CAB230 Web Computing
Lecture 9 – Node Deployment

Faculty of Science and Engineering

Semester 2, 2020

Aims of the Lecture

- To understand more about production deployment
- To understand the role of reverse proxy systems

An Agenda

- Introducing NGINX
- Proxy and Reverse Proxy
- Node and NGINX
- Installing NGINX on a VM
- Securing NGINX
- Deploying Node

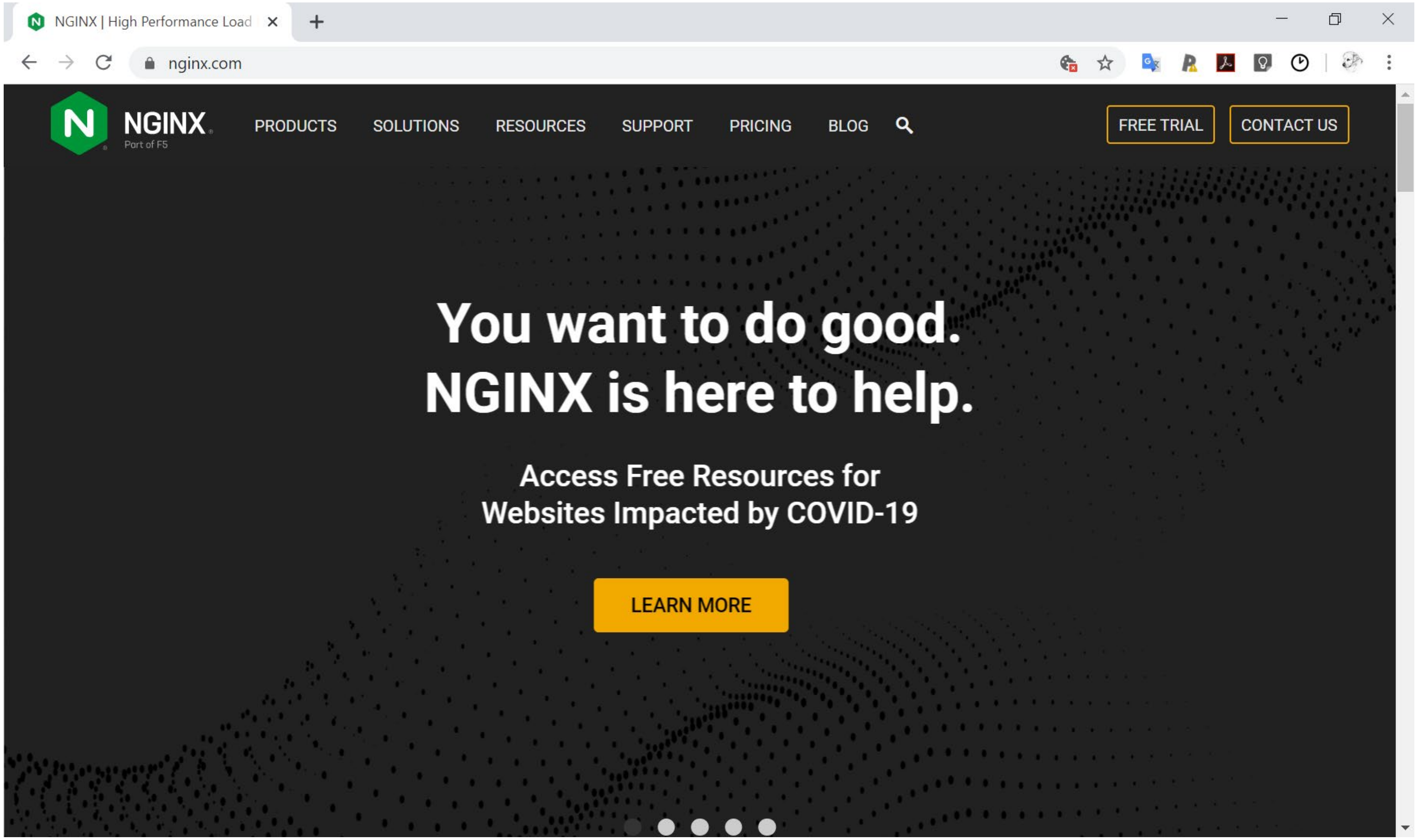
Node is a server, isn't it?

WHY NGINX?

The Basics

- Node is perfectly capable of being a production server
- Node servers handle billions of connections each day
- But we typically don’t expose them directly
- We take some aspects away from node and assign them to another server which does them better
- The most common choice is NGINX
- (usually pronounced *“Engine-X”*)
- See www.nginx.com

NGINX



You want to do good.
NGINX is here to help.

Access Free Resources for
Websites Impacted by COVID-19

NGNIX Web Server

- Free, open-source, high-performance HTTP server
  - Reverse Proxy support (see later)
  - Scalable asynchronous event driven architecture
  - But much more careful with memory than node
  - Extremely scalable
- Designed to solve the C10K problem
  - https://en.wikipedia.org/wiki/C10k_problem
  - Older server architectures can’t support 10K concurrent connections

NGNIX Web Server

- Some high profile users (from the site):
  - Netflix, Hulu,Pinterest, CloudFlare,
  - Airbnb,WordPress.com, GitHub,
  - SoundCloud, Zynga, Eventbrite,
  - Zappos, Media Temple, Heroku,
  - RightScale, Engine Yard, StackPath, CDN77
- General purpose web server
- Host multiple servers
- Capture and assign requests as a reverse proxy
- Load balancing across multiple server 'instances'

Load Balancing Configuration

```
http {
  upstream myproject {
    server 127.0.0.1:8000 weight=3;
    server 127.0.0.1:8001;
    server 127.0.0.1:8002;
    server 127.0.0.1:8003;
  }

  server {
    listen 80;
    server_name www.domain.com;
    location / {
      proxy_pass http://myproject;
    }
  }
}
```

- Main NGINX server specified at the bottom
- Upstream servers listed above share the work

Understanding Proxy Servers

- A proxy server handles request and response traffic
- The proxy server intercepts a request and forwards it on to another server that handles it and sends a response
- The proxy server forwards the response to the client
- The process is transparent to the client

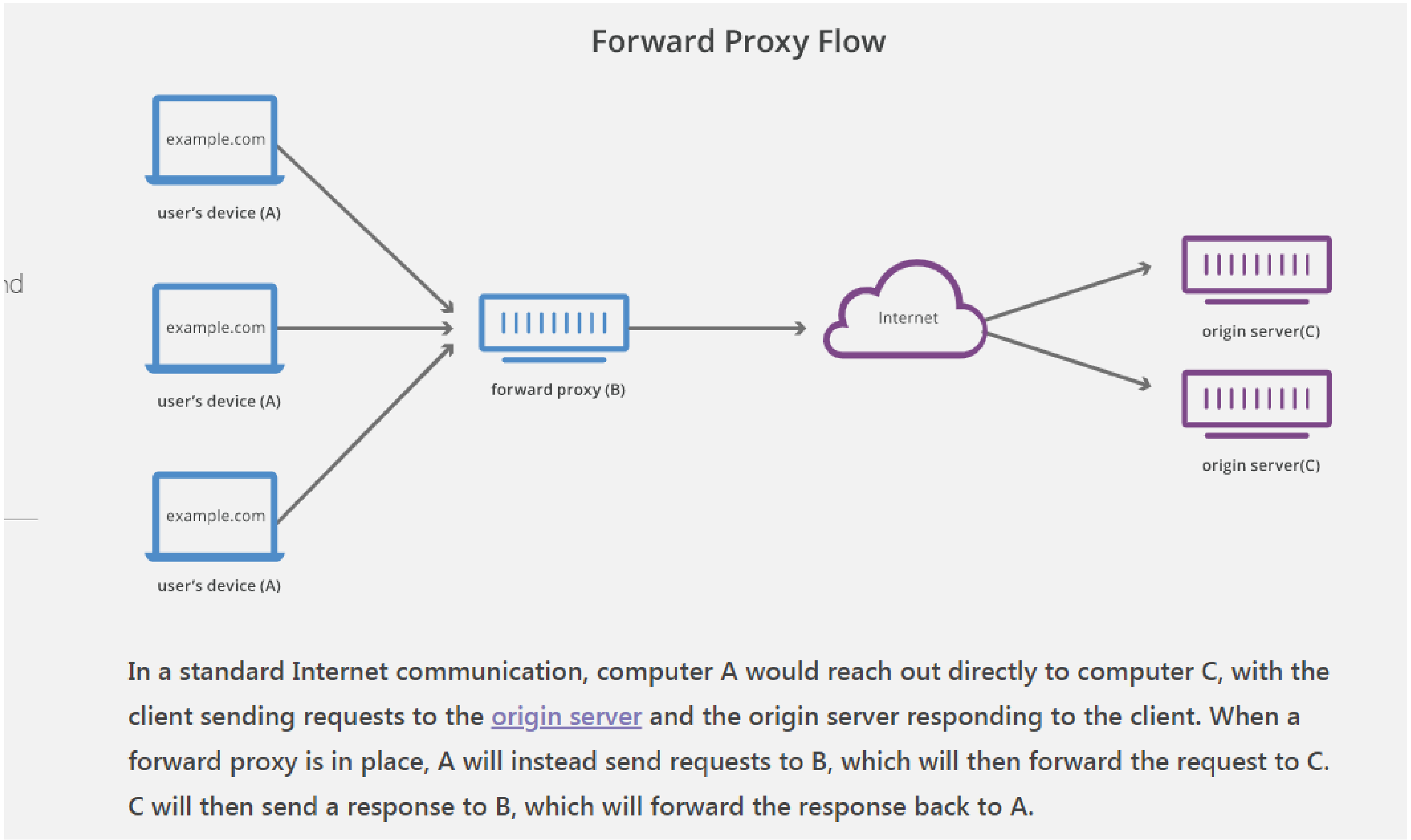
https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/

Understanding Proxy Servers

- Here will introduce forward and reverse proxies
- The label depends on where the proxy sits
  - The forward proxy is on the 'client side'
  - The reverse proxy is on the 'server side'

- Much clearer from the diagrams to follow

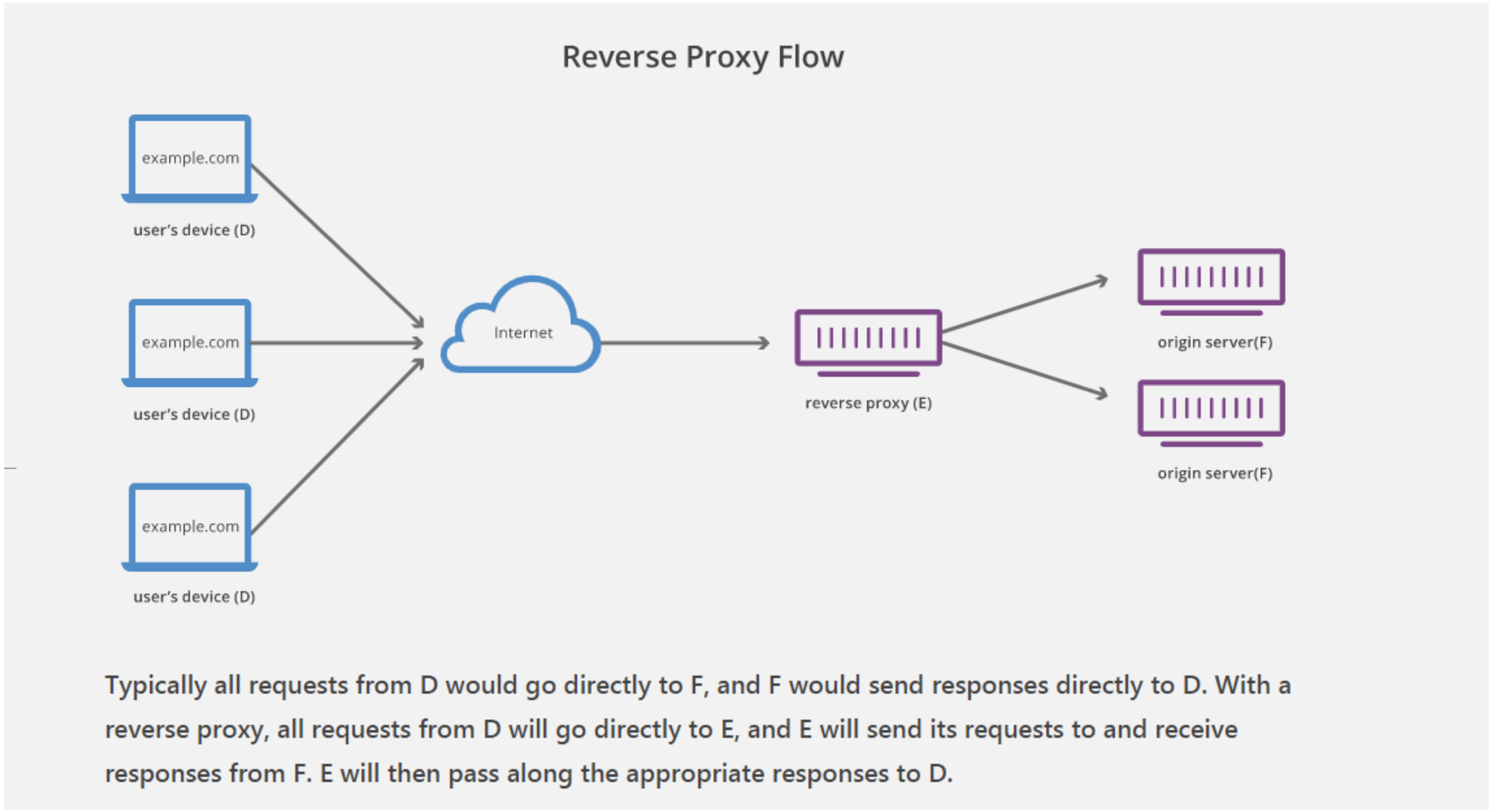
The Forward Proxy



In a standard Internet communication, computer A would reach out directly to computer C, with the client sending requests to the origin server and the origin server responding to the client. When a forward proxy is in place, A will instead send requests to B, which will then forward the request to C. C will then send a response to B, which will forward the response back to A.

https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/

The Reverse Proxy



Typically all requests from D would go directly to F, and F would send responses directly to D. With a reverse proxy, all requests from D will go directly to E, and E will send its requests to and receive responses from F. E will then pass along the appropriate responses to D.

https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/

Uses of a Forward Proxy

- Usually for content filtering
- Very common in schools to block external sites
  - Students make a request to Facebook or some other site
  - Request passes to the proxy, but isn't forwarded to FB

- Conversely, allow some users more extensive access
  - General policy of restricted access to the world
  - Single proxy server with much freer access
  - Privileged users connect to the proxy and get better access

- VPNs and anonymity and avoidance of geoblocking

Uses of a Reverse Proxy

- TLS/SSL encryption
  - Essentially outsourcing from the origin server
  - NGINX does this better than Node
  - See benchmarks in Hunter's piece

- As we have seen in the pracs, node can do TLS/SSL
- But it is safer in production to separate these issues
  - Isolate certificate handling from the application
  - Access from any node module vs access only from NGINX code
  - Easier management

Uses of a Reverse Proxy

- Load balancing (as we have seen before)
- Caching of response content
- Compression of content
  - again CPU intensive, so free up the origin server
  - Very important part of the decision
- Security
  - Isolation and access as discussed above

Production Node

- Here we explore NGINX as a reverse proxy
- Use multiple node servers as the origin servers
  - Load balancing
  - SSL/TLS management
  - Isolation of the server and the application

- Code vs configuration
  - We need to look at a lot of config files
  - We will explore the process for a modern Ubuntu server

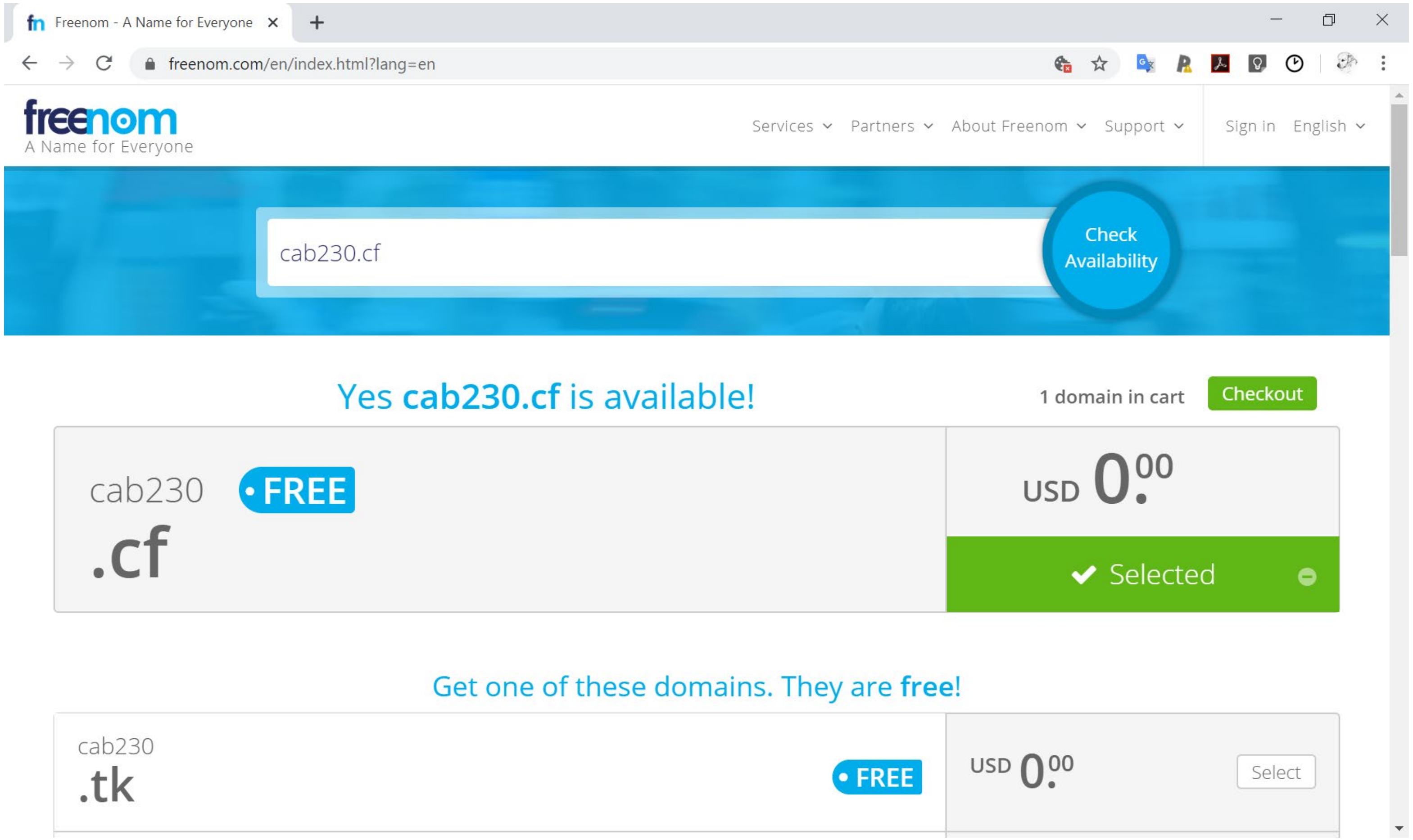
How to make them work together.

NODE AND NGINX

Our Approach

- Most of this content follows the Digital Ocean guides
  - These are linked at the end of the lecture.
- Most of the details are out of scope for this unit
  - But the concepts are important.
- We will work with a very basic node server.
  - Easily replaced with a more complex use case
- We will show the highlights but not every detail
- We will work with the (real) domain `cab230.cf`

The Domain Name



cab230.cf

Yes cab230.cf is available!

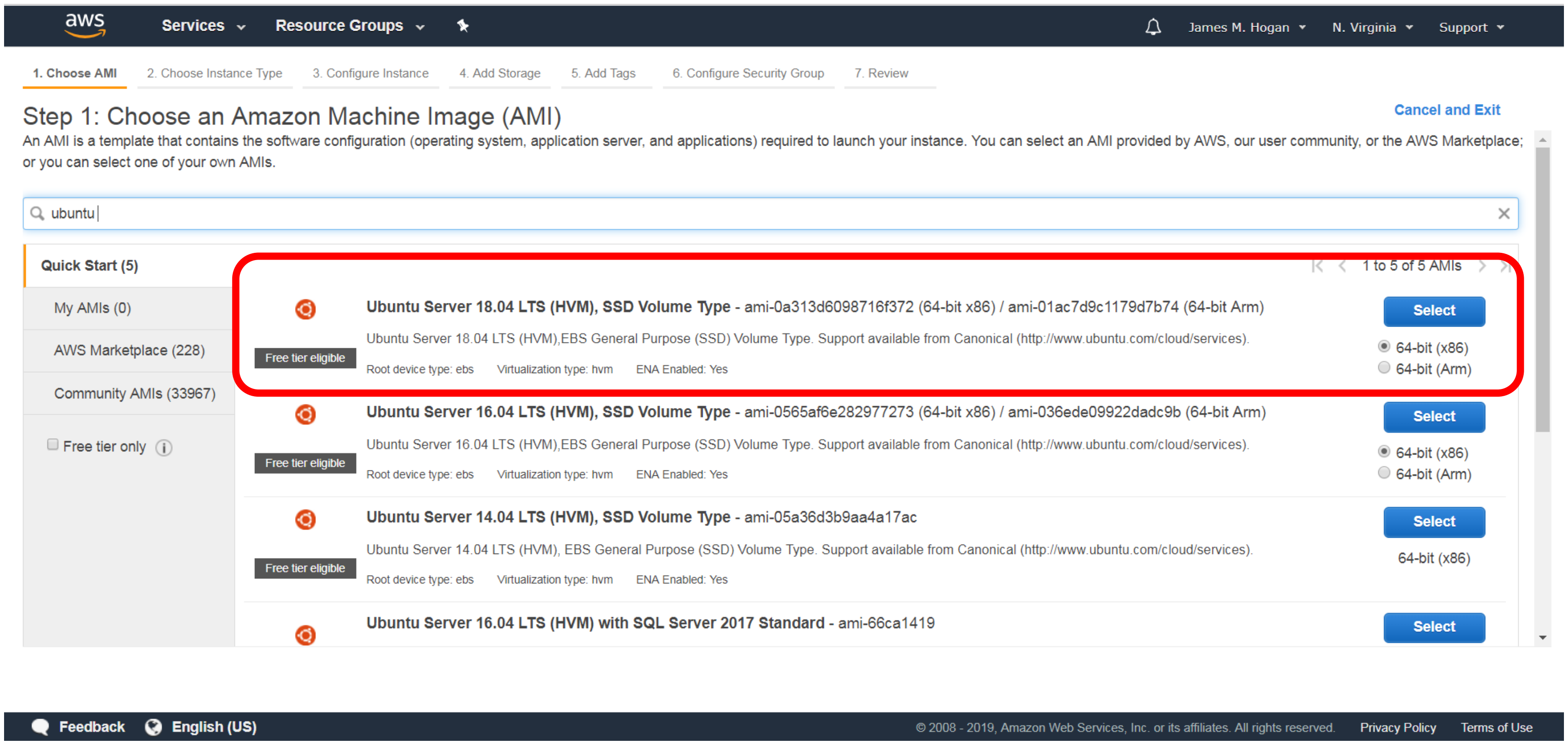
1 domain in cart Checkout

cab230 .cf • FREE	USD 0.00 ✓ Selected

Get one of these domains. They are free!

cab230 .tk • FREE	USD 0.00 Select

The VM



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

ubuntu

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0a313d6098716f372 (64-bit x86) / ami-01ac7d9c1179d7b74 (64-bit Arm)

Ubuntu Server 18.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Free tier eligible — Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select — [✓ 64-bit (x86)] 64-bit (Arm)

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-0565af6e282977273 (64-bit x86) / ami-036ede09922dadc9b (64-bit Arm)

Ubuntu Server 16.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Free tier eligible — Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select — [✓ 64-bit (x86)] 64-bit (Arm)

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-05a36d3b9aa4a17ac

Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Free tier eligible — Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

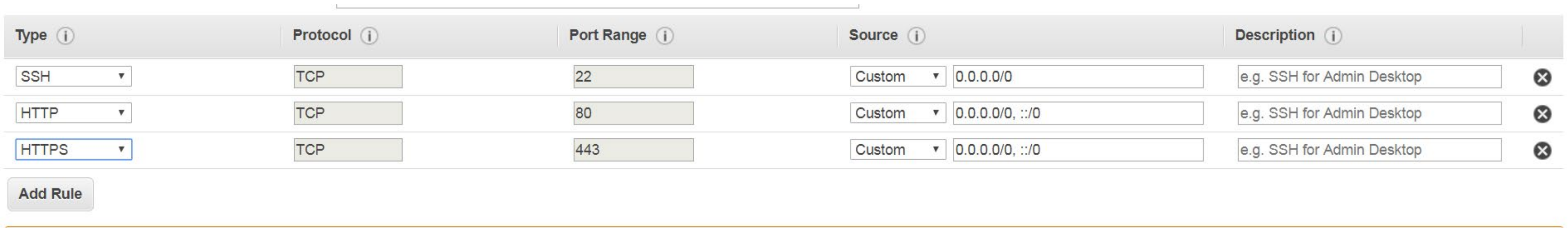
Select — 64-bit (x86)

Ubuntu Server 16.04 LTS (HVM) with SQL Server 2017 Standard - ami-66ca1419

Select

Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

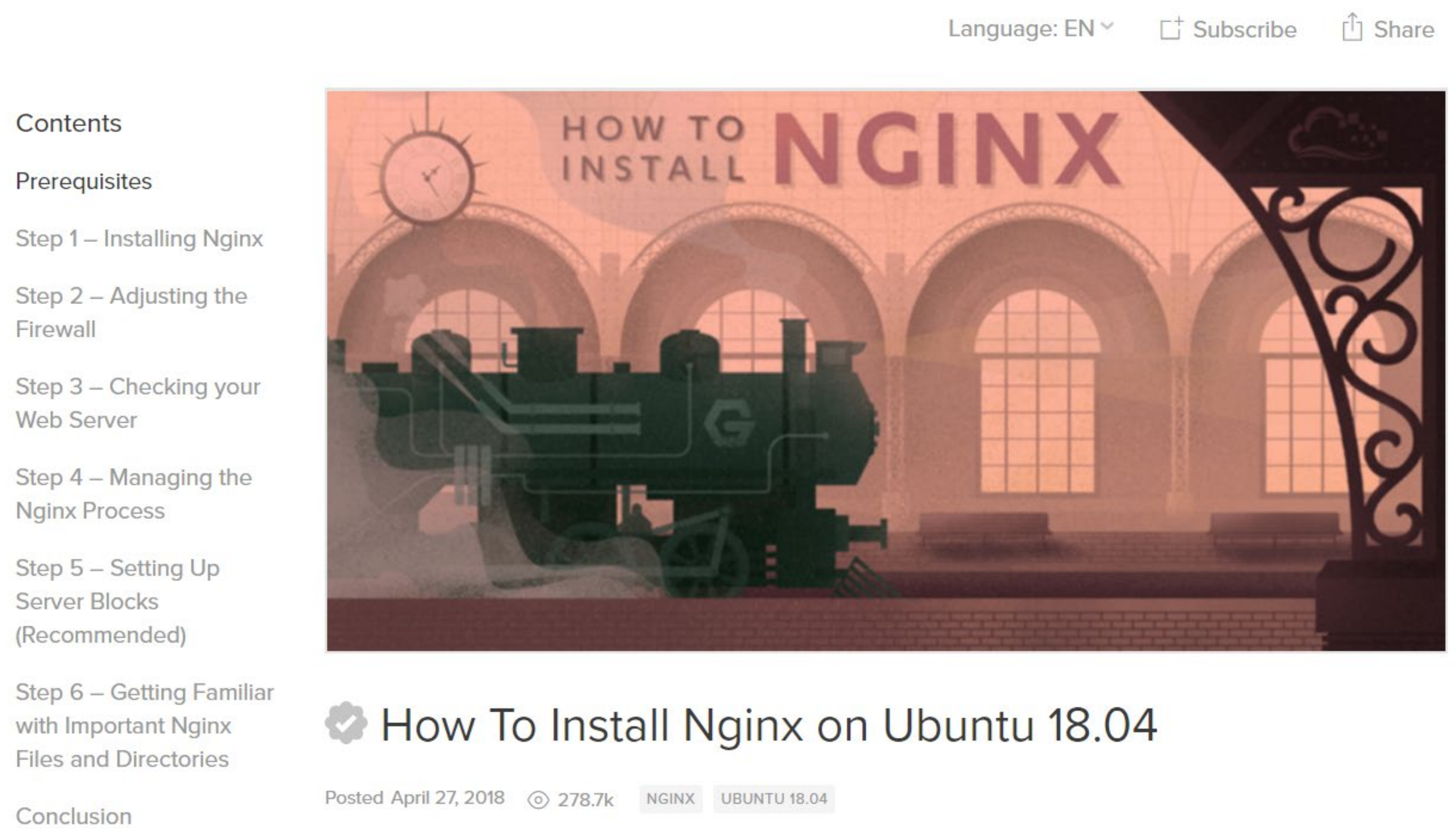
Access



Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Custom	0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

- Allow access for standard methods
- HTTP on port 80
- HTTPS on port 443
- (SSH on 22 for login to the machine)

NGINX Installation


Language: EN Subscribe Share

Contents

Prerequisites

Step 1 – Installing Nginx

Step 2 – Adjusting the Firewall

Step 3 – Checking your Web Server

Step 4 – Managing the Nginx Process

Step 5 – Setting Up Server Blocks (Recommended)

Step 6 – Getting Familiar with Important Nginx Files and Directories

Conclusion

How To Install Nginx on Ubuntu 18.04

Posted April 27, 2018 278.7k NGINX UBUNTU 18.04

https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04

NGINX Highlights

- Much more configuration file work than node
- More like Apache or other web servers
- Simple installation from the package manager:

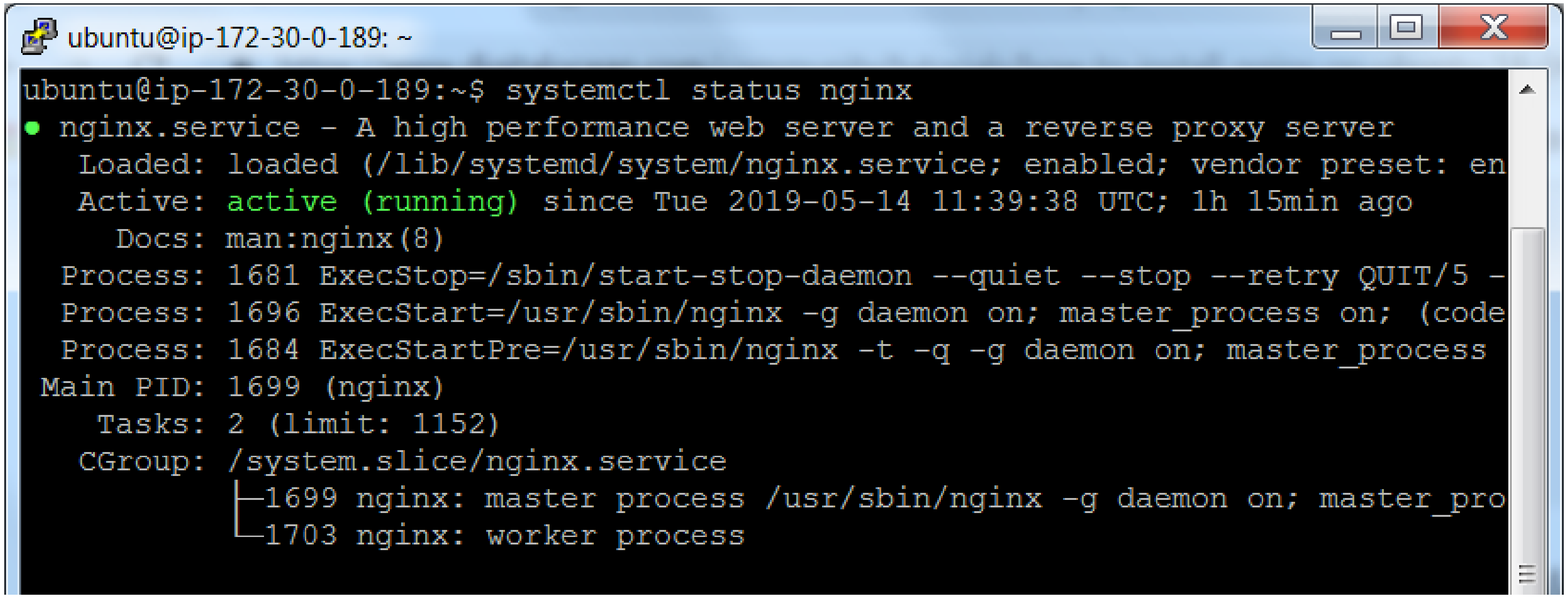
```
$ sudo apt update
$ sudo apt install nginx
```

- Adjust the firewall – we will be permissive:

```
$ sudo ufw allow 'Nginx Full'
```

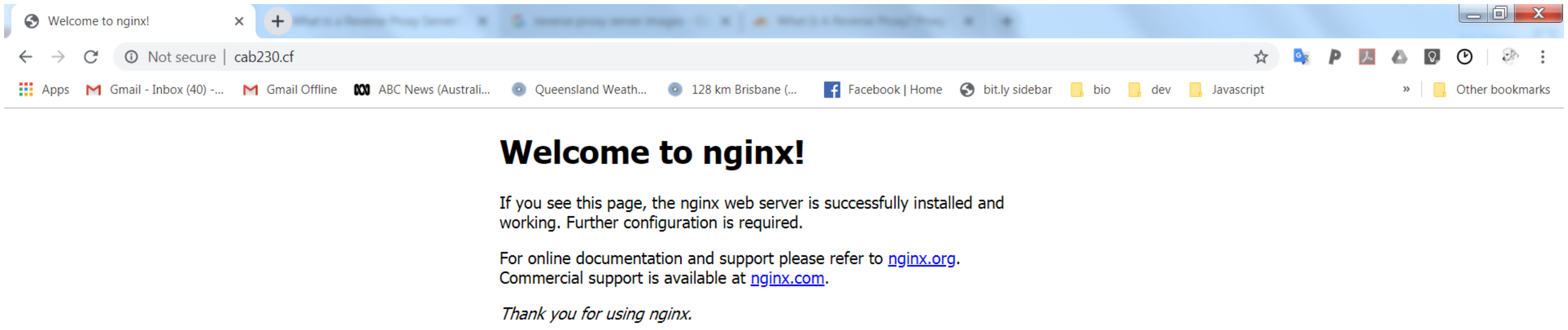
- We have associated `cab230.cf` with the VM IP

NGINX status


ubuntu@ip-172-30-0-189: ~

```
ubuntu@ip-172-30-0-189:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: en
   Active: active (running) since Tue 2019-05-14 11:39:38 UTC; 1h 15min ago
     Docs: man:nginx(8)
  Process: 1681 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 -
  Process: 1696 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code
  Process: 1684 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process
 Main PID: 1699 (nginx)
    Tasks: 2 (limit: 1152)
   CGroup: /system.slice/nginx.service
           ├─1699 nginx: master process /usr/sbin/nginx -g daemon on; master_pro
           └─1703 nginx: worker process
```

Simple Load



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.

Server Blocks

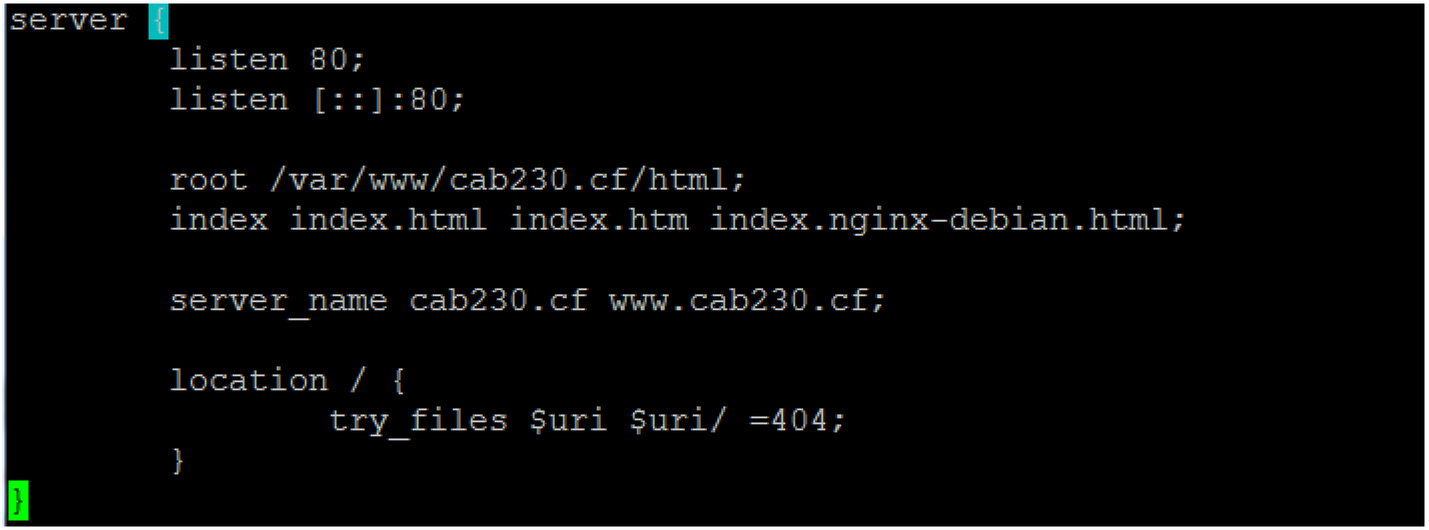
- Host additional servers using server blocks

  `/var/www/cab230.cf/html`

- Make them known in the sites-available

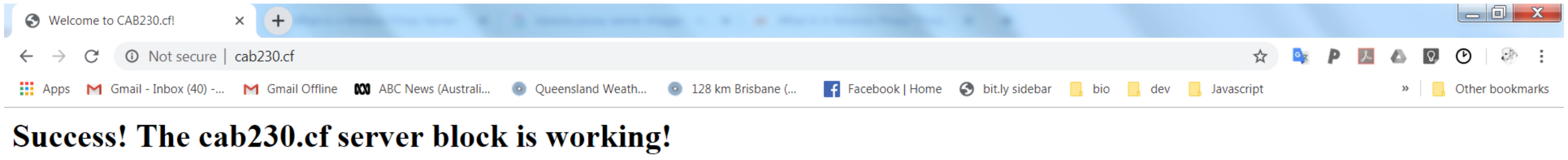
  `/etc/nginx/sites-available/cab230.cf`

- Need to adjust the config:

- 

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/cab230.cf/html;
        index index.html index.htm index.nginx-debian.html;

        server_name cab230.cf www.cab230.cf;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Updated request

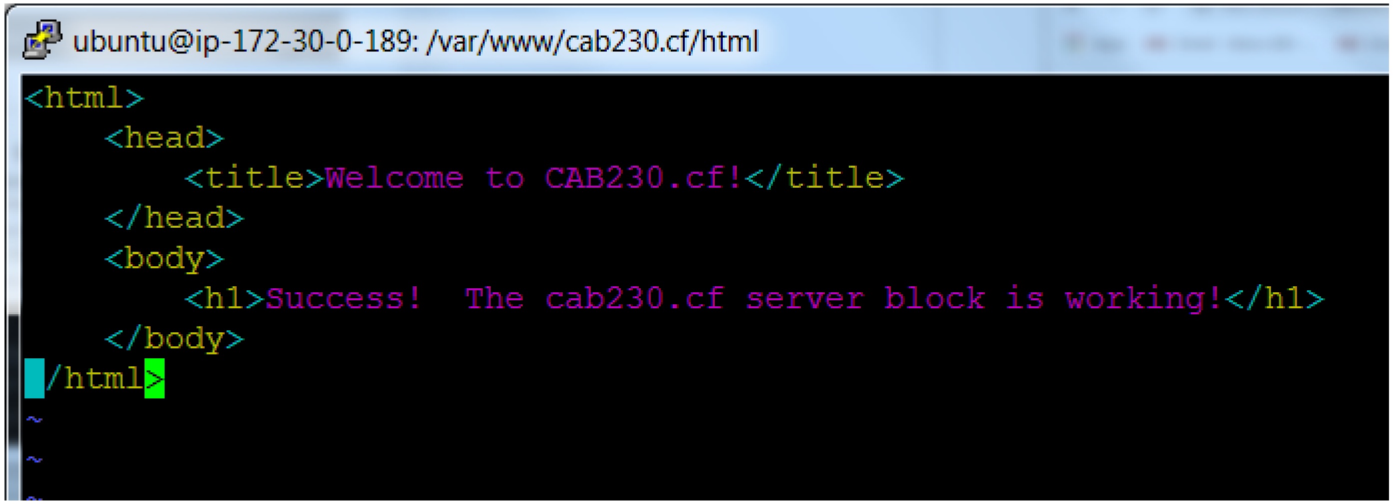


Welcome to CAB230.cf!

Not secure | cab230.cf

Apps Gmail - Inbox (40) -... Gmail Offline ABC News (Australi... Queensland Weath... 128 km Brisbane (... Facebook | Home bit.ly sidebar bio dev Javascript Other bookmarks

Success! The cab230.cf server block is working!



ubuntu@ip-172-30-0-189: /var/www/cab230.cf/html

```
<html>
    <head>
        <title>Welcome to CAB230.cf!</title>
    </head>
    <body>
        <h1>Success!  The cab230.cf server block is working!</h1>
    </body>
</html>
~
~
```

Securing NGINX

Mark as Complete

How To Secure Nginx with Let's Encrypt on Ubuntu 18.04

https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-18-04

SSL/TLS Certificates

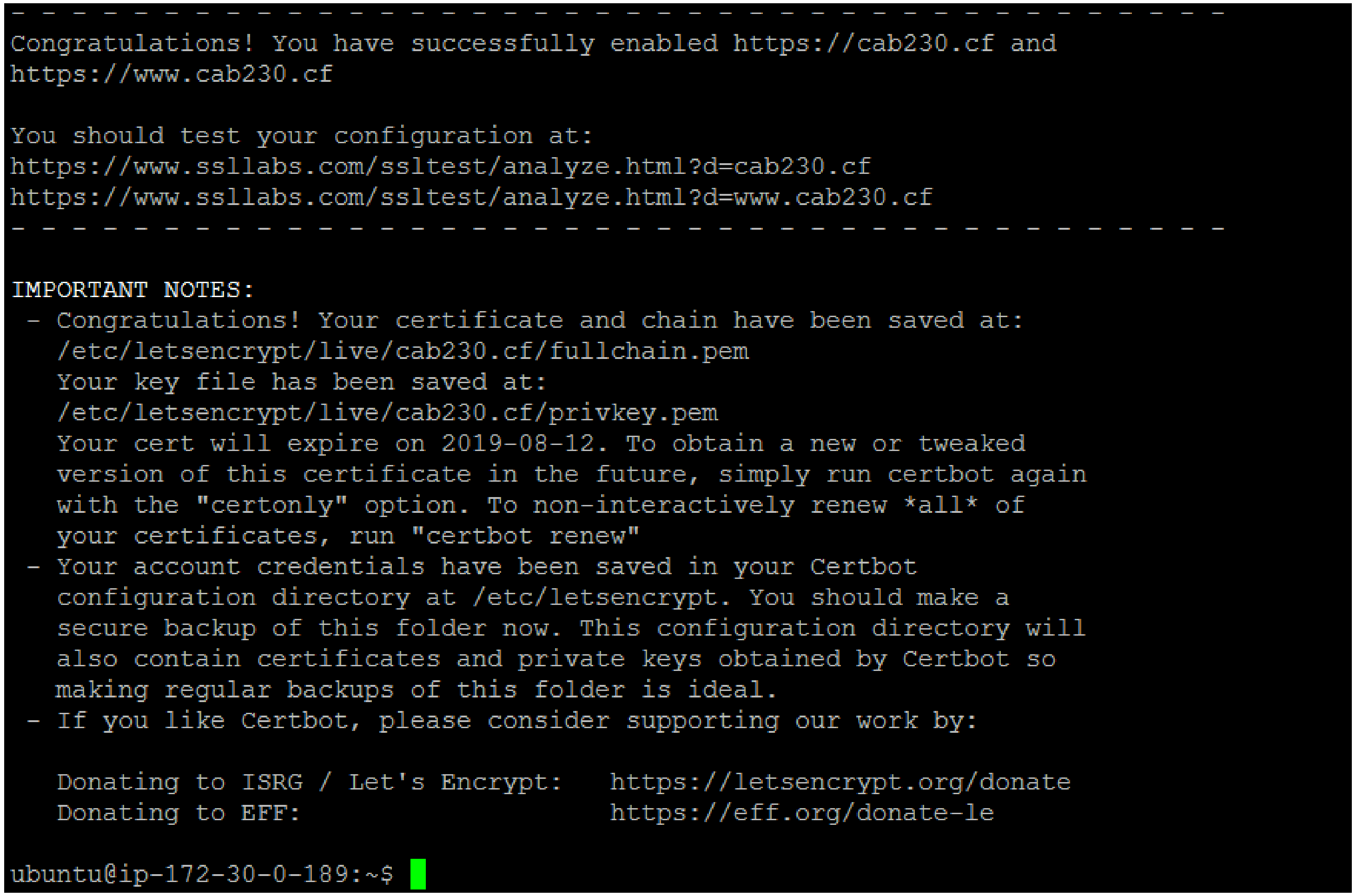
- This workflow can also be applied directly to node
- Here the Certificate Signing Request is real
- The CA is Let’s Encrypt and this is genuinely trusted
- We install certbot to manage the process
- See https://certbot.eff.org/

```
$ sudo add-apt-repository ppa:certbot/certbot

$ sudo apt install python-certbot-nginx
```

- We then install the new certificate (painlessly)

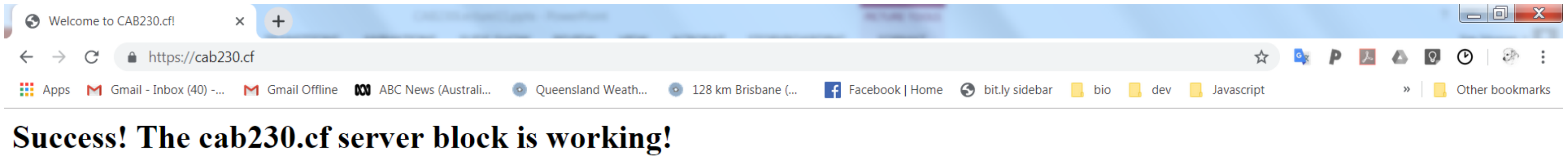
```
$ sudo certbot --nginx -d cab230.cf -d www.cab230.cf
```

Installation



```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Congratulations! You have successfully enabled https://cab230.cf and
https://www.cab230.cf

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=cab230.cf
https://www.ssllabs.com/ssltest/analyze.html?d=www.cab230.cf
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/cab230.cf/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/cab230.cf/privkey.pem
   Your cert will expire on 2019-08-12. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le

ubuntu@ip-172-30-0-189:~$
```

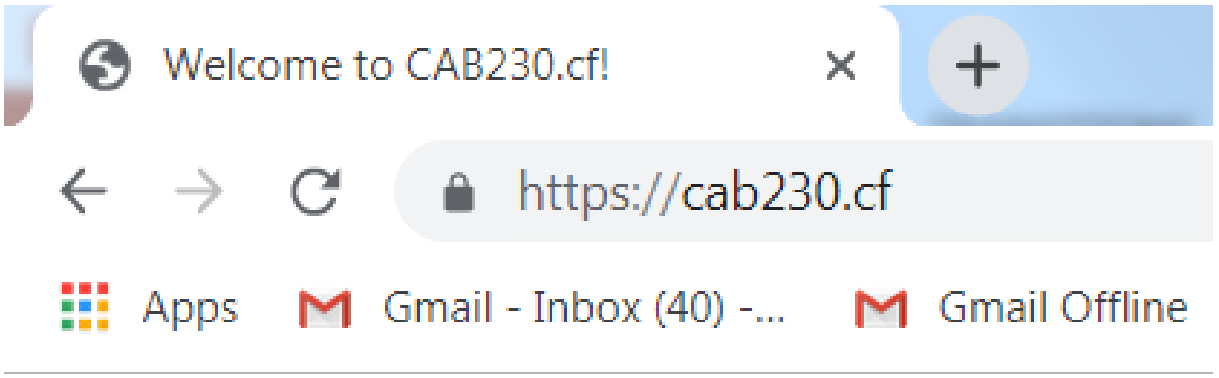
Secure


Welcome to CAB230.cf!

https://cab230.cf

Apps Gmail - Inbox (40) -... Gmail Offline ABC News (Australi... Queensland Weath... 128 km Brisbane (... Facebook | Home bit.ly sidebar bio dev Javascript Other bookmarks

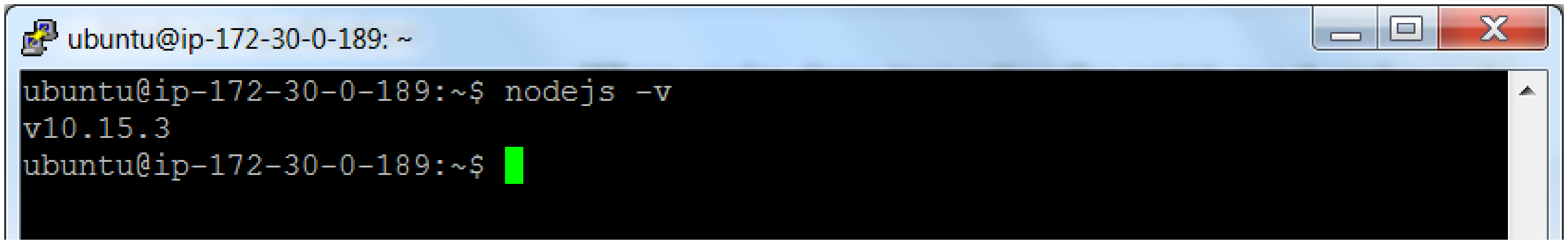
Success! The cab230.cf server block is working!



And certbot automates the certificate renewal process.

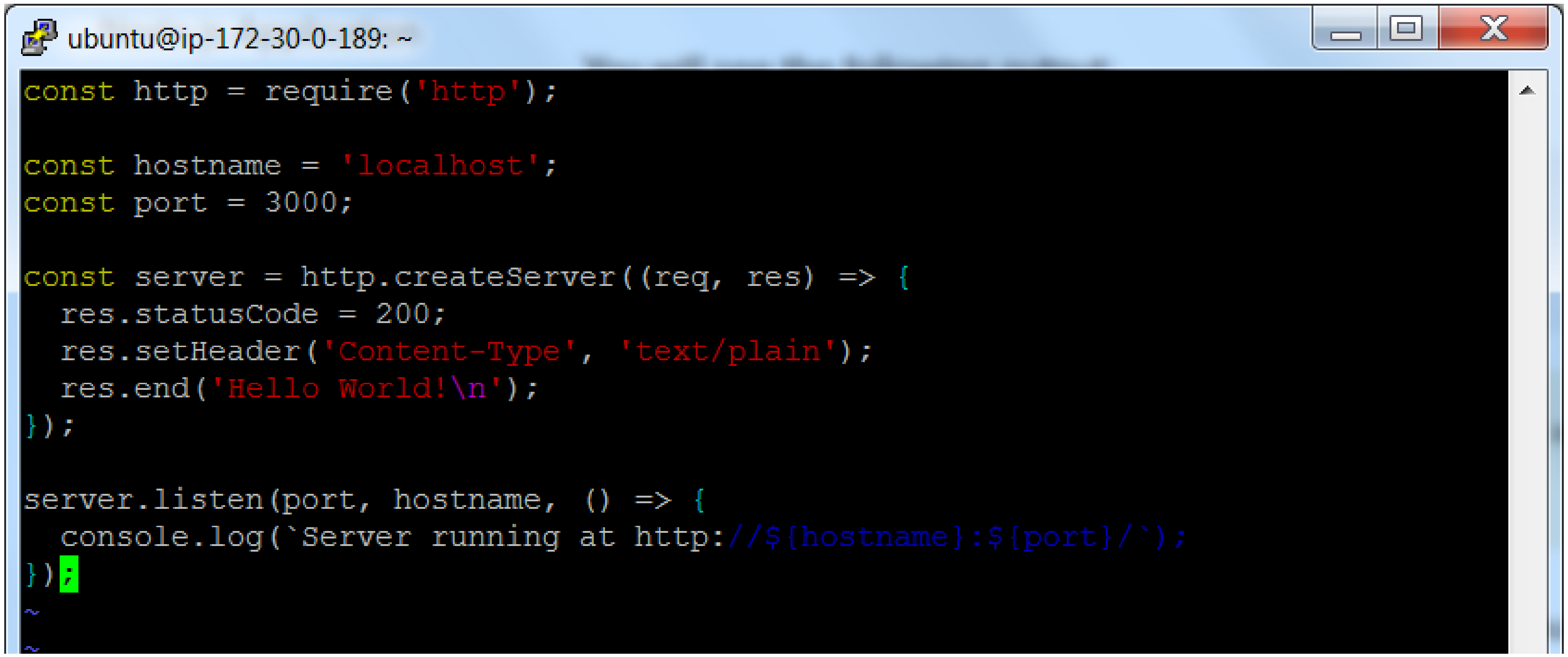
Node Install

- Node installation instructions for linux are in the guide.
- This is version 10 from 2019
- We grab from the repo and use a script and then the package manager
- Finally we verify that node is there.



```
ubuntu@ip-172-30-0-189:~$ nodejs -v
v10.15.3
ubuntu@ip-172-30-0-189:~$
```

The Hello World Server (again)


ubuntu@ip-172-30-0-189: ~

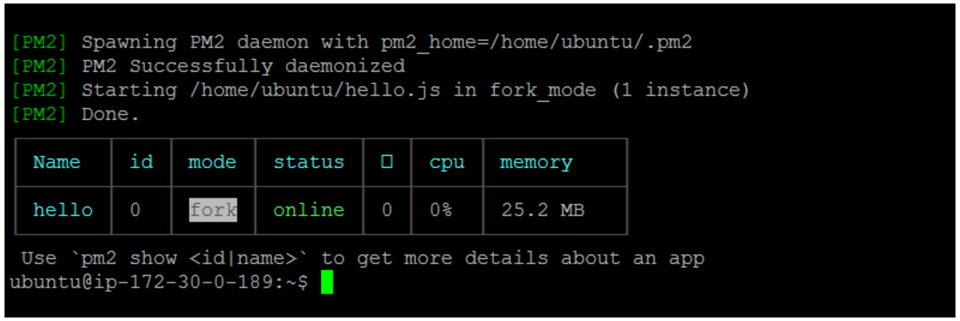
```
const http = require('http');

const hostname = 'localhost';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World!\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
~
~
```

Process Managing Node apps

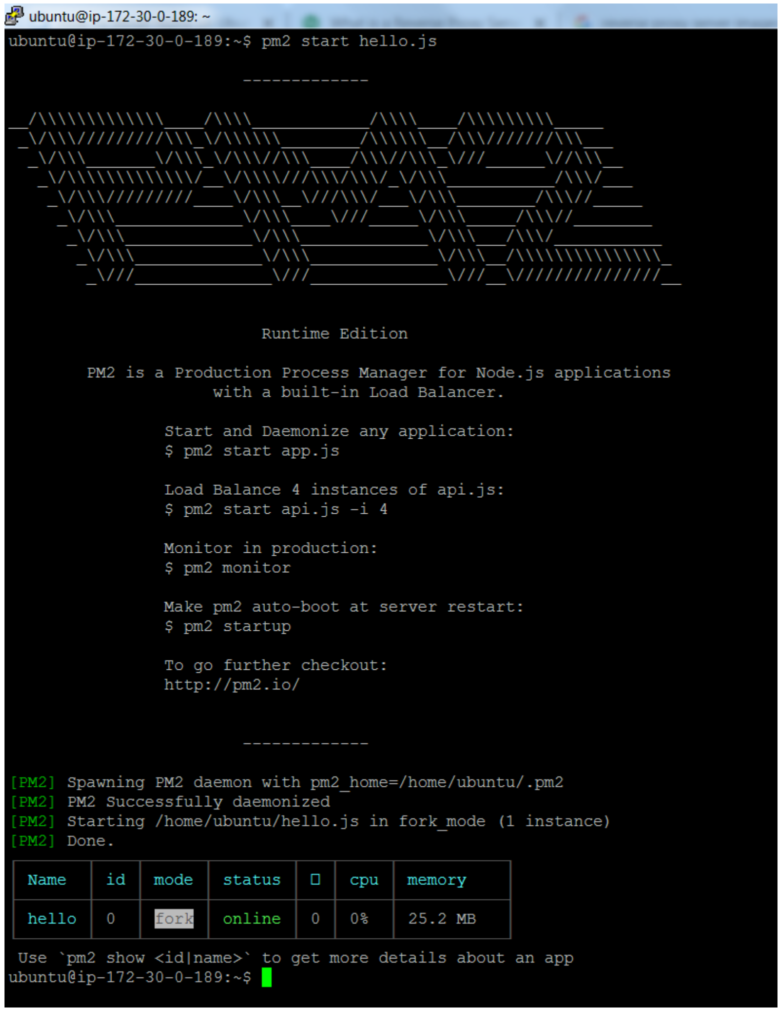
Install the process manager PM2 globally

```
$ sudo npm install pm2@latest -g
```

The output is wild but it daemonises the app:



```
[PM2] Spawning PM2 daemon with pm2_home=/home/ubuntu/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ubuntu/hello.js in fork_mode (1 instance)
[PM2] Done.
┌──────┬────┬──────┬────────┬───┬─────┬─────────┐
│ Name │ id │ mode │ status │ ↺ │ cpu │ memory  │
├──────┼────┼──────┼────────┼───┼─────┼─────────┤
│ hello│ 0  │ fork │ online │ 0 │ 0%  │ 25.2 MB │
└──────┴────┴──────┴────────┴───┴─────┴─────────┘
 Use `pm2 show <id|name>` to get more details about an app
ubuntu@ip-172-30-0-189:~$
```



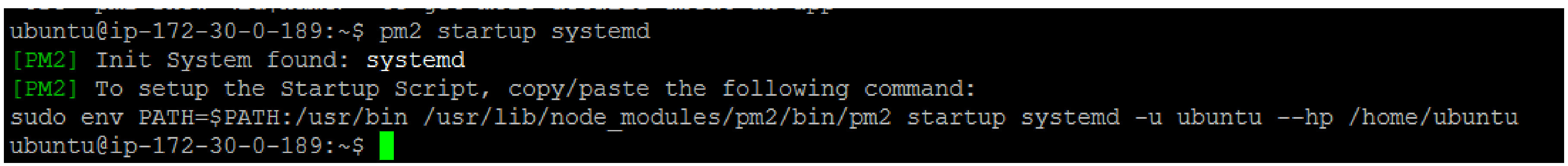
ubuntu@ip-172-30-0-189: ~

```
ubuntu@ip-172-30-0-189:~$ pm2 start hello.js

                        -------------

__/\\\\\\\\\\\\\____/\\\\____________/\\\\____/\\\\\\\\\_____
 _\/\\\/////////\\\_\/\\\\\\________/\\\\\\__/\\\///////\\\___
  _\/\\\_______\/\\\_\/\\\//\\\____/\\\//\\\_\///______\//\\\__
   _\/\\\\\\\\\\\\\/__\/\\\\///\\\/\\\/_\/\\\___________/\\\/___
    _\/\\\/////////____\/\\\__\///\\\/___\/\\\________/\\\//_____
     _\/\\\_____________\/\\\____\///_____\/\\\_____/\\\//________
      _\/\\\_____________\/\\\_____________\/\\\___/\\\/___________
       _\/\\\_____________\/\\\_____________\/\\\__/\\\\\\\\\\\\\\\_
        _\///______________\///______________\///__\///////////////__


                          Runtime Edition

        PM2 is a Production Process Manager for Node.js applications
                     with a built-in Load Balancer.

                Start and Daemonize any application:
                $ pm2 start app.js

                Load Balance 4 instances of api.js:
                $ pm2 start api.js -i 4

                Monitor in production:
                $ pm2 monitor

                Make pm2 auto-boot at server restart:
                $ pm2 startup

                To go further checkout:
                http://pm2.io/


                        -------------

[PM2] Spawning PM2 daemon with pm2_home=/home/ubuntu/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ubuntu/hello.js in fork_mode (1 instance)
[PM2] Done.
┌──────┬────┬──────┬────────┬───┬─────┬─────────┐
│ Name │ id │ mode │ status │ ↺ │ cpu │ memory  │
├──────┼────┼──────┼────────┼───┼─────┼─────────┤
│ hello│ 0  │ fork │ online │ 0 │ 0%  │ 25.2 MB │
└──────┴────┴──────┴────────┴───┴─────┴─────────┘
 Use `pm2 show <id|name>` to get more details about an app
ubuntu@ip-172-30-0-189:~$
```

Process Managing Node apps

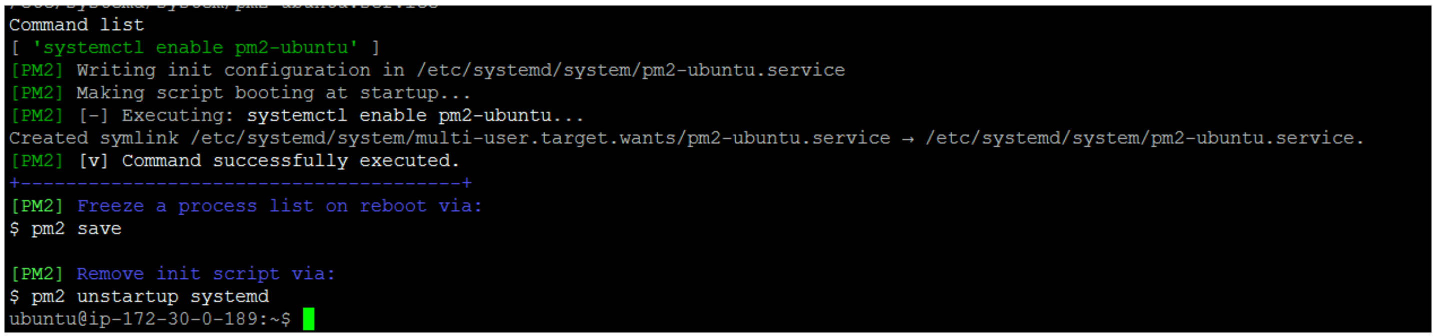
- We can also go ahead and create a startup script
- This allows the server to start on boot

```
$ pm2 startup systemd
```



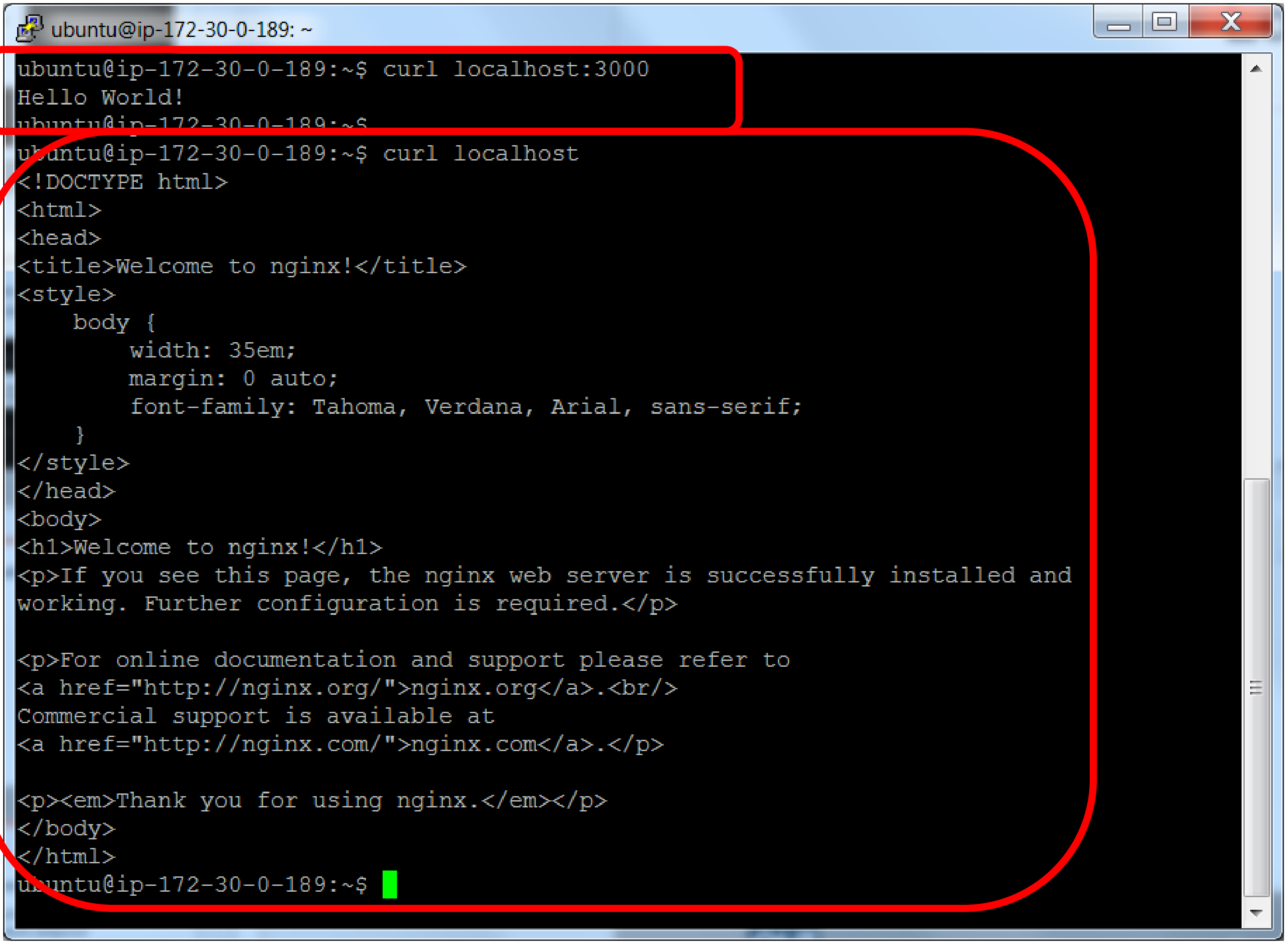
```
ubuntu@ip-172-30-0-189:~$ pm2 startup systemd
[PM2] Init System found: systemd
[PM2] To setup the Startup Script, copy/paste the following command:
sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -u ubuntu --hp /home/ubuntu
ubuntu@ip-172-30-0-189:~$
```

```
$ sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2
startup systemd -u ubuntu --hp /home/ubuntu
```



```
Command list
[ 'systemctl enable pm2-ubuntu' ]
[PM2] Writing init configuration in /etc/systemd/system/pm2-ubuntu.service
[PM2] Making script booting at startup...
[PM2] [-] Executing: systemctl enable pm2-ubuntu...
Created symlink /etc/systemd/system/multi-user.target.wants/pm2-ubuntu.service → /etc/systemd/system/pm2-ubuntu.service.
[PM2] [v] Command successfully executed.
+---------------------------------------+
[PM2] Freeze a process list on reboot via:
$ pm2 save

[PM2] Remove init script via:
$ pm2 unstartup systemd
ubuntu@ip-172-30-0-189:~$
```

localhost:3000 and localhost

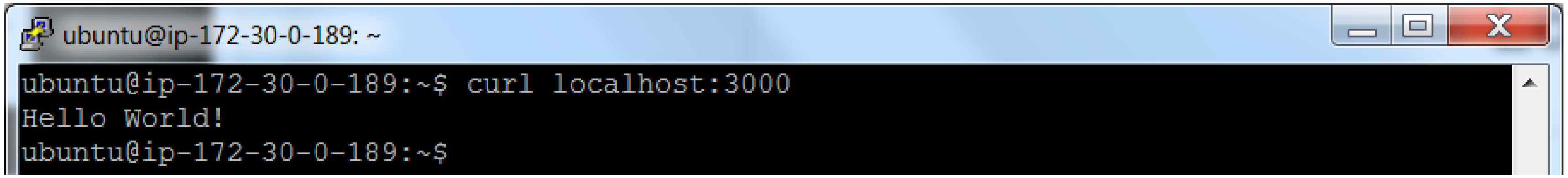


ubuntu@ip-172-30-0-189: ~

```
ubuntu@ip-172-30-0-189:~$ curl localhost:3000
Hello World!
ubuntu@ip-172-30-0-189:~$
ubuntu@ip-172-30-0-189:~$ curl localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
ubuntu@ip-172-30-0-189:~$
```

Access

- But the node server is not internet-facing
- So we need to provide access to it.
- We will do this using a reverse proxy approach
- The update is much simpler than one might expect



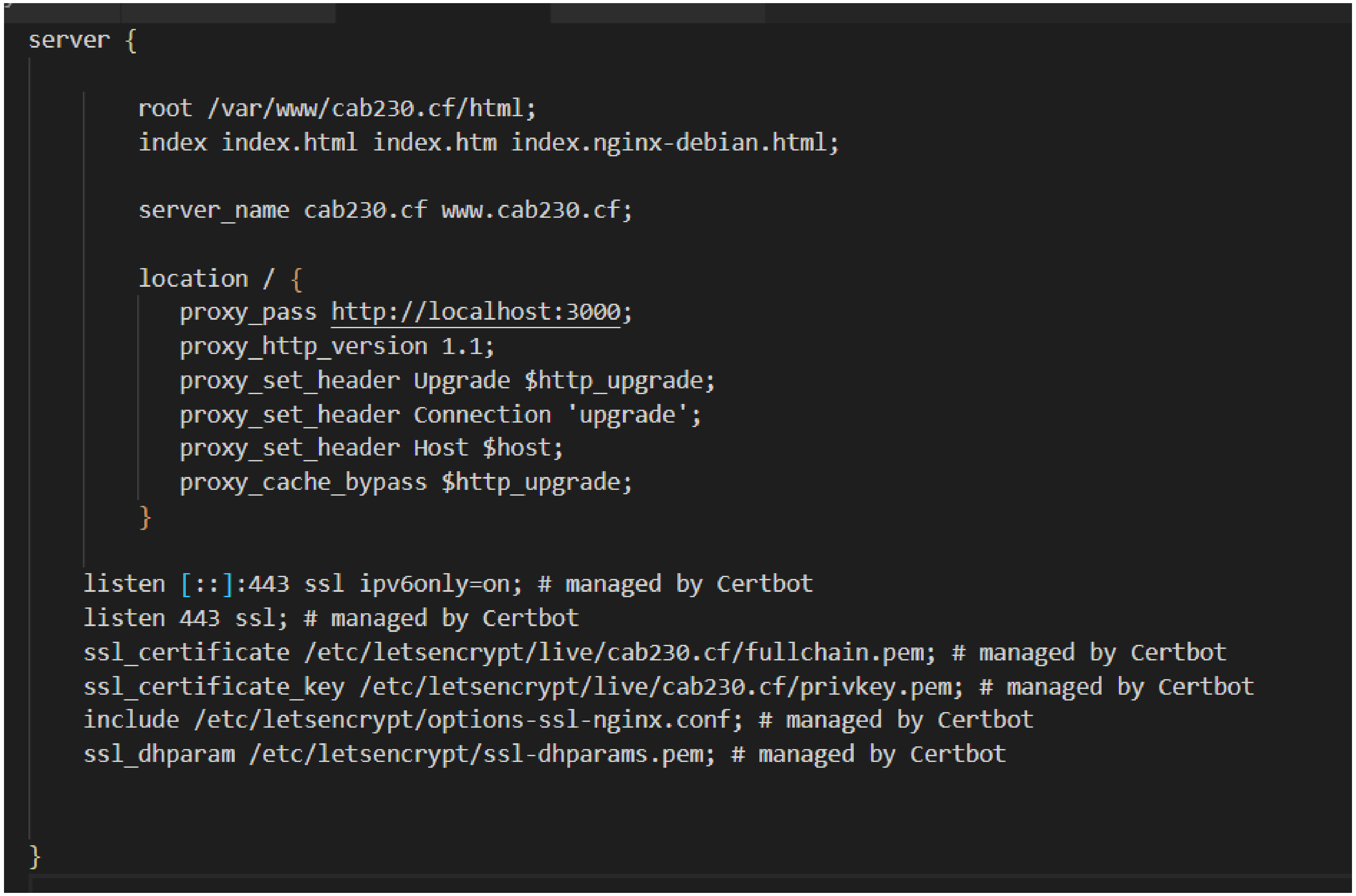
```
ubuntu@ip-172-30-0-189:~$ curl localhost:3000
Hello World!
ubuntu@ip-172-30-0-189:~$
```

Access

- We simply update the location block in the server config
  - We will handle the site root: `location / {…}`
  - This will point to http://localhost:3000
  - We upgrade the headers
  - We reset the host, and serve HelloWorld via HTTPS

- Other apps, say `https://cab230.cf/search`
  - Handle: `location /search {…}`
  - Have this point to say http://localhost:8000
  - Handled by *another* node server app

Excerpt from the sites-available file



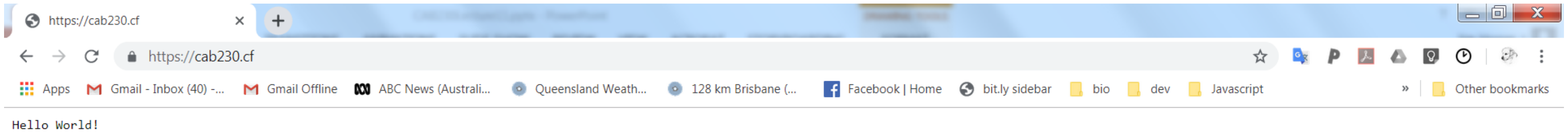
```
server {

        root /var/www/cab230.cf/html;
        index index.html index.htm index.nginx-debian.html;

        server_name cab230.cf www.cab230.cf;

        location / {
           proxy_pass http://localhost:3000;
           proxy_http_version 1.1;
           proxy_set_header Upgrade $http_upgrade;
           proxy_set_header Connection 'upgrade';
           proxy_set_header Host $host;
           proxy_cache_bypass $http_upgrade;
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/cab230.cf/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/cab230.cf/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


}
```

https://cab230.cf

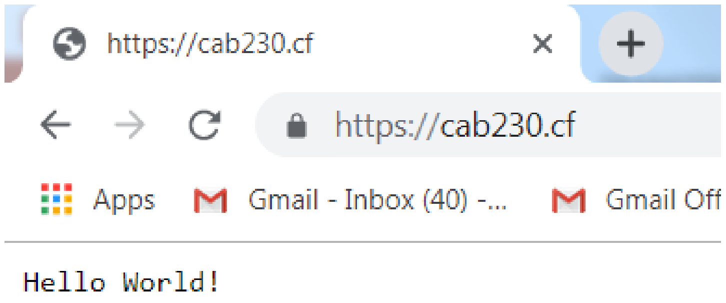
- We restart the NGINX server:

```
$ sudo systemctl restart nginx
```

- And, rather absurdly, we are ready to go:



```
Hello World!
```



```
Hello World!
```

And we achieve a securely deployed Hello World ☺ via reverse proxy…

Some Final Points

- Reverse proxies can handle numerous apps
  - Add more location blocks and handle different routes
  - Use the upstream config to manage load balancing

- And enhance application performance:

  SSL encryption and gzip compression are two highly CPU-bound operations. Dedicated reverse proxy tools, like Nginx and HAProxy, typically perform these operations faster than Node.js. Having a web server like Nginx read static content from disk is going to be faster than Node.js as well. Even clustering can sometimes be more efficient as a reverse proxy like Nginx will use less memory and CPU than that of an additional Node.js process.

  - See Hunter’s blog for supporting benchmarks
  - These are especially convincing for memory usage.

References

- Blog on use of NGINX and Reverse Proxies
  - Superb general and node-specific background
  - https://medium.com/intrinsic/why-should-i-use-a-reverse-proxy-if-node-js-is-production-ready-5a079408b2ca
- Digital Ocean Guides:
  - Installation of NGINX on Ubuntu
  - https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-18-04
  - Securing NGINX
  - https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-18-04
  - Setting up node for production under NGINX
  - https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-ubuntu-18-04

Next week

- Advanced, non-examinable JS ☺