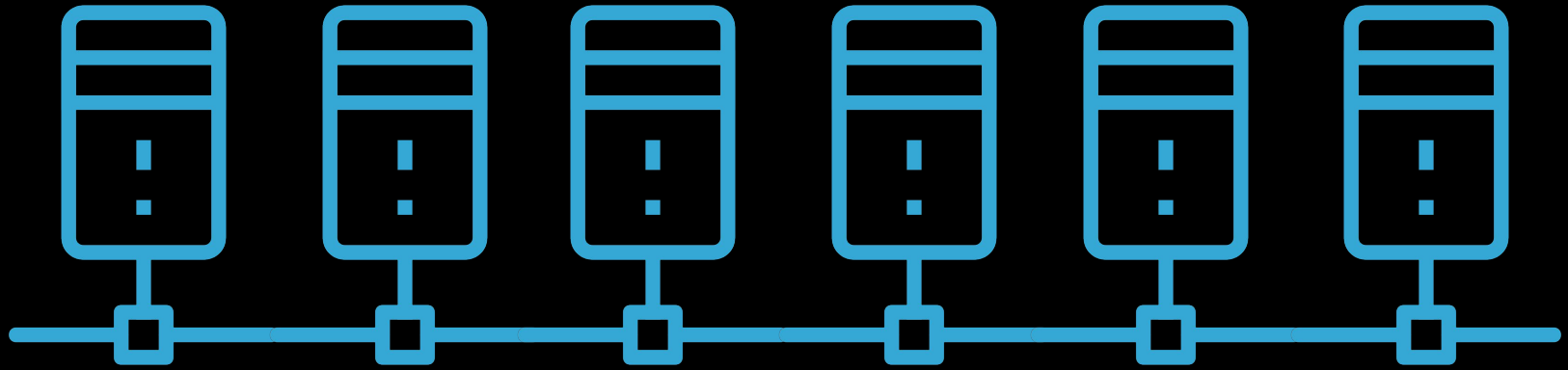
```
$ vi simpleServer.js
```

```
 1 const http = require('http');
 2
 3 http.createServer(function (req, res) {
 4   res.write('Hello, World!');
 5   res.end();
 6 }).listen(8080);
 7
 8 console.log('Server started! Listening on port 8080');
 9
10
```

Run NodeJS script

```
$ node simpleServer.js
```

Servers

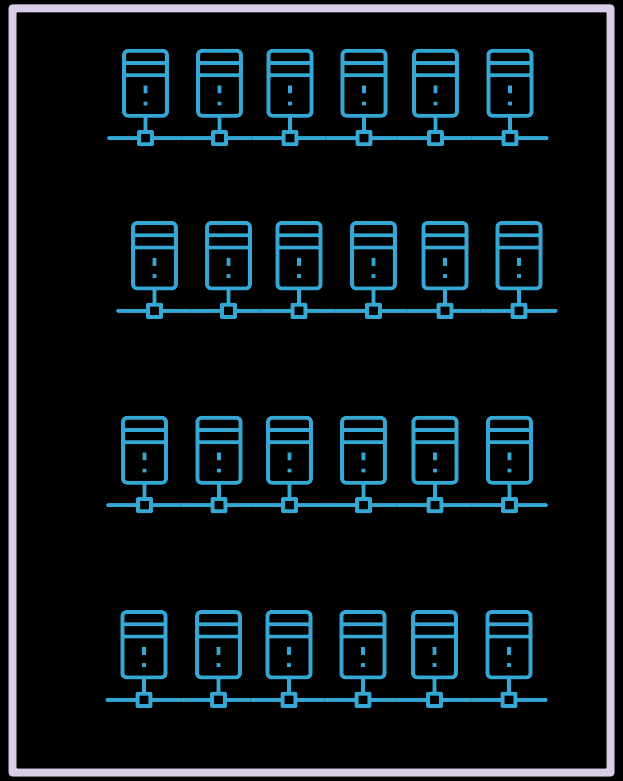


Servers


MAIL SERVERS
WORLD WIDE WEB SERVERS
DATABASE SERVERS
PC DESKTOP
EMERGENCY EXIT
CERN

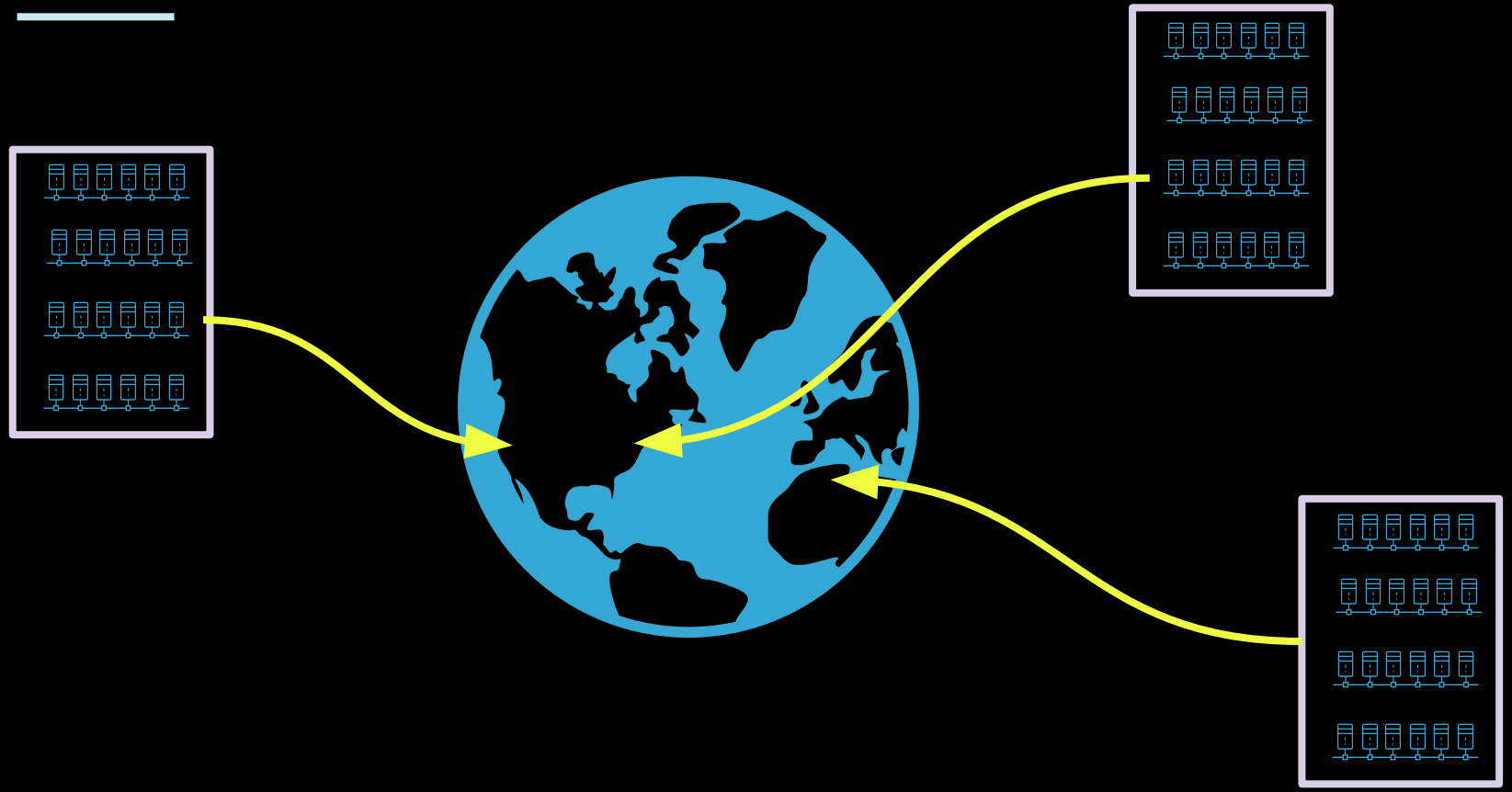
Servers

Datacenter

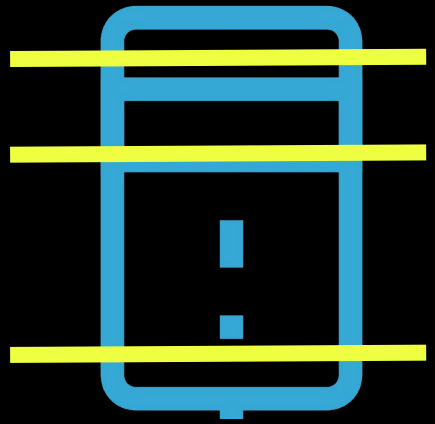


The cloud

The cloud



The cloud



The cloud

Virtual Private Server

VPS

Exercise

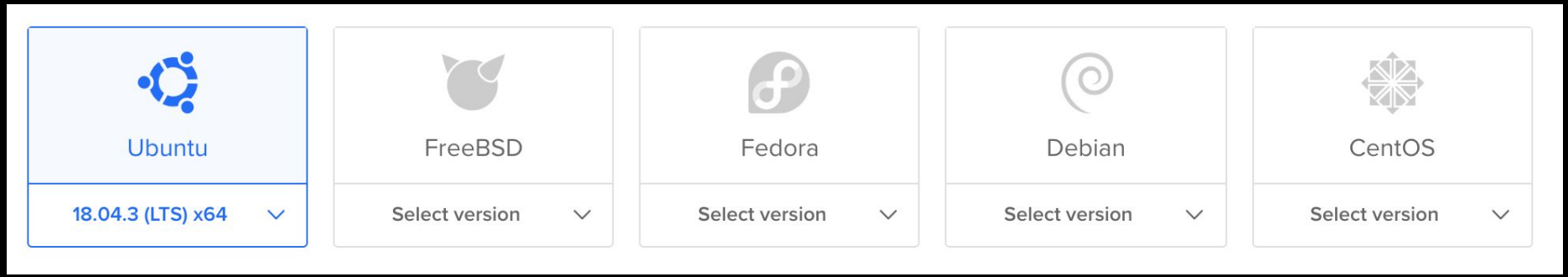
Buying a VPS

VPS

www.digitalocean.com

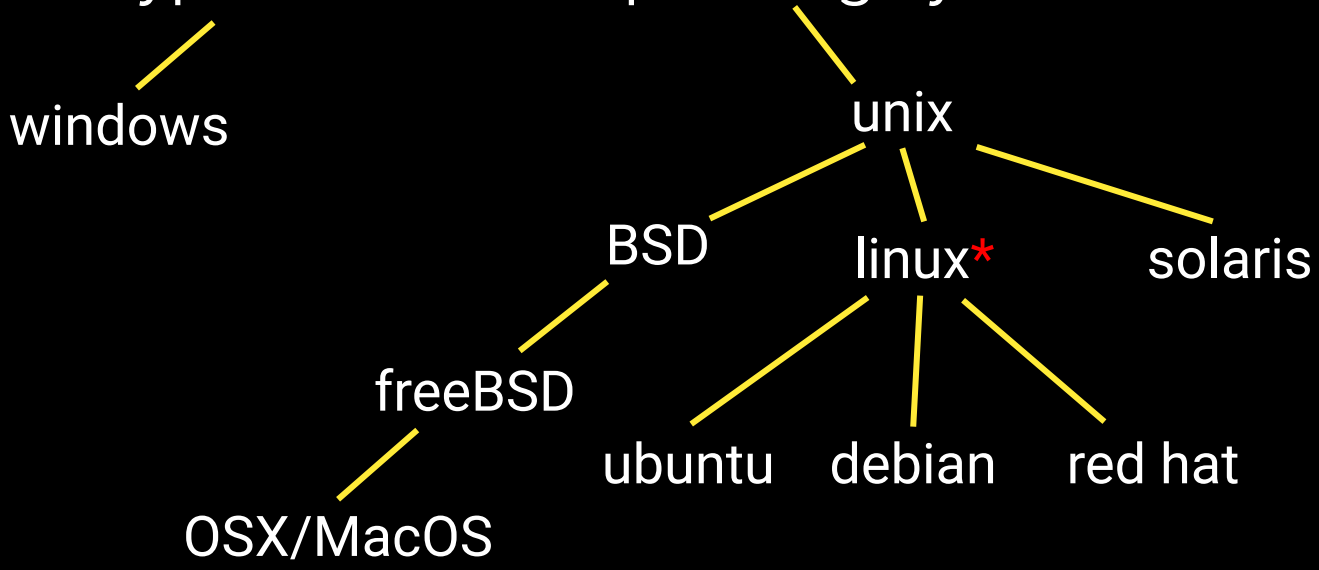
Operating Systems

Operating Systems



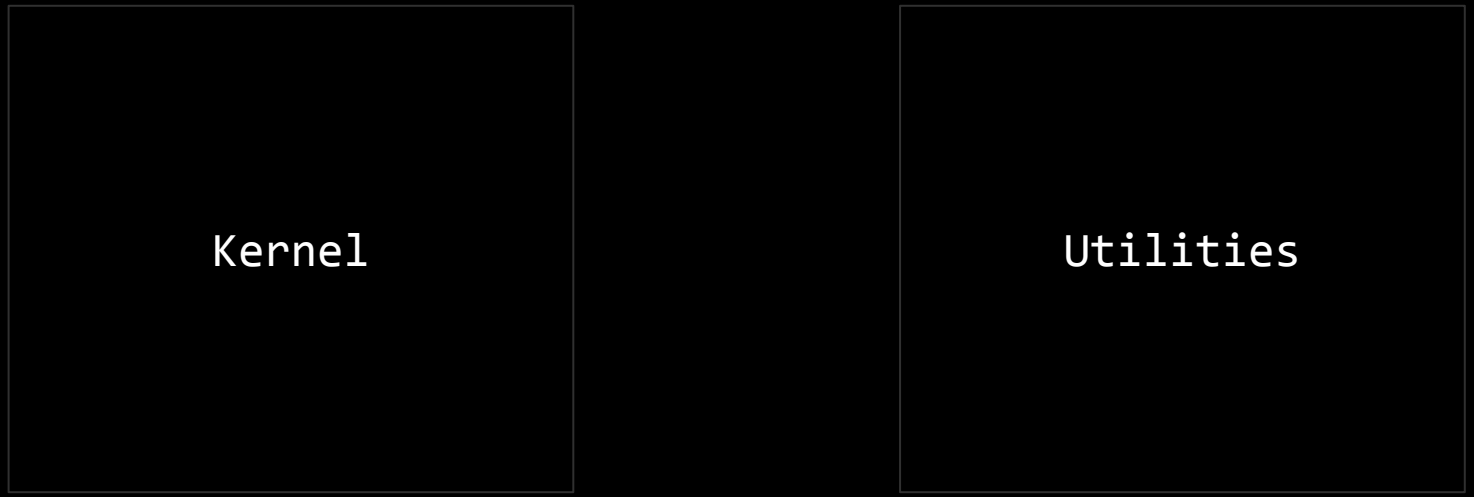
Operating Systems

Two main types of server operating systems

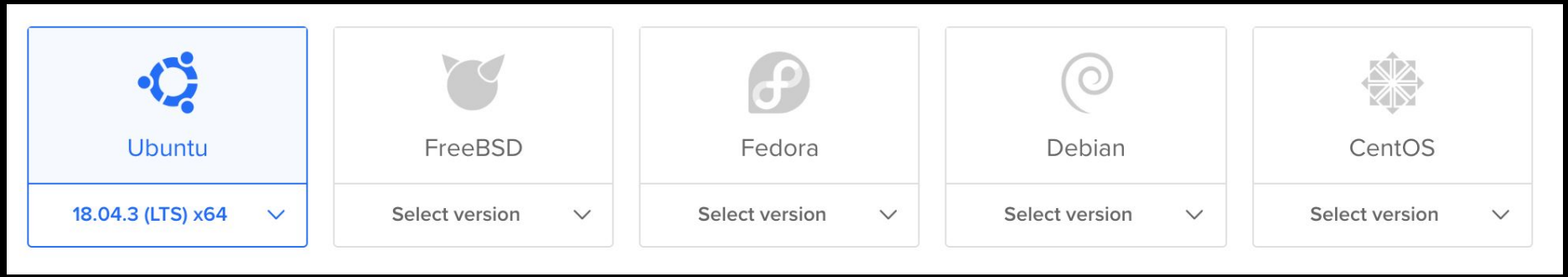


*there are a lot more than 3

Operating Systems



Operating Systems

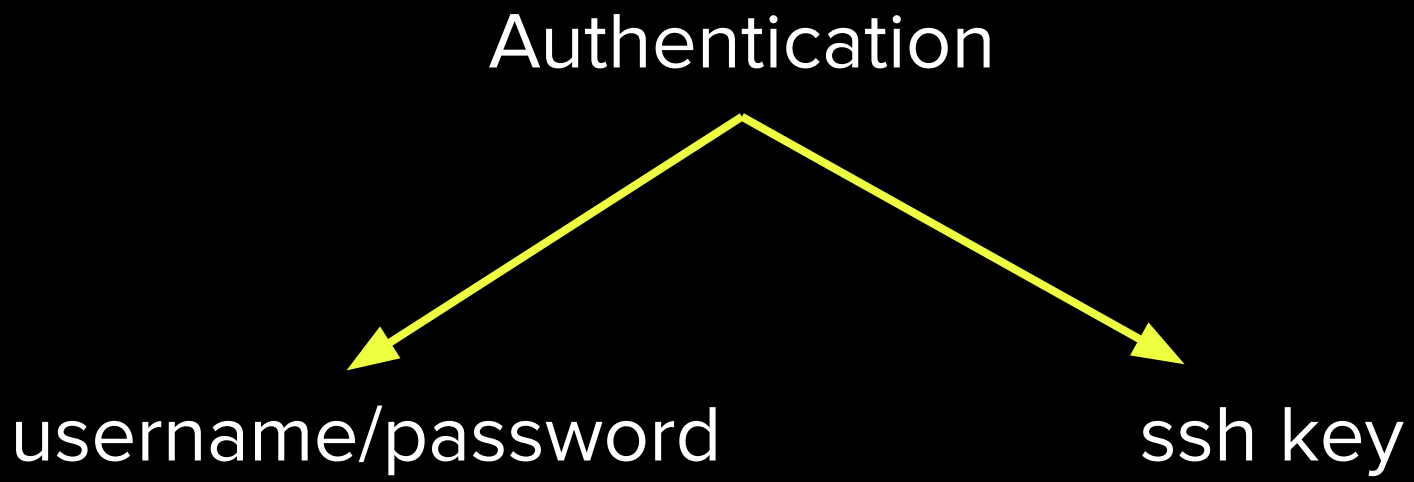


Operating Systems

Long Term Support LTS

SSH

SSH

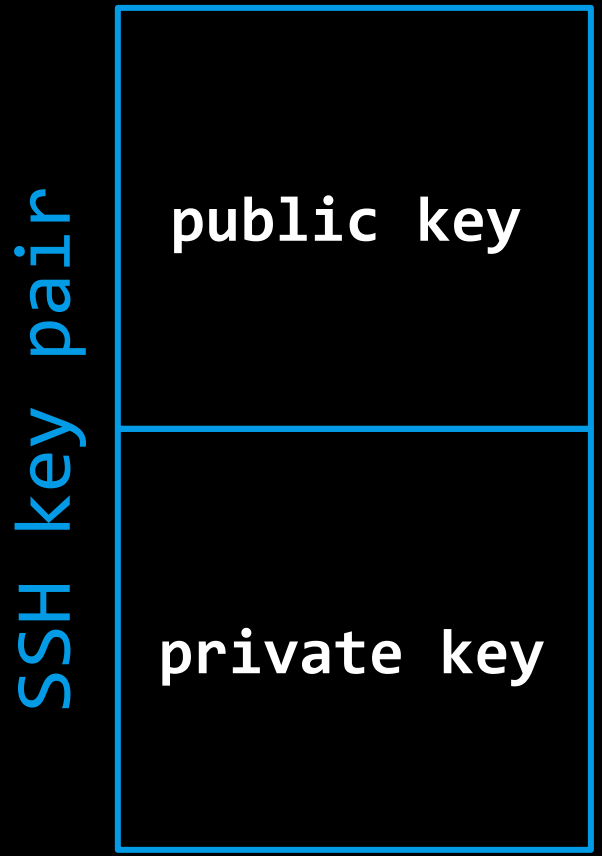


SSH

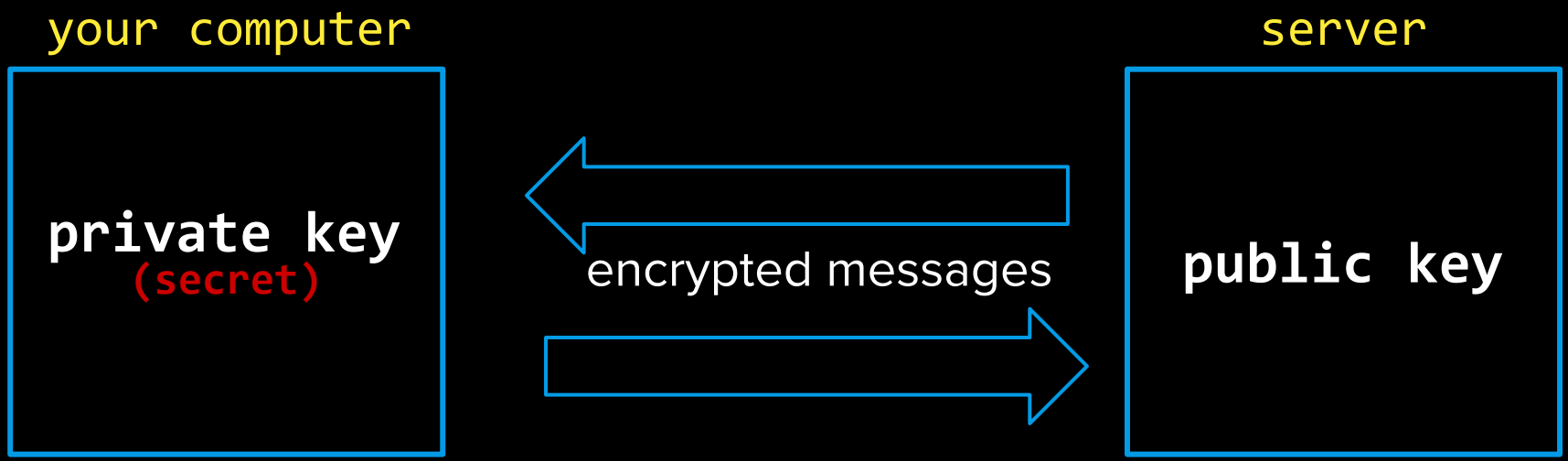
Secure Socket Shell

SSH

SSH



SSH


your computer
private key
(secret)
encrypted messages
server
public key

Move into .ssh directory

```
$ cd ~/.ssh
```

List files

```
$ ls
```

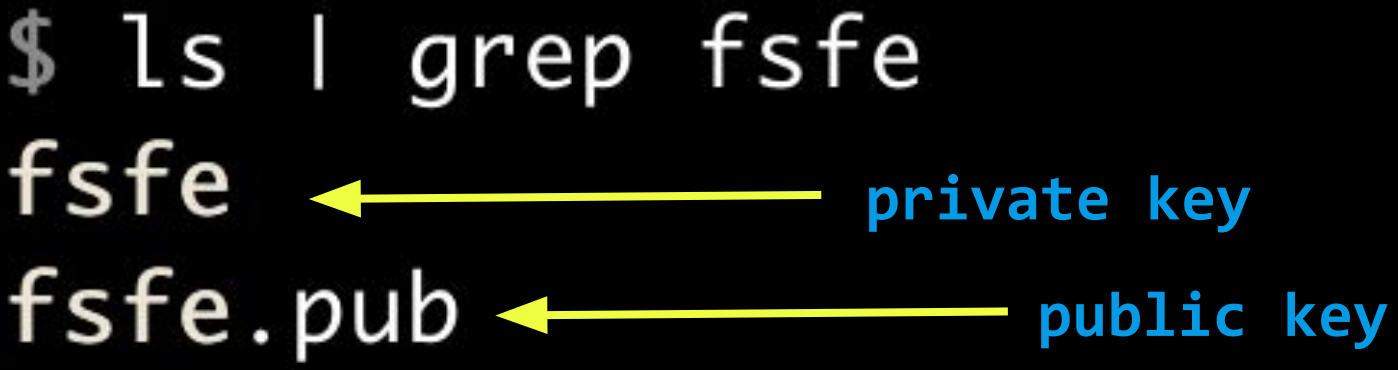
Generate ssh key

```
$ ssh-keygen
```

Exercise

SSH Key

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/jem/.ssh/id_rsa): fsfe
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in fsfe.
Your public key has been saved in fsfe.pub.
The key fingerprint is:
SHA256:40LYSHF/dHqBkvmMAXYo+gVhiXdPAc5iqDn3pju+dxA jem@jem.local
The key's randomart image is:
+---[RSA 2048]----+
|   .+o=+o+..o    |
|  .o=*oo*..o .   |
|  .o++oo.*o .    |
| o.oE=. o.o.     |
|+ ..ooo S        |
| o .o. . .       |
|    o.. .        |
|  .o. ..         |
| .== .           |
+----[SHA256]-----+
```


$ ls | grep fsfe
fsfe
private key
fsfe.pub
public key

Exercise

Logging in

SSH into server

```
$ ssh root@YOUR_IP
```

SSH into server with key file

```
$ ssh -i ~/.ssh/fsfe root@YOUR_IP
```

```
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Sep 11 01:44:06 UTC 2019

  System load:  0.0               Processes:           81
  Usage of /:   4.0% of 24.06GB   Users logged in:     0
  Memory usage: 11%               IP address for eth0: 134.209.122.30
  Swap usage:   0%

0 packages can be updated.
0 updates are security updates.


Last login: Wed Sep 11 01:38:46 2019 from 69.53.245.200
root@ubuntu-s-1vcpu-1gb-nyc1-01:~#
```

Make sure keychain is active

```
$ vi ~/.ssh/config
```

```
Host *
  AddKeysToAgent yes
  UseKeychain yes
```

```
ssh-add -K ~/.ssh/fsfe
```

Add private key to keychain

SSH into server

```
$ ssh root@YOUR_IP
```

Exercise

Buying a domain

Buying a domain

www.namecheap.com

DNS Records

DNS Records

Maps name to IP address `A record`

Maps name to name `CNAME`

DNS Records

www.jemyoug.com

23.23.185.61

DNS Records

blog.jemyoug.com

23.23.185.61

DNS Records

$ dig blog.jemyoung.com

```
;; ANSWER SECTION:
blog.jemyoung.com.      1308    IN      CNAME   jemyoung.com.
jemyoung.com.           1220    IN      A       23.23.185.61
```

Exercise

Domain Setup

DNS Records

Add 2 A Records with your IP address

- @
- *www*

- Use digital ocean to add domain
- Update nameserver on namecheap

Part 3

Satisfying Server Setup

- Server setup
- Nginx
- NodeJS

Server Setup

Server Setup

1. Update software
2. Create a new user
3. Make that user a **super user**
4. Enable login for new user
5. Disable root login

Server Setup

Update software

```
# apt update
```

Upgrade software

```
# apt upgrade
```

Add new user

```
# adduser $USERNAME
```

Server Setup

Add user to “sudo” group

```
# usermod -aG sudo $YOUR_USERNAME
```

Switch user

```
# su $YOUR_USERNAME
```

Check sudo access

```
$ sudo cat /var/log/auth.log
```

Server Setup

Change to home directory

```
$ cd ~
```

Create a new directory if it doesn’t exist

```
$ mkdir -p ~/.ssh
```

Create authorized_keys file and paste PUBLIC key

```
$ vi ~/.ssh/authorized_keys
```

Server Setup

Exit

```
$ exit
# exit
```

Login with new user

```
$ ssh $USERNAME@IP_ADDRESS
```

Server Setup

Change file permissions

```
$ chmod 644 ~/.ssh/authorized_keys
```

Disable root login

```
$ sudo vi /etc/ssh/sshd_config
```

Server Setup

Restart ssh daemon

```
$ sudo service sshd restart
```

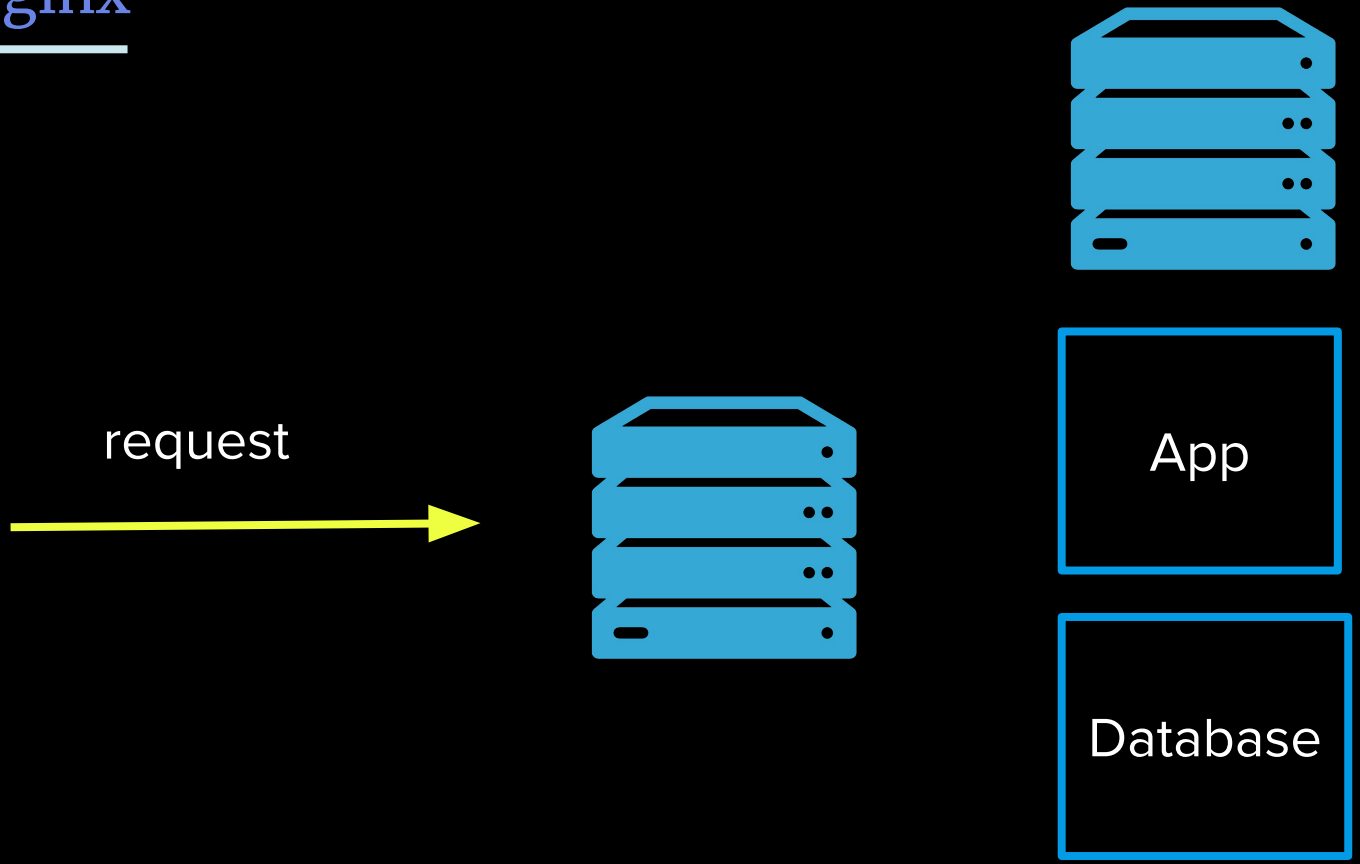
Nginx

Nginx

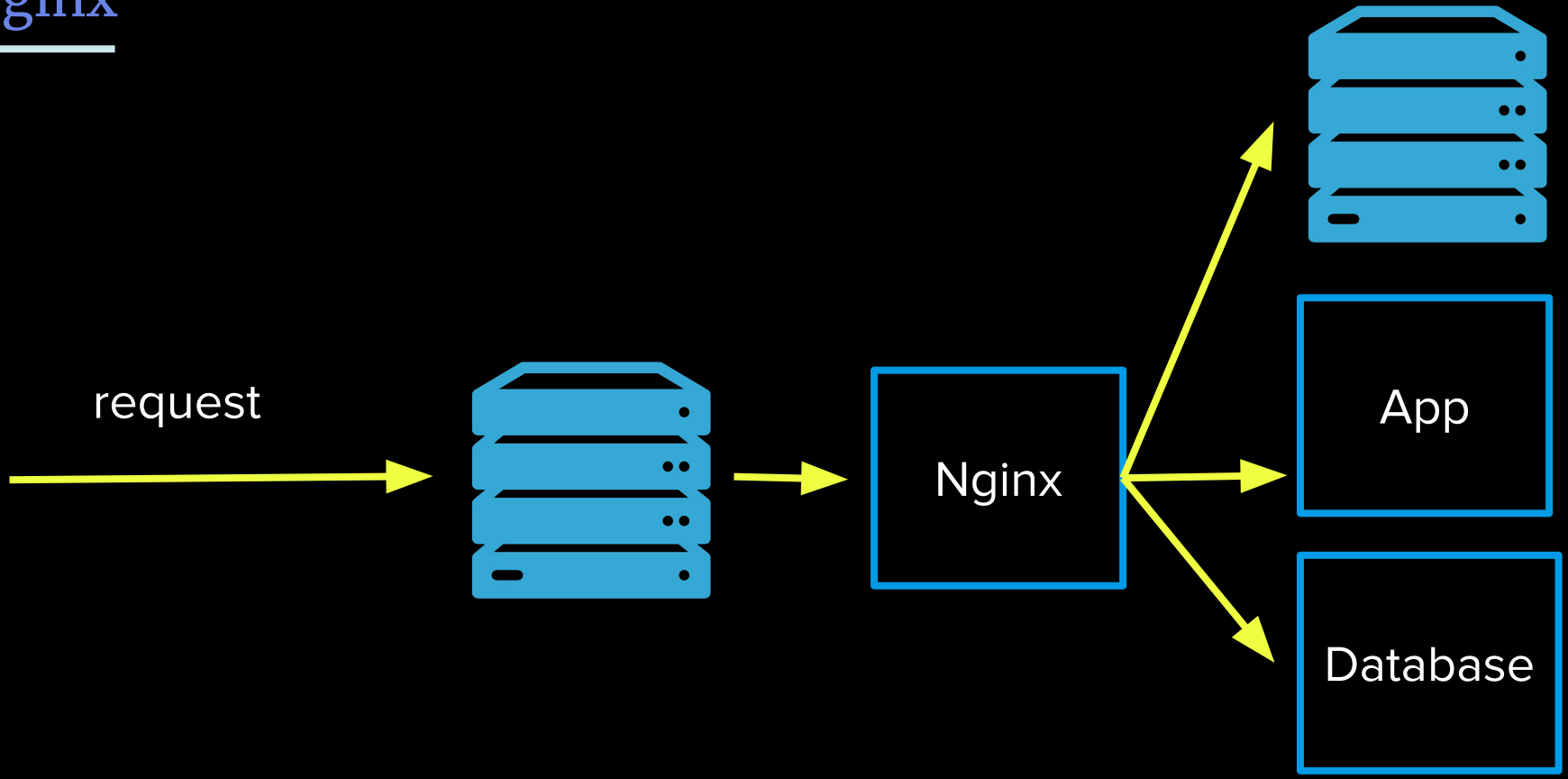
- Reverse proxy
- Web server
- Proxy server

Nginx
(engine-x)

Nginx



Nginx



Exercise

Nginx

Install nginx

```
$ sudo apt install nginx
```

Start nginx

```
$ sudo service nginx start
```

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Nginx

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Nginx

Show nginx configuration

```
$ sudo less /etc/nginx/sites-available/default
```

Nginx

```
root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
}
```

base directory for requests

Nginx

```
root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;
```

location block

```
location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
}
```

Nginx

```
root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
}
```

directive

Nginx

html defaults

```
root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
}
```

Exercise

Default page

Create and edit index.html

```
$ sudo vi /var/www/html/index.html
```

NodeJS

NodeJS

NodeJS

Install NodeJS and npm

```
$ sudo apt install nodejs npm
```

Install git

```
$ sudo apt install git
```

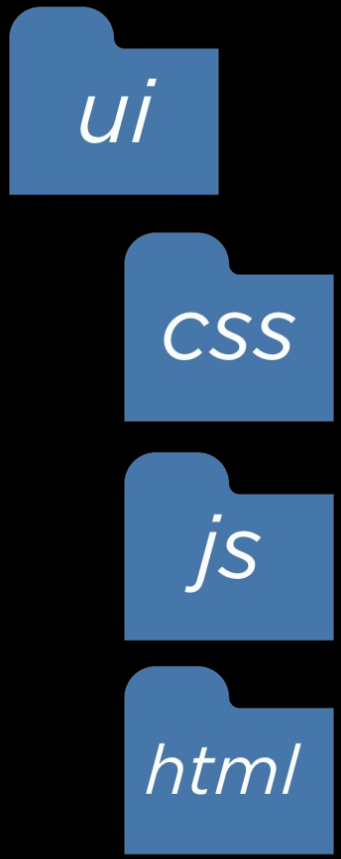
Part 4

Master of the stack

- Application Server
- Process Manager
- Git

Application Architecture

Application Architecture



Exercise

Application Setup

Change ownership of the www directory to the current user

```
$ sudo chown -R $USER:$USER /var/www
```

Create application directory

```
$ mkdir /var/www/app
```

Move into app directory and initialize empty git repo

```
$ cd /var/www/app && git init
```

Create directories

```
$ mkdir -p ui/js ui/html ui/css
```

Create empty app file

```
$ touch app.js
```

Initialize project

```
$ npm init
```

install express

```
$ npm i express --save
```

edit app

```
$ vi app.js
```

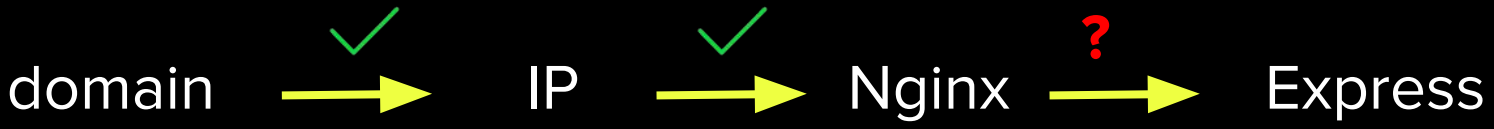
```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {

        res.send('Hello World!');

});

app.listen(port, () => console.log(`Example app listening on port ${port}!`));
```

https://expressjs.com/en/starter/hello-world.html

Run application

```
$ node app.js
```

Nginx



Nginx

```
location / {
  proxy_pass URL_TO_PROXY_TO;
}
```

Exercise

Proxy pass

Edit nginx config

```
$ sudo vi /etc/nginx/sites-available/default
```

```
location / {
  proxy_pass http://127.0.0.1:3000/;
}
```

What about server restarts?

Process Manager

Process Manager

- Keeps your application running
- Handles errors and restarts
- Can handle logging and clustering

Exercise

Process Manager

Install PM2

```
$ sudo npm i -g pm2
```

Start PM2

```
$ pm2 start app.js
```

Setup auto restart

```
$ pm2 save
$ pm2 startup
```

Version Control

Version Control

Record changes to a file system to preserve the history.

Version Control

- Git
- Subversion
- Mercurial

Static Files

Static Files

```
app.use(express.static(‘ui’))
```

Exercise

Git

1. Create git repository
2. Create SSH key
3. Add SSH key to Github
4. Add remote repo
5. Push local repository to Github*

*Don’t forget to add a .gitignore file so you don’t commit *node_modules/*

Congratulations!

Further Exploration

Install Fail2ban

- https://www.techrepublic.com/article/how-to-install-fail2ban-on-ubuntu-server-18-04/

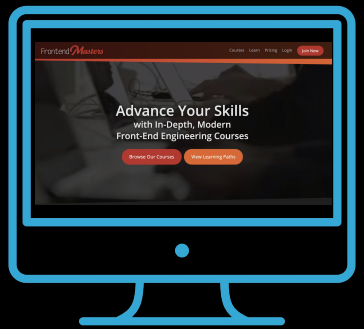
ExpressJS performance tips

- http://expressjs.com/en/advanced/best-practice-performance.html

Recap

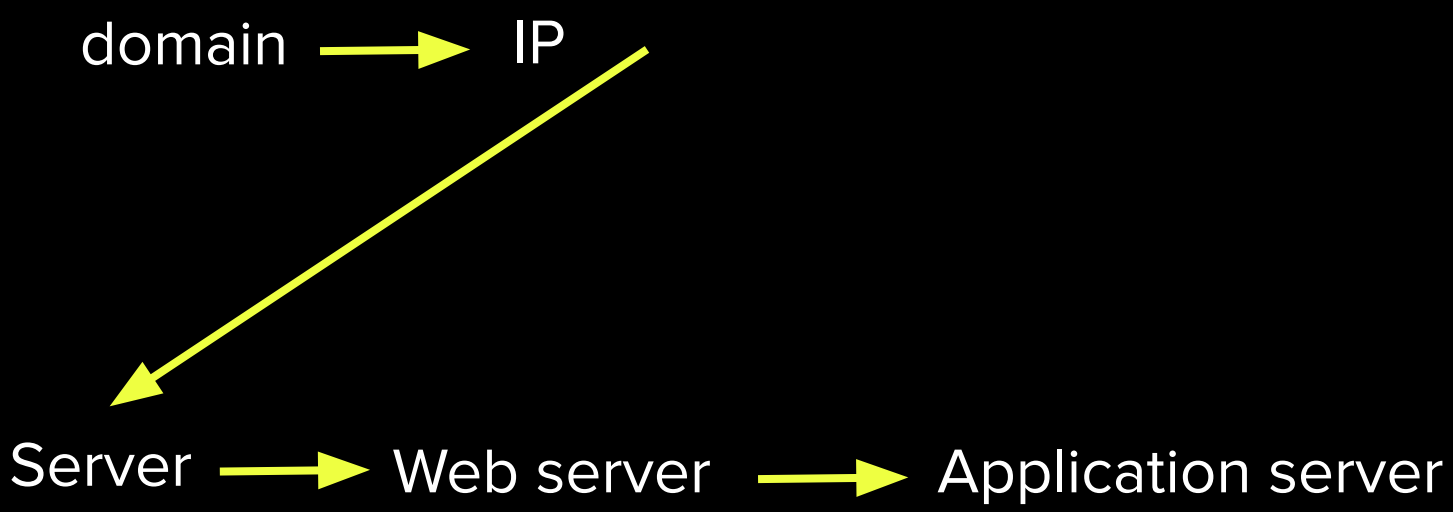
Frontend

Backend



magic?

Recap



Recap

- Command line basics
- How the internet works
- How to create and manage a web server
- Create a Node application

Part 1

bit of this, bit of that

- Standard streams
- Redirection
- Finding things
- Nginx

Standard Streams

Standard Streams

- standard output
  - stdout
- standard input
  - stdin
- standard error
  - stderr

Redirection

Redirection

- |
  - read from stdout
- >
  - write stdout to file
- >>
  - append stdout to file
- <
  - read from stdin
- 2>
  - read from stderr

Redirection

What does this do?

```
foo > bar
```

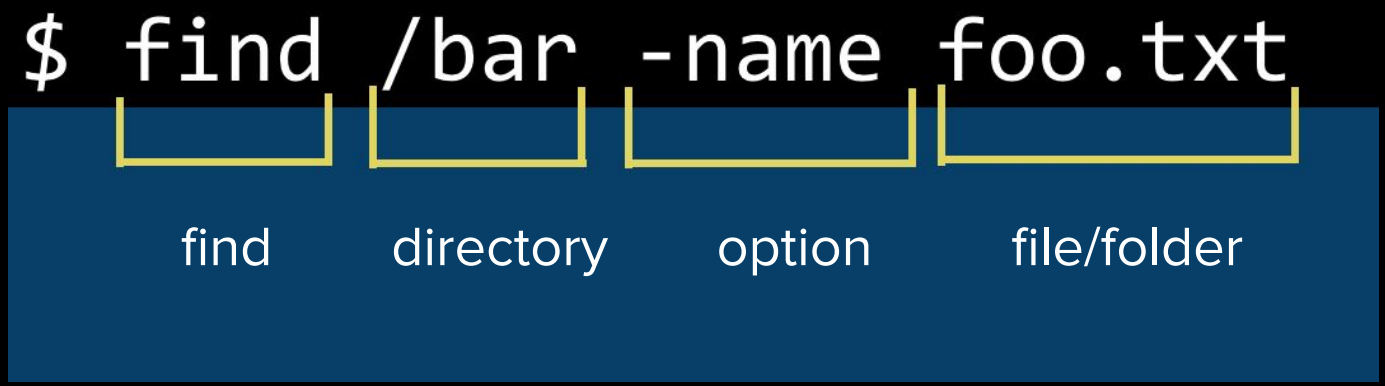
Finding Things

Finding things

search file names — find

search file contents — grep

Finding things



find

Finding things

useful options

- -name
- -type
- -empty
- -executable
- -writable

find

Exercise

Find

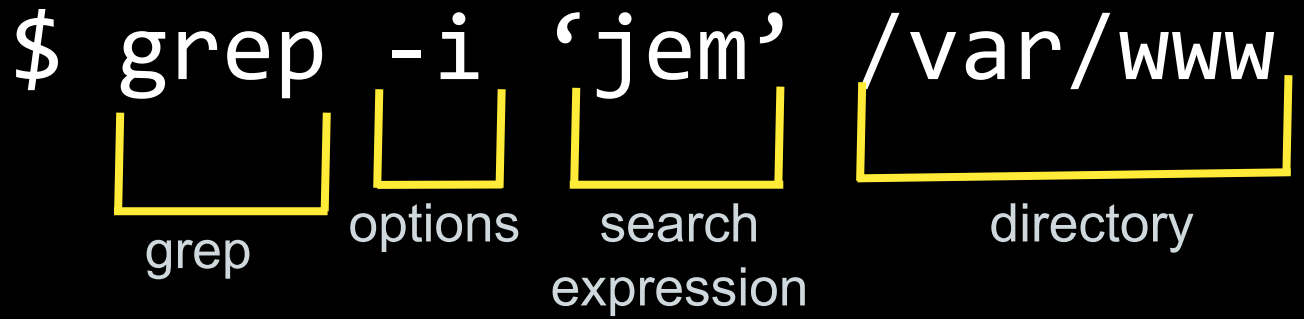
Find all log files in /var/log

```
$ find /var/log/nginx -type f -name “*.log”
```

Find all directories with the name ‘log’

```
$ find / -type d -name log
```

Finding things



grep

```
$ zgrep FILE
```

search inside gzip file

Exercise

Grep

Find running node processes

```
$ ps aux | grep node
```

Nginx

Nginx - Redirect

```
location /help {
  return 301 https://developer.mozilla.org/en-US/;
}
```

Nginx - Subdomain

```
server {
        listen 80;
        listen [::]80; # IPV6 notation

        server_name test.jemisthe.best;

         location / {
            proxy_pass http://localhost:3000;
           }
}
```

Nginx - Gzip

/etc/nginx/nginx.conf

```
##
# Gzip Settings
##

gzip on;
gzip_disable "msie6";

# gzip_vary on;
# gzip_proxied any;
# gzip_comp_level 6;
# gzip_buffers 16 8k;
# gzip_http_version 1.1;
```

Part 2

Security

- Controlling access
- Securing applications
- Firewalls
- Permissions
- Upgrade NodeJS

Security

Read auth.log

```
$ sudo cat /var/log/auth.log
```

Security

What could someone do if they gained ~~root~~ access to your server?

Security

- SSH
- Firewalls
- Updates
- Two factor authentication
- VPN

Updates

Application Updates

Install unattended upgrades

```
$ sudo apt install unattended-upgrades
```

View conf file

```
$ cat /etc/apt/apt.conf.d/50unattended-upgrades
```

Firewalls

Firewall

“A network security device that monitors incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules.”

Firewall

https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html

Firewall

Exercise

nmap

Install nmap

```
$ sudo apt install nmap
```

```
$ man nmap
```

Run nmap

```
$ nmap YOUR_SERVER_IP_ADDRESS
```

Run nmap with more service/version info

```
$ nmap -sV YOUR_SERVER_IP_ADDRESS
```

```
Starting Nmap 7.60 ( https://nmap.org ) at 2019-09-24 03:37 UTC
Nmap scan report for ubuntu-s-1vcpu-1gb-sfo2-01 (157.245.171.164)
Host is up (0.000076s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http    nginx 1.14.0 (Ubuntu)
3000/tcp open  http    Node.js (Express middleware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Firewall

communication endpoint that maps to a specific process or network service

`port`

Firewall

$ less /etc/services

```
tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
fsp             21/udp          fspd
ssh             22/tcp                          # SSH Remote Login Protocol
```

Firewall

```
iptables -p tcp --dport 80 -j REJECT
```

Firewall

uncomplicated
firewall

`ufw`

Firewall

```
$ ufw allow ssh
      allow http
      allow https
      deny
      reject
```

Exercise

ufw

Check firewall status

```
$ sudo ufw status
```

Enable ssh

```
$ sudo ufw allow ssh
```

Enable firewall

```
$ sudo ufw enable
```

```
Status: active

To                         Action      From
--                         ------      ----
80                         ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
80 (v6)                    ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)
```

How would you create a ufw rule to block all HTTP connections?

```
sudo ufw reject http
```

Permissions

Permissions

rwx rwx rwx	chmod 777 filename chmod -R 777 dir	Anybody can read, write, execute.
rwx rwx r-x	chmod 775 filename chmod -R 775 dir	Owner & Group can read, write, execute. Everyone else can read, execute.
rwx rwx r-	chmod 774 filename chmod -R 774 dir	Owner & Group can read, write, execute. Everyone else can read.
rwx r-x r-x	chmod 755 filename chmod -R 755 dir	Owner can read, write, execute. Everyone else can read, execute.

https://isabelcastillo.com/linux-chmod-permissions-cheat-sheet

Exercise

Upgrade node

Download setup script from nodesource

```
$ curl -sL https://deb.nodesource.com/setup_10.x -o
nodesource_setup.sh
```

Run script

```
$ sudo bash nodesource_setup.sh
```

Install nodejs

```
$ sudo apt install nodejs
```

Application Updates

Update outdated packages

```
$ sudo npm update -g
```

Part 3

HTTP/*

- HTTP
- Headers
- Status codes
- HTTPS
- Certbot
- HTTP/2
- HTTP/3

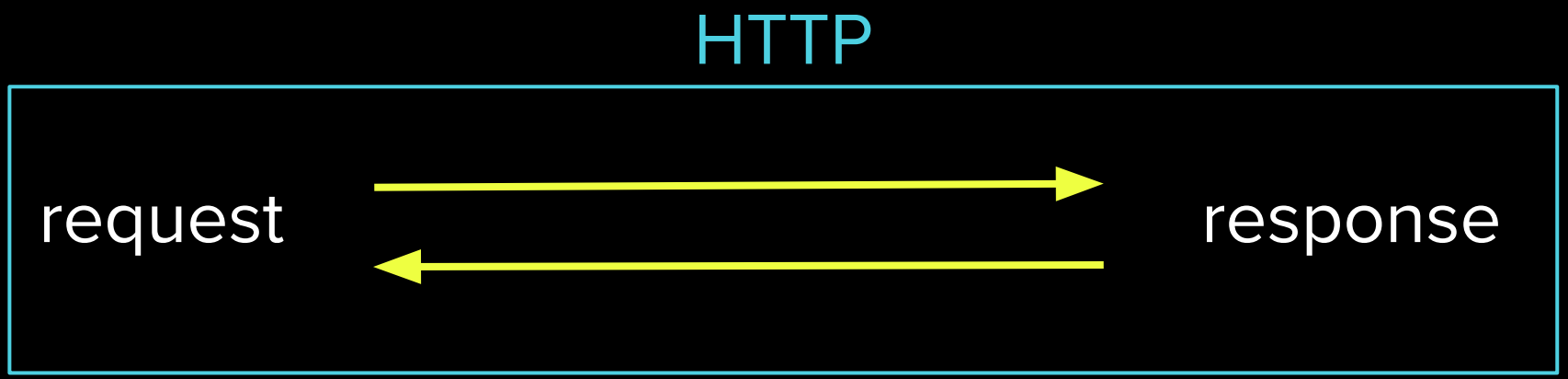
HTTP

HTTP

hypertext transport protocol

HTTP

HTTP



HTTP

```
GET / HTTP/1.1
Host: jemisthe.best
User-Agent: Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_14_5) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/76.0.3809.132
Safari/537.36
Accept: text/html
Accept-Encoding: gzip, br
Accept-Language: en,en-US
```

request

HTTP Headers

HTTP

```
GET / HTTP/1.1
Host: jemisthe.best
User-Agent: Mozilla/5.0 (Macintosh; Intel
Mac OS X 10_14_5) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/76.0.3809.132
Safari/537.36
Accept: text/html
Accept-Encoding: gzip, br
Accept-Language: en,en-US
```

request

HTTP

common headers

User-agent	The requesting device type
Accept	What the device will handle
Accept-language	Browser languages
Content-type	The type of media
Set-cookie	Sets stateful information
X-	Typically used for custom headers

HTTP

```
HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 25 Sep 2019 02:13:13 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 12
```

response

Status Codes

Status Codes

Indicates the status of an HTTP request

HTTP Status Code

Status Codes

200	OK
301	Moved permanently
302	Found (temporary redirect)
401	Not authorized
500	Internal server error

Status Codes

1xx	Information
2xx	Success
3xx	Redirect
4xx	Client error
5xx	Server error

What is the proper status code for a successful POST request?

Exercise

Headers and
Status Codes

In your app, create a path that returns 418 and a custom header.

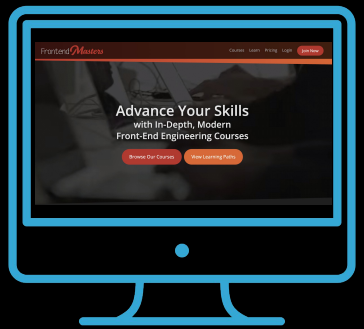
hint: res.status(), res.set()

```
app.get('/demo', (req, res) => {
        res.set('X-full-stack', '4life');
        res.status(418);
        res.send('I prefer coffee');
});
```

HTTPS

HTTPS

frontendmasters.com



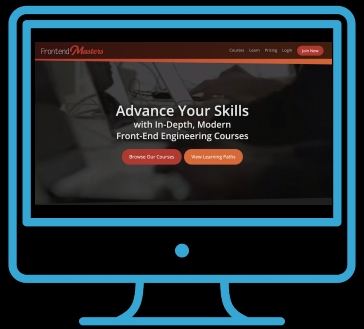
Visa 4235529596901600

HTTPS


frontendmasters.com
Advance Your Skills
with In-Depth, Modern
Front-End Engineering Courses
Evil Server
Visa 4235529596901600

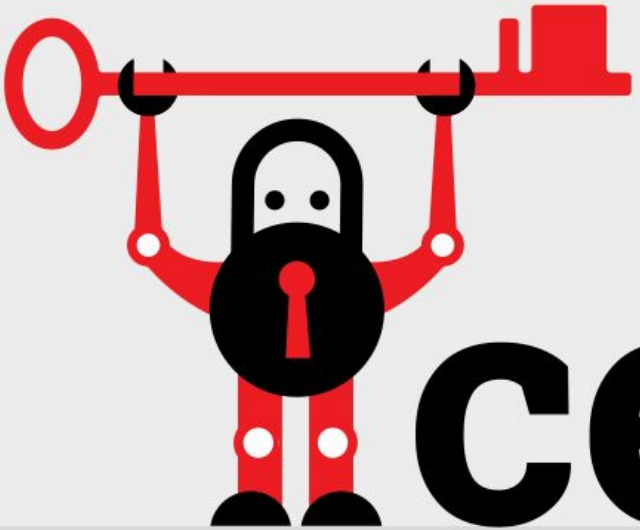
HTTPS

https://frontendmasters.com



*Ifkfi24fh243&&o9#$kD#!8b

HTTPS



certbot

Automatically enable HTTPS on your website.

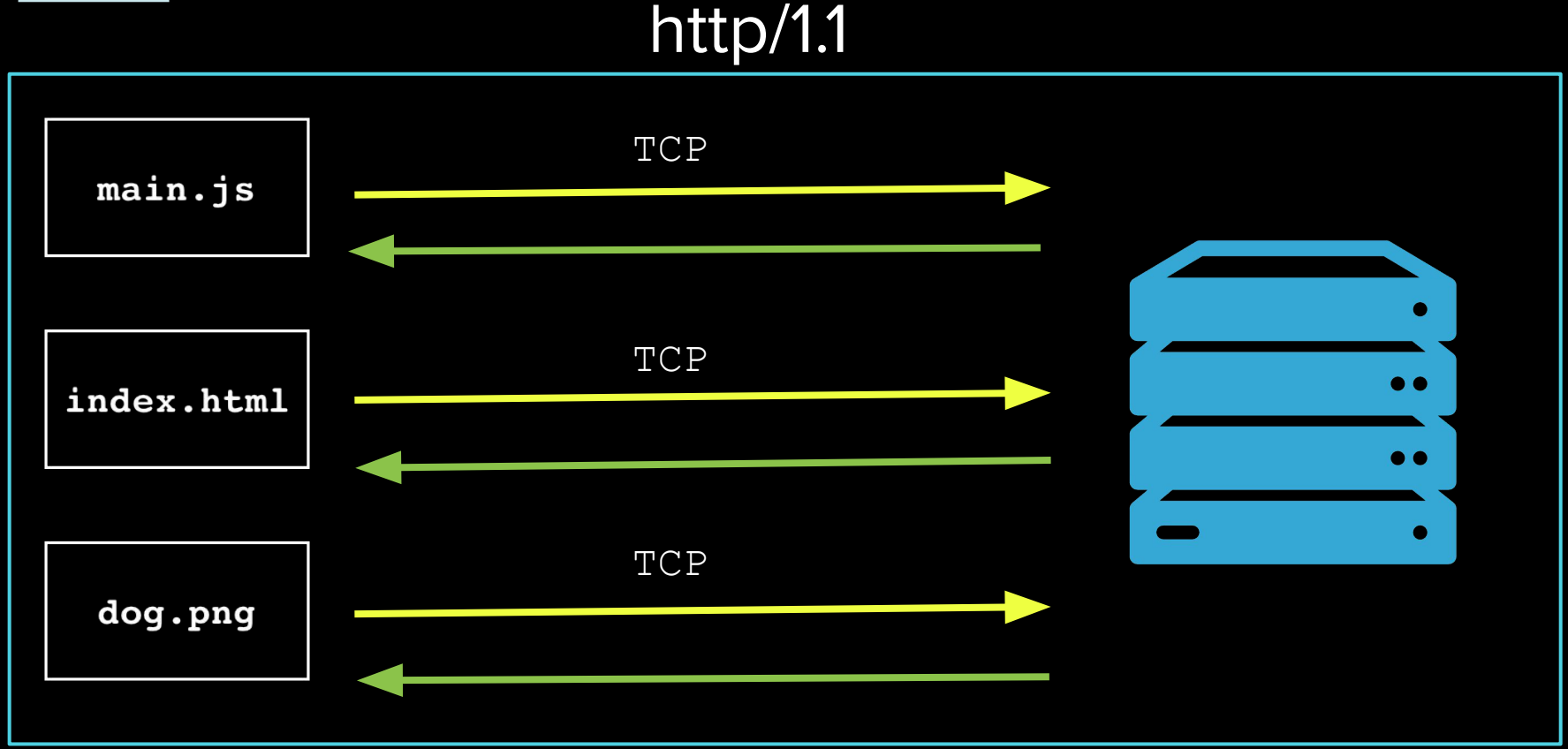
Exercise

Certbot

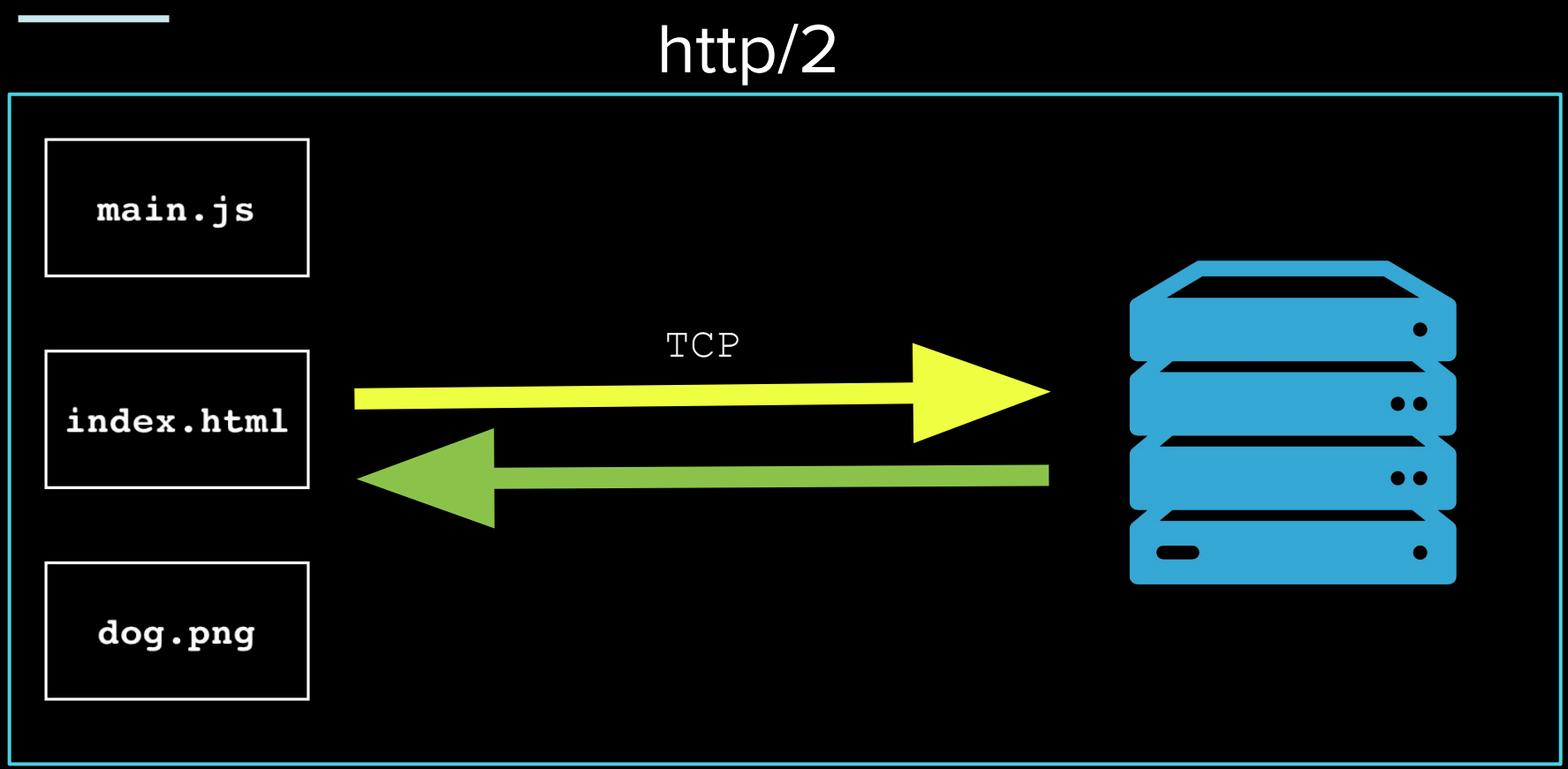
https://certbot.eff.org/

HTTP/2

HTTP/2



HTTP/2



Exercise

http/2

Edit nginx config

```
$ sudo vi /etc/nginx/sites-available/default
```

```
listen 443 http2 ssl;
```

HTTP/3

HTTP/3

Quick UDP Internet Connections

QUIC

Method	Status	Protocol
GET	200	http/2+quic/46

Part 4

Containers

- Microservices
- Containers
- Orchestration
- Load balancers
- Deployments

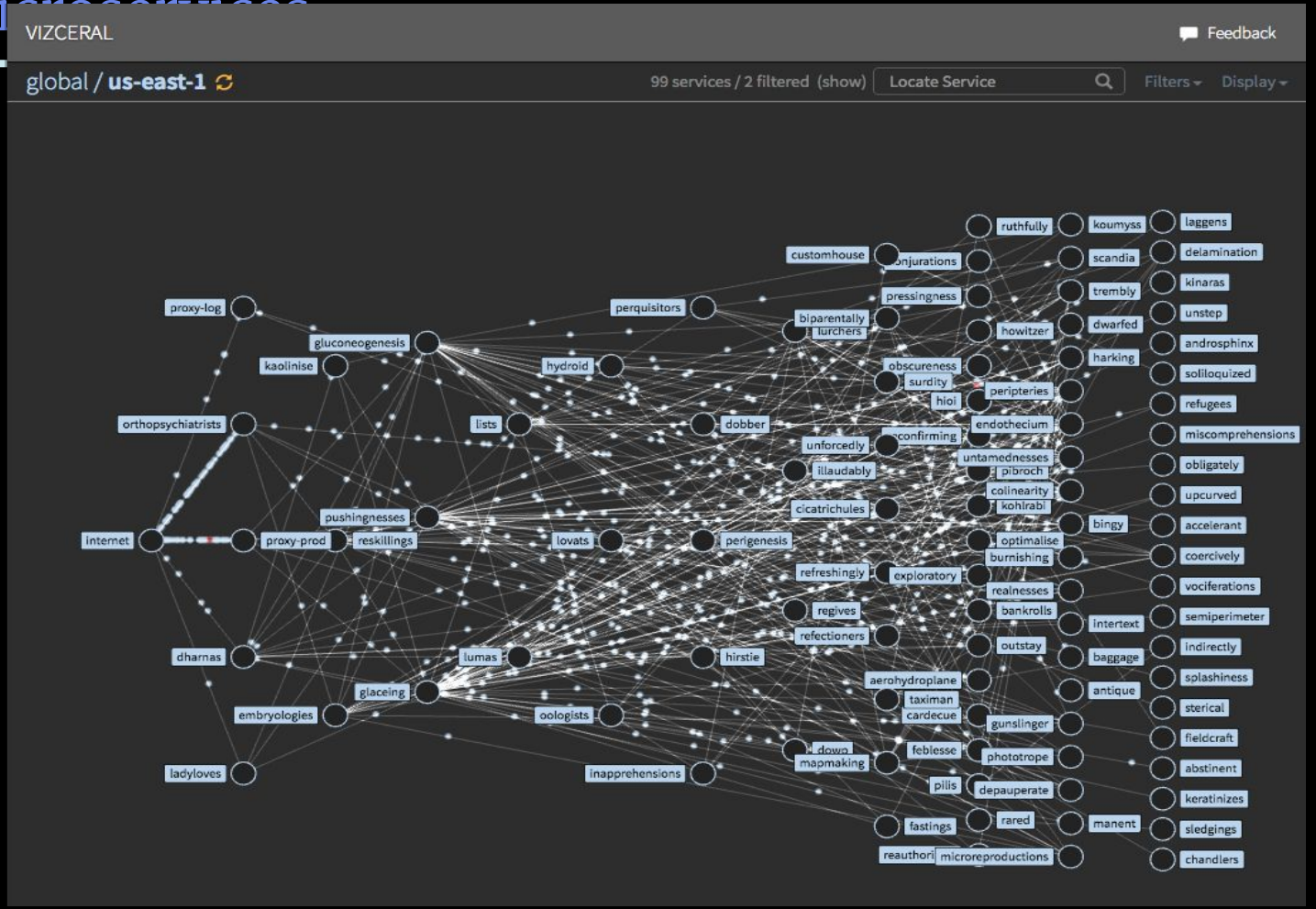
Microservices

Microservices

Architecture of loosely connected services

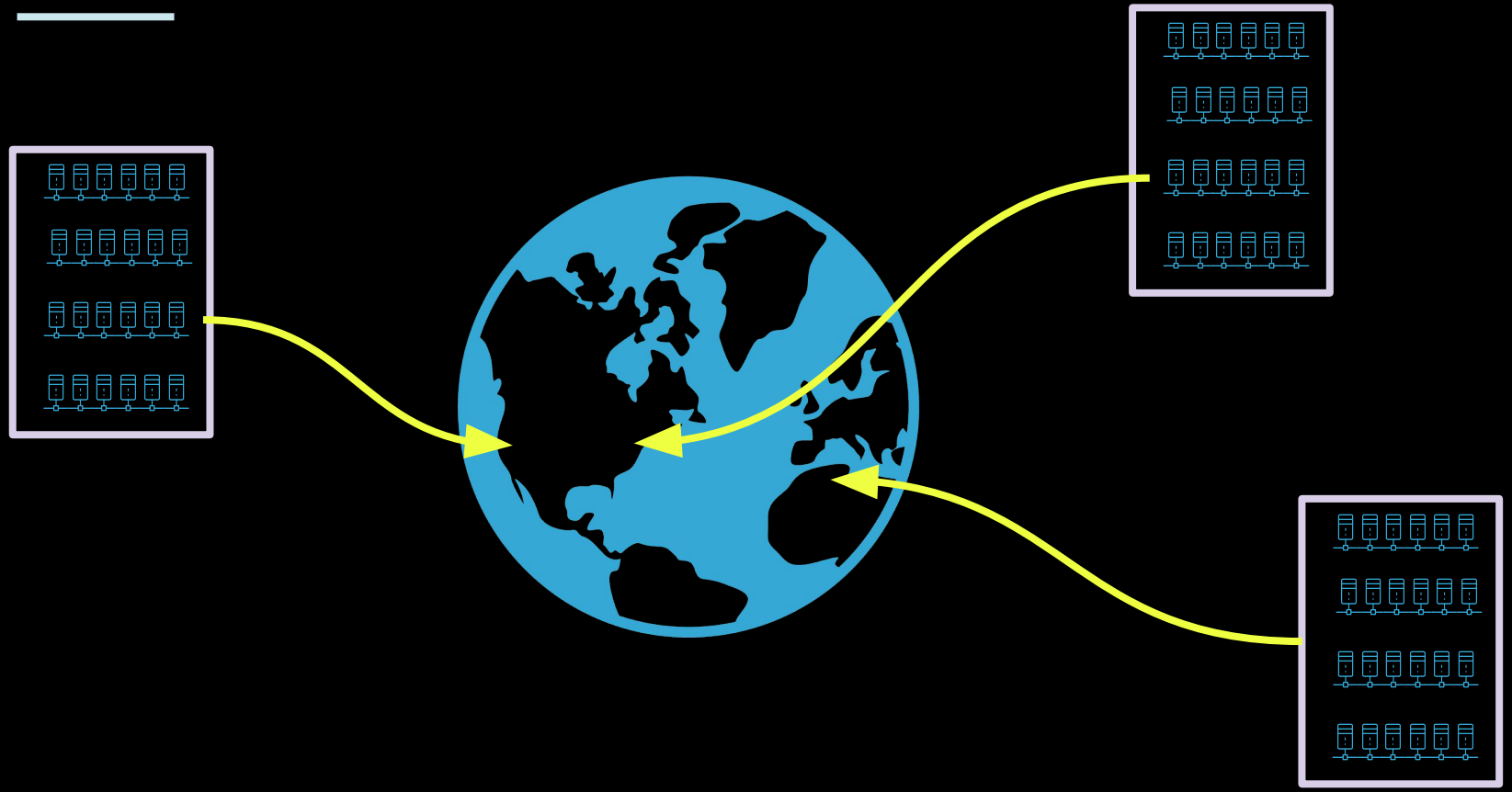
Microservices

Microservices

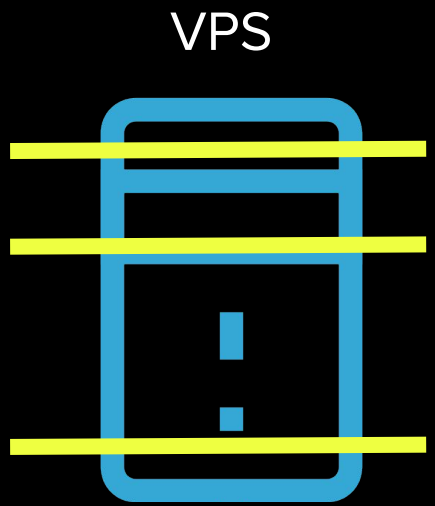


Containers

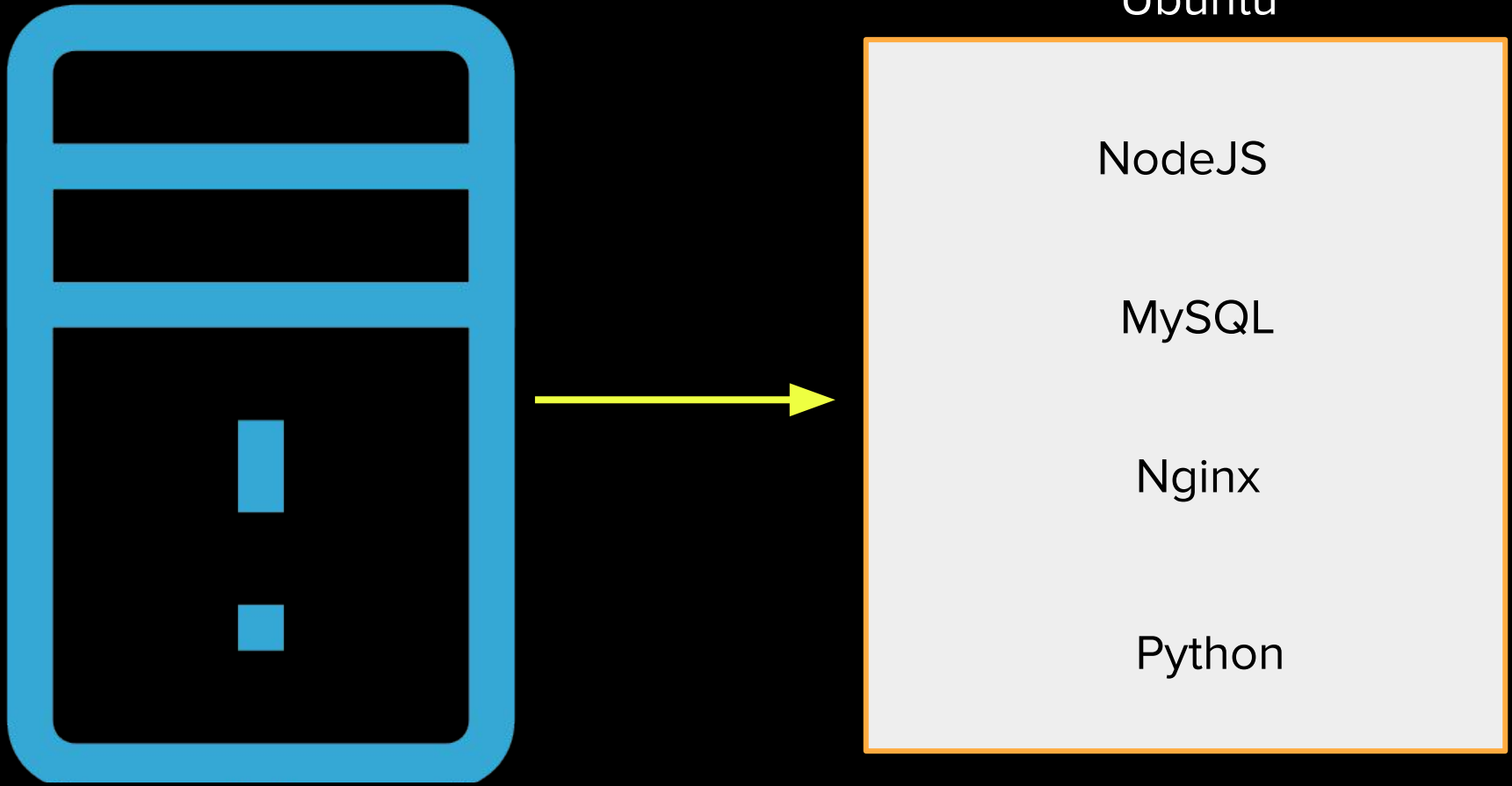
The cloud



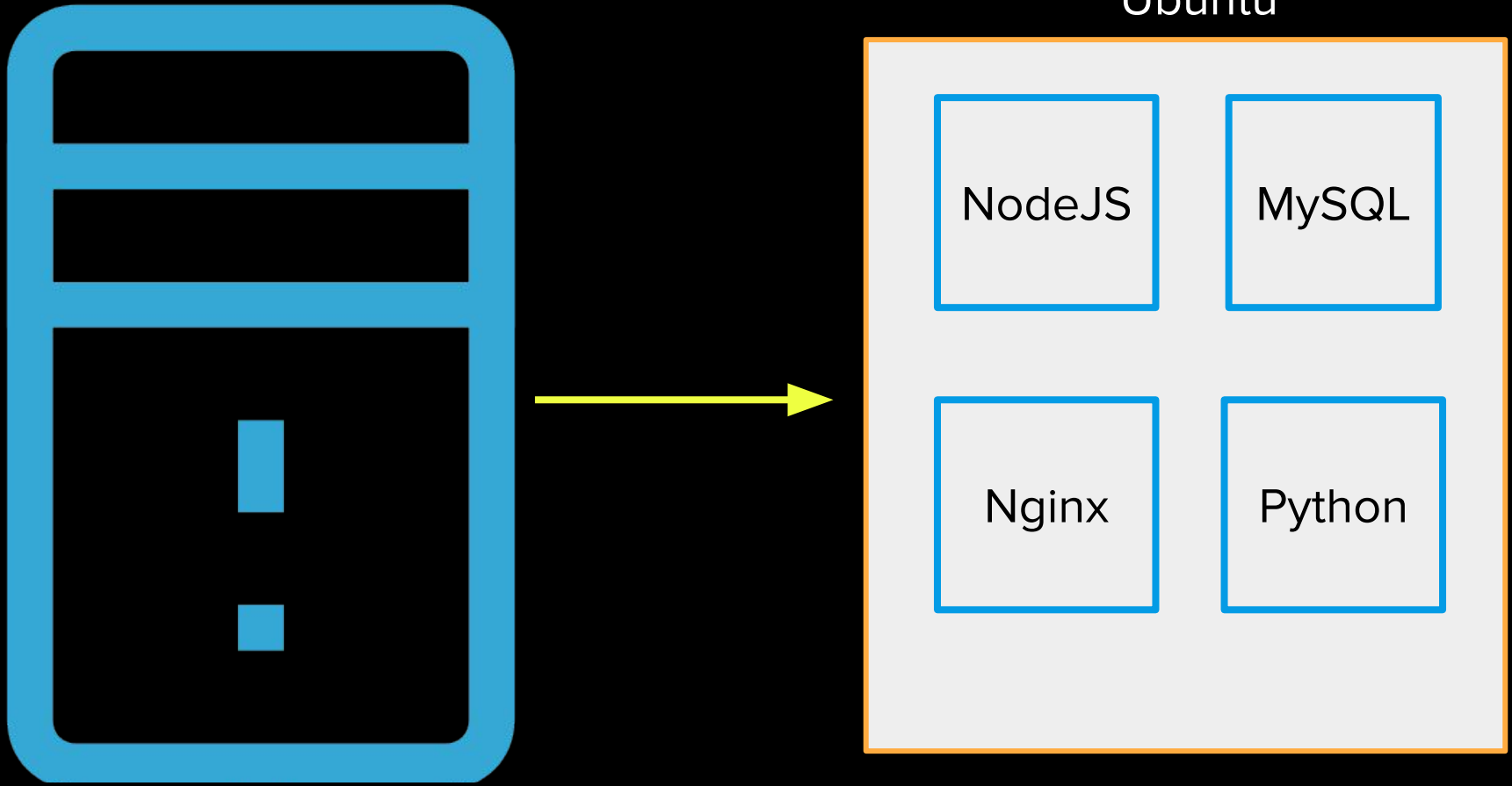
Containers



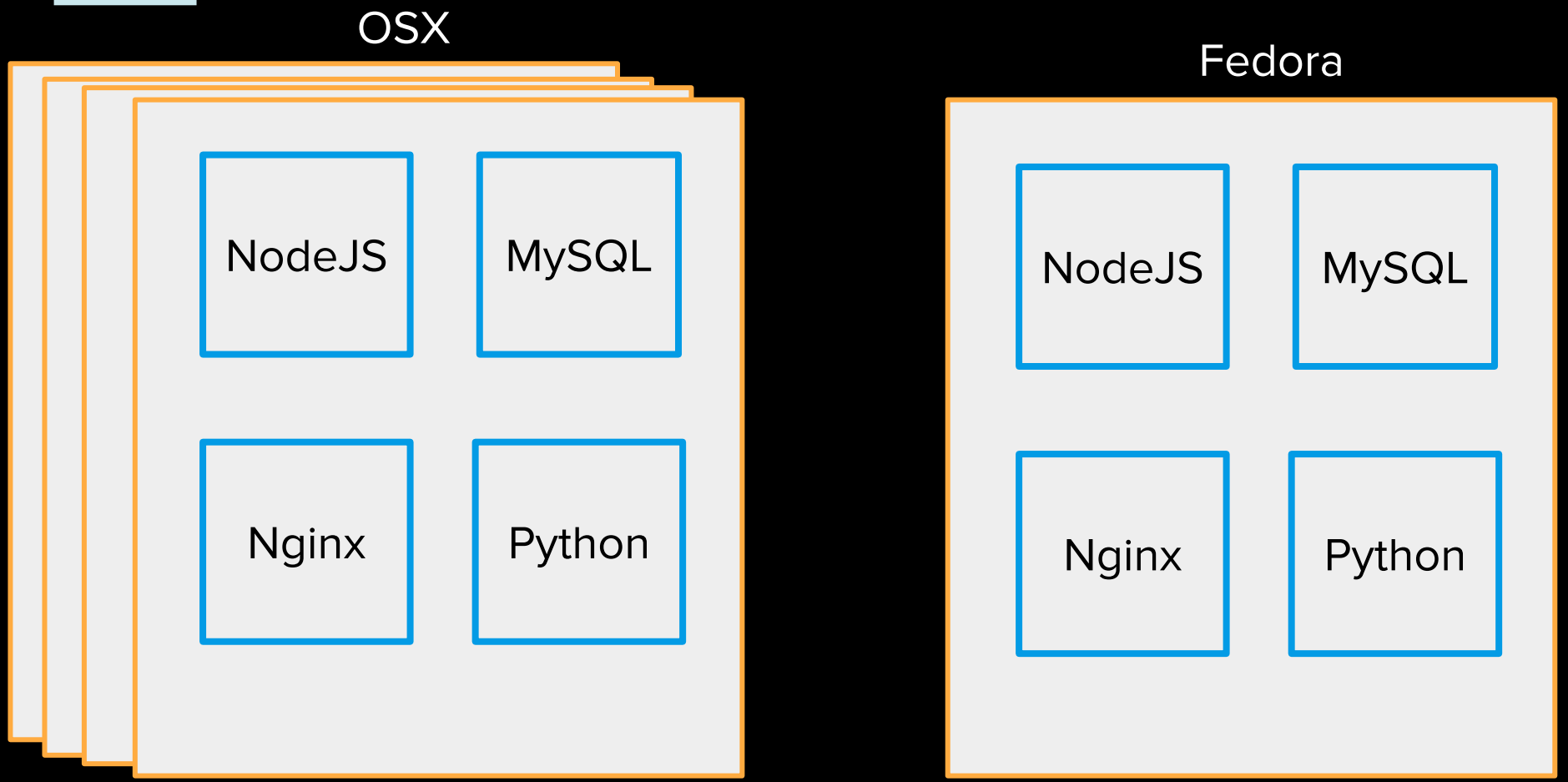
Containers



Containers



Containers


OSX
NodeJS
MySQL
Nginx
Python
Fedora
NodeJS
MySQL
Nginx
Python

Containers

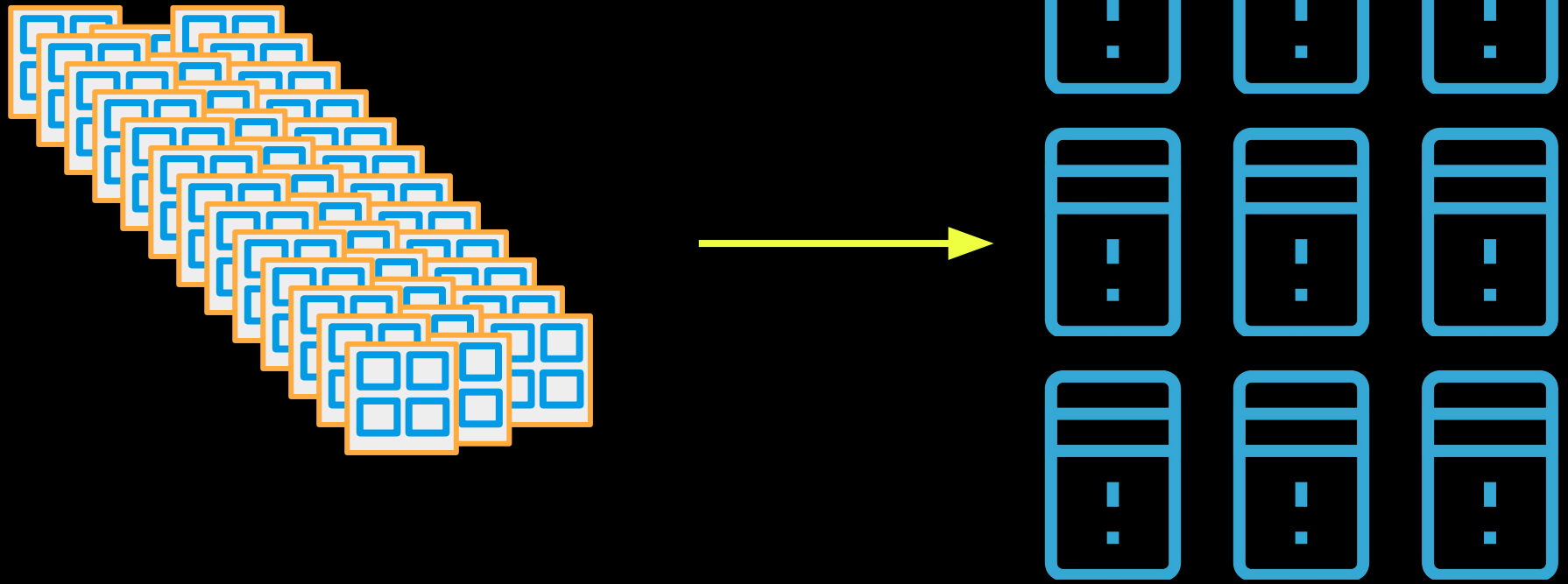
- Lightweight
- Portable
- Easier for development
- Easier to manage
- Faster startup
- Decouple application from infrastructure

Containers

- Amazon ECS
- Apache Mesos
- CoreOS rkt

Orchestration

Orchestration



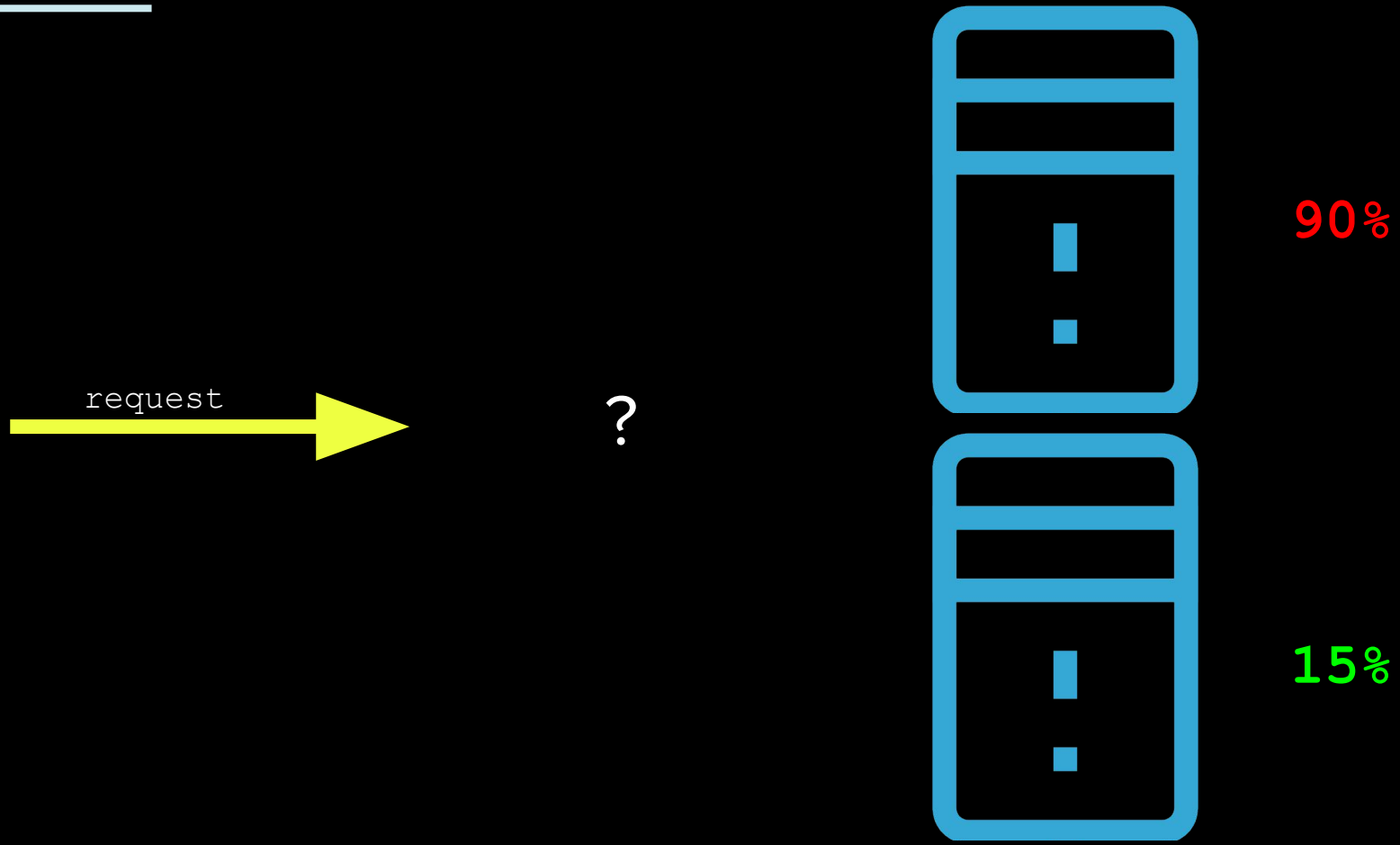
Orchestration

Kubernetes
“K8s”

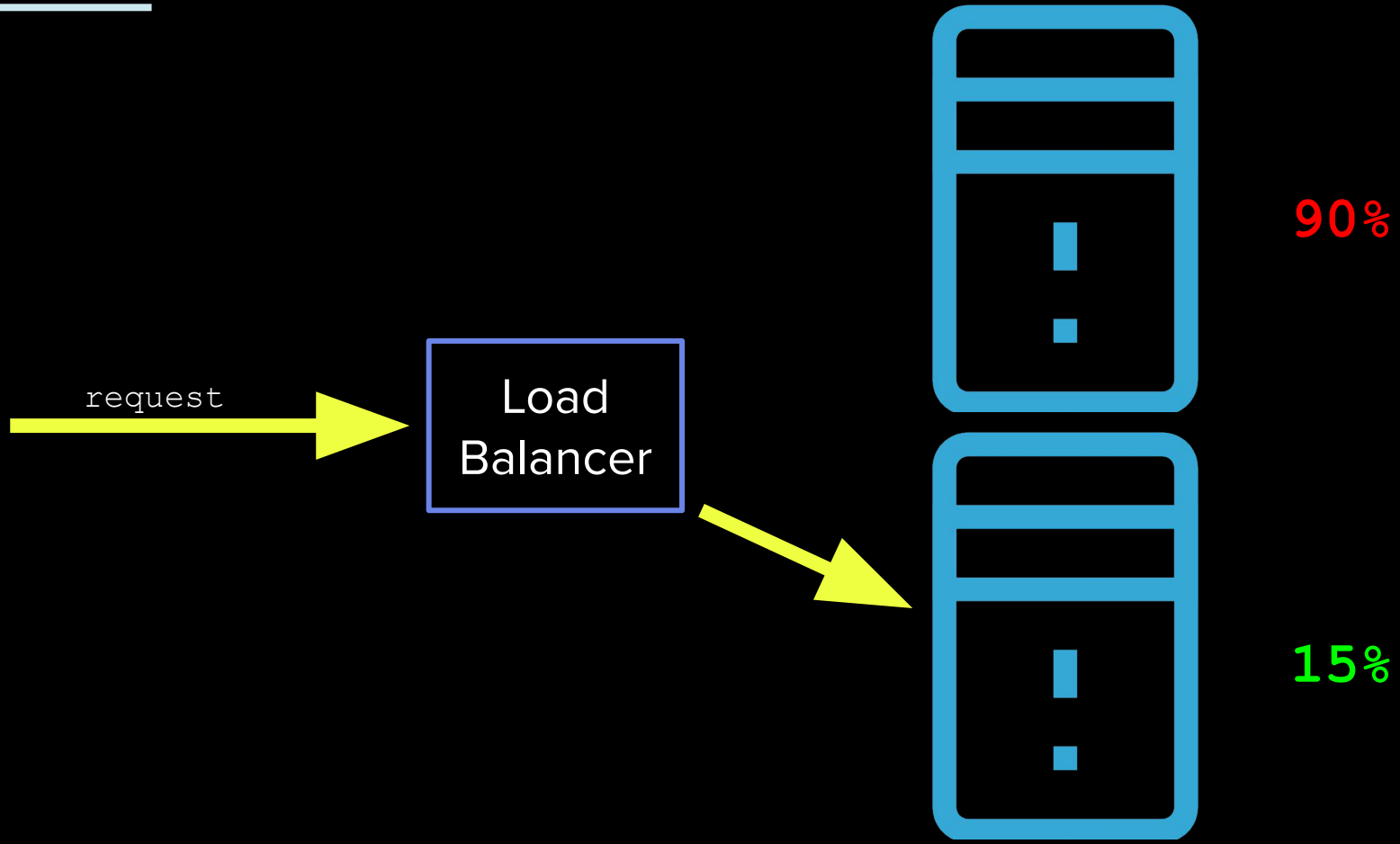
- Docker Swarm
- Amazon EKS
- Apache Mesos
- AKS

Load Balancers

Load Balancers



Load Balancers



Load Balancers

Scheduling Algorithms

- Round Robin*
- IP Hashing
- Random Choice
- Least Connections
- Least Load

Exercise

top

Display running processes

```
$ top
```

Install htop

```
$ sudo apt install htop
```

Display running processes

```
$ htop
```

Load Balancers

```
upstream backend {
    server backend1.example.com;
    server backend2.example.com;
    server 192.0.0.1 backup;
}

server {
    location / {
        proxy_pass http://backend;
    }
}
```

Load Balancers

```
upstream backend {
    least_conn;
    server backend1.example.com;
    server backend2.example.com;
}
```

https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/

Deployments

Deployments

Part 5

Saving Data

- Files
- Databases

Files

Files

Why not save everything to a file?

Databases

Databases

relational

- SQL
- Tables
- Related data
- Strict structure

Databases

non-relational

- NoSQL
- Data agnostic
- Loose structure

Redis

Redis

Install redis server

```
$ sudo apt install redis-server
```

Edit config to start with system

```
$ sudo vi /etc/redis/redis.conf
```

```
supervised systemd
```

Restart redis server

```
$ sudo systemctl restart redis.service
```

Redis

https://github.com/NodeRedis/node_redis

MySQL

MySQL

Install mysql server

```
$ sudo apt install mysql-server
```

Run setup

```
$ mysql_secure_installation
```

MySQL

https://github.com/mysqljs/mysql

Part 6

Final Project

- Websockets

Websockets

Websockets

Persistent bidirectional connection between client and server

`Websocket`

Websockets

```
location / {
  proxy_set_header Upgrade $http_upgrade;
  proxy_set_header Connection "upgrade";

  proxy_pass http://127.0.0.1:3000;
}
```

Final Project

Final Project

https://github.com/young/fsfev2

1. Create a chat bot using websockets

Congratulations!