NGINX POWERS 1 IN 3

of the world's busiest sites and applications—
from Airbnb to Netflix to Uber.

NGINX 101

Now with more Docker

THE MOST INNOVATIVE DEVELOPERS

have chosen NGINX to deliver their apps to the world.

NGINX

Core NGINX functionality includes HTTP request, proxy and caching services which can be combined into a complete application delivery platform. Or, as we like to think of it....

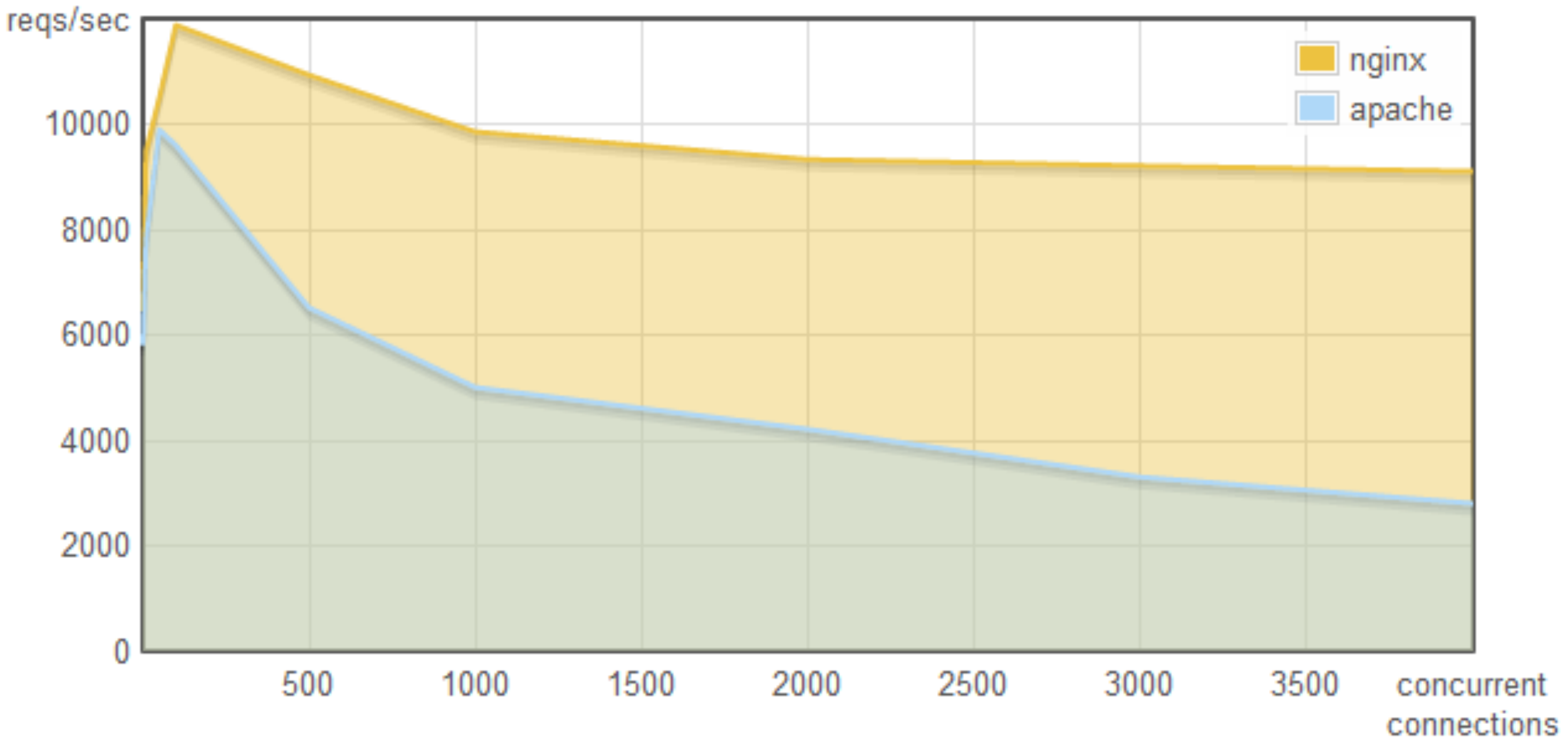
THE SECRET HEART OF THE MODERN WEB

The origins

NGINX development began at Rambler.ru by Igor Sysoev to solve c10k problem

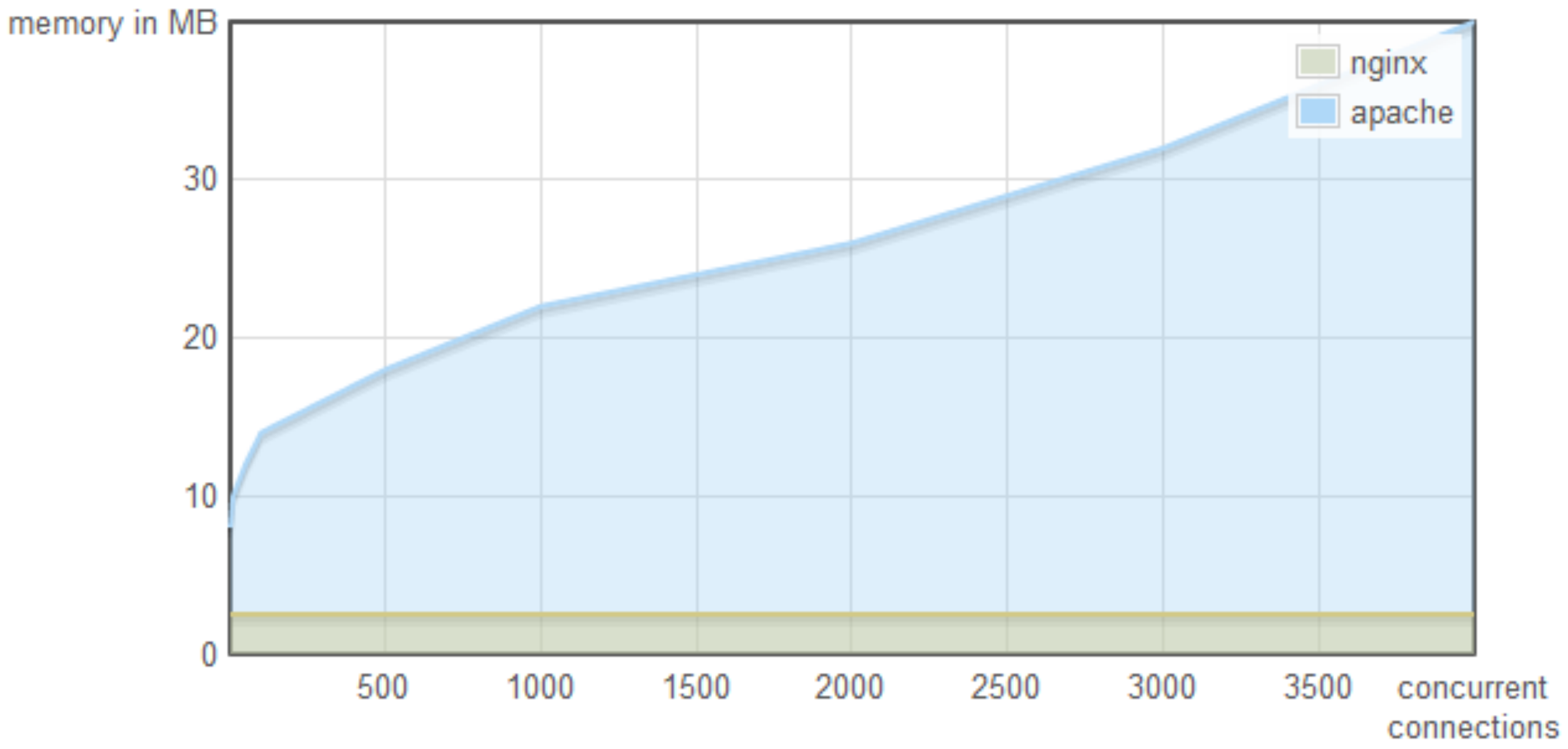
- High concurrency
- Low memory use
- 2002 commodity hardware

High Concurrency



Source: Webfaction Blog: http://blog.webfaction.com/2008/12/a-little-holiday-present-10000-reqssec-with-nginx-2/

Low Memory Use



Source: Webfaction Blog: http://blog.webfaction.com/2008/12/a-little-holiday-present-10000-reqssec-with-nginx-2/

Apache is like Microsoft Word, it has a million options but you only need six. Nginx does those six things, and it does five of them 50 times faster than Apache.

- Chris Lea

Questions before you begin

1. What functionality do you require?

- Standard modules
- NGINX Plus functionality
- Optional NGINX and third-party modules

2. What branch do you want to track?

- Mainline (1.7)
- Stable (1.6)
- Something older?

3. How do you want to install?

- "Official" NGINX packages (nginx.org)
- Build from Source
- From Operating System repository
- From Amazon AWS Marketplace
- From Docker Hub Registry

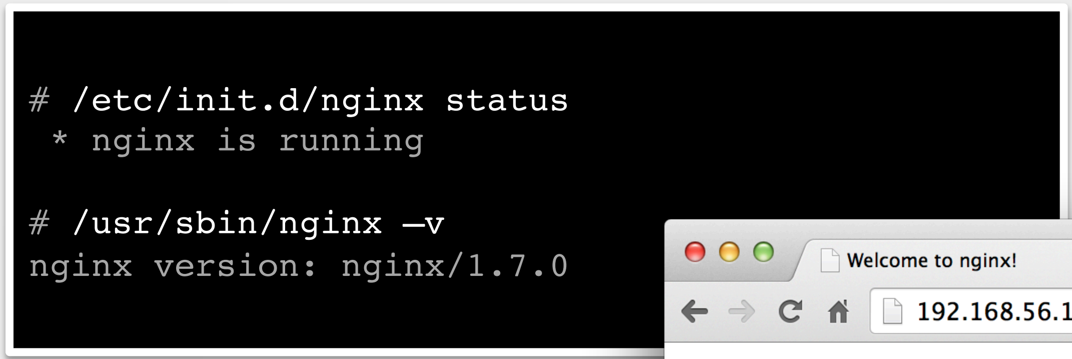
http://nginx.com/blog/nginx-1-6-1-7-released/

Traditional Installation

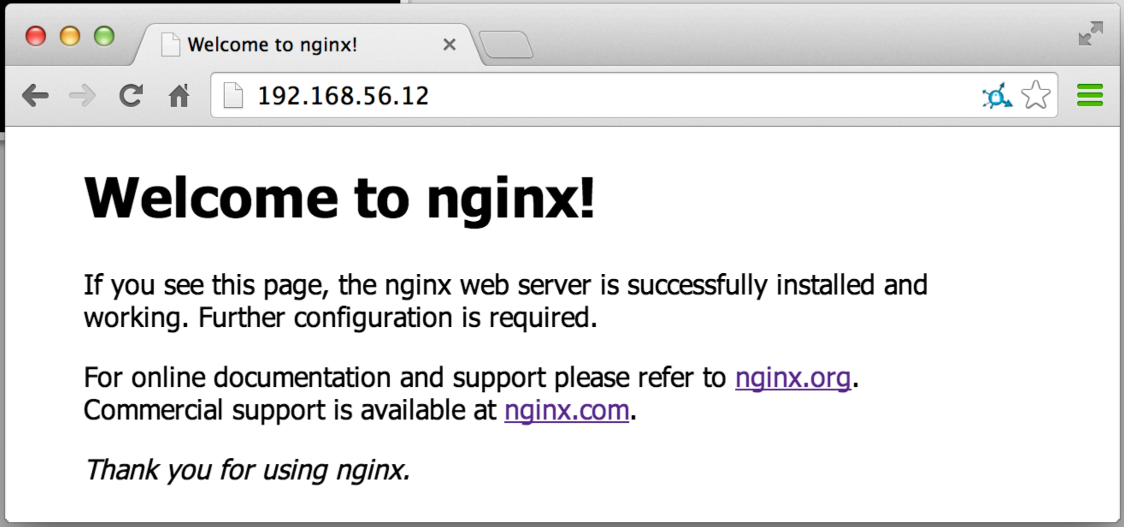
```
$ wget http://nginx.org/keys/nginx_signing.key
$ sudo apt-key add nginx_signing.key

# cat > /etc/apt/sources.list.d/nginx.list
deb http://nginx.org/packages/mainline/ubuntu/ trusty nginx
deb-src http://nginx.org/packages/mainline/ubuntu/ trusty nginx

# apt-get update
# apt-cache policy nginx
nginx:
  Installed: (none)
  Candidate: 1.7.0-1~trusty
  Version table:
     1.7.0-1~trusty 0
        500 http://nginx.org/packages/mainline/ubuntu/ trusty/nginx amd64 Packages
     1.4.6-1ubuntu3 0
        500 http://us.archive.ubuntu.com/ubuntu/ trusty/main amd64 Packages
```

http://nginx.org/en/linux_packages.html#mainline

Verify it's working



```
# /etc/init.d/nginx status
 * nginx is running

# /usr/sbin/nginx -v
nginx version: nginx/1.7.0
```



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.

The basics of the install

Where are the things

- NGINX executable is at /usr/sbin/nginx
- Configuration files at /etc/nginx
- Log files at /var/log/nginx

NGINX processes

- One **master** process and many **worker** processes
- The master process evaluates the configuration file and manages the worker processes
- Worker processes handle actual requests

```
[root@localhost ~]# ps -ef |grep nginx
root      1991     1  0 08:06 ?        00:00:00 nginx: master
process /usr/sbin/nginx -c /etc/nginx/nginx.conf
nginx     2974  1991  0 08:22 ?        00:00:00 nginx: worker
process
nginx     2975  1991  0 08:22 ?        00:00:00 nginx: worker
process
```

Basic NGINX commands

- To start NGINX, simply run the executable file at /usr/sbin/nginx
- The executable can be run with a "-s" parameter followed by a signal.

Reload configuration

```
nginx -s reload
```

Graceful shutdown. NGINX will wait for workers to finish processing requests

```
nginx -s quit
```

Fast shutdown

```
nginx -s stop
```

The NGINX configuration file

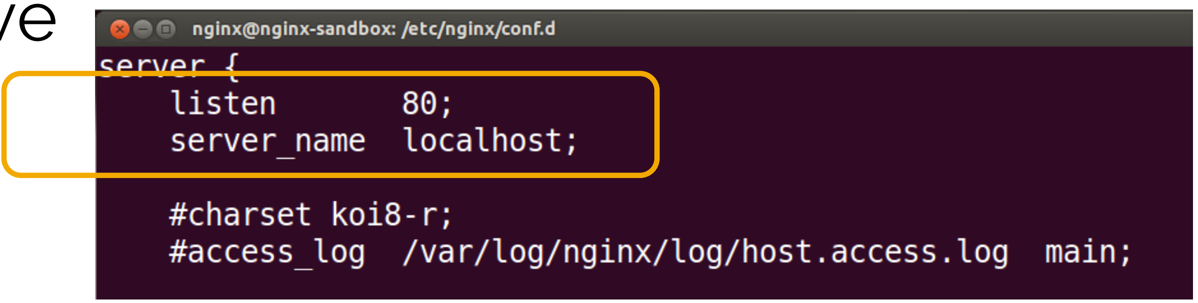
- The configuration file determines how NGINX and its modules behave
- The main file is named **nginx.conf** and is located in **/etc/nginx**
- The main configuration file may include references to additional configuration files
- Configuration consists of
  - Directives
  - Blocks
  - Contexts

Configuration directives

Directives

A **Directive** is a configuration statement that controls the behaviour of NGINX modules

- Consists of the directive name, followed by parameters and ends in a semicolon
- Two types of directives
  - Simple directive
  - Block directive

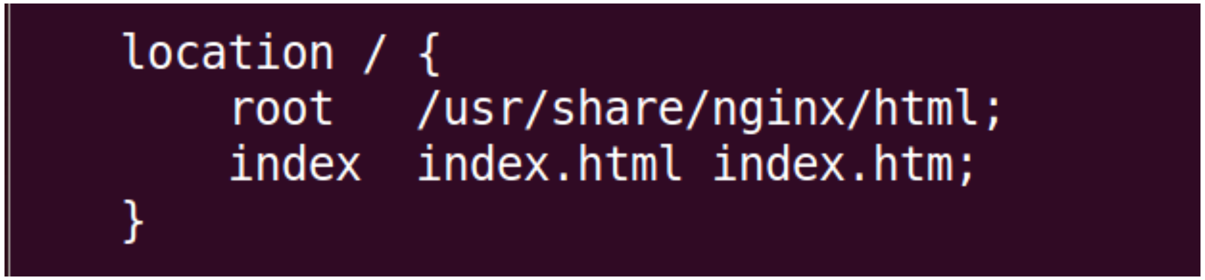


```
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;
    #access_log  /var/log/nginx/log/host.access.log  main;
```

Block Directives

A **Block Directive** is a directive that contains multiple configuration instructions

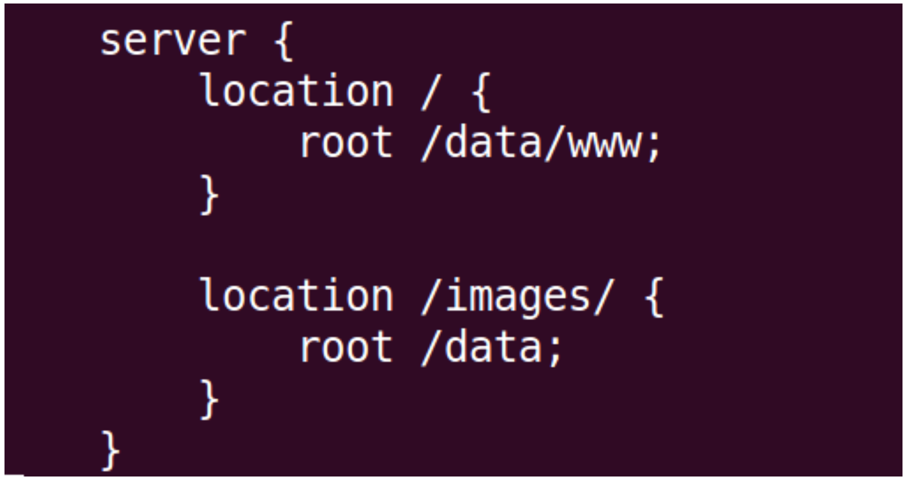
- The configurations instructions inside a block directive are surrounded by braces (i.e { })



```
location / {
    root   /usr/share/nginx/html;
    index  index.html index.htm;
}
```

Context example

- Example of a Server context, which has two location blocks
- The server context here can also be referred to as a server block



```
server {
    location / {
        root /data/www;
    }

    location /images/ {
        root /data;
    }
}
```

Specify the Server Block

The **Server** block defines the configuration for a virtual server

- Goes inside the HTTP context
- Can contain a **listen** directive, **server_name** directive and **root** directive
- Can specify many server blocks
- Equivalent to VirtualHost in Apache

Specify the Server Block

The **Server** block defines the configuration for a virtual server

- NGINX will choose which server to process a request based on the server name and the listen port

```
Define a virtual server that listens for requests on port 80
http {
    server {
        listen 80;
    }
}
```

Location Block

- The **location block** defines the configuration that will apply based on a matching request URI

- Placed inside a **server** block
- Server **block** can contain many location blocks
- Can contain a **Root** directive, which will override the **Root** directive of the server
- Can be nested inside a **location** block
- Two types of location blocks

  Prefix location + Regex location

Example Server and Location

- **Root** directive sets the root directory for a request.
- A request to localhost:8080 will return the
- index.html file in /home/nginx/public_html

```
server {
    listen 8080;
    root /home/nginx/public_html;
    location /application1 {
    }
    location /images/ {
        root /data;
    }
}
```

The Include directive

- The include directive allows you to include additional configuration files
- **Syntax**: `include <path to file>;`
- **Best Practices:**
  - For each server, create a separate configuration file in /etc/nginx/conf.d
  - nginx.conf includes all files in the conf.d folder ending in .conf   by default

Defining server names

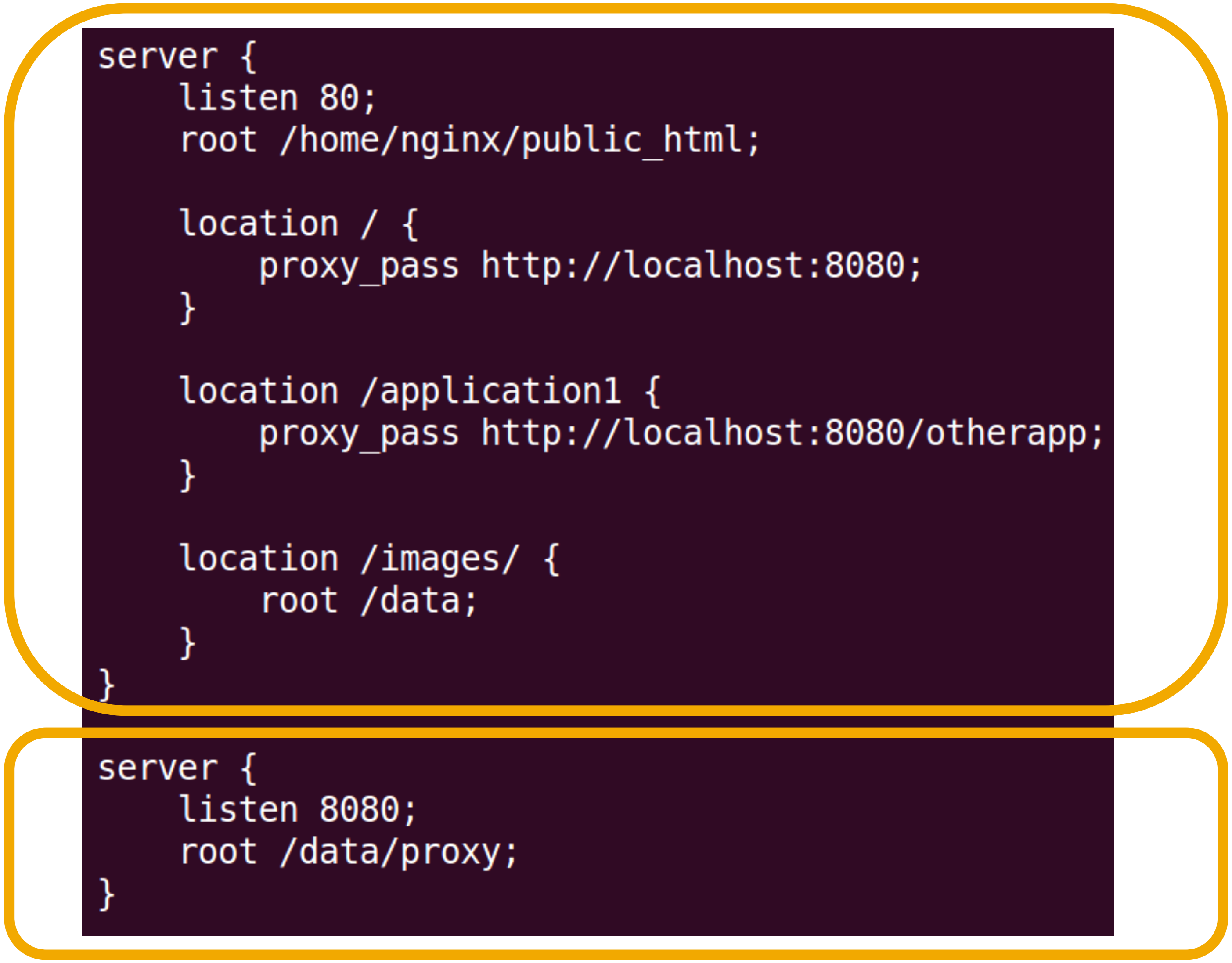
- Use the **server_name** directive in the server context to define the names for your server

```
server {
   server_name mycompany.com *.mycompany.com;
}
```

Simple Proxy Scenario

- **Server one** listening for requests on port 80 and serves content from /home/nginx/public_html
- **Server two** listens on port 8080 and serves content from /data/proxy
- Requests for localhost are proxied over to the server on port 8080

Simple Proxy Scenario



```
server {
    listen 80;
    root /home/nginx/public_html;

    location / {
        proxy_pass http://localhost:8080;
    }

    location /application1 {
        proxy_pass http://localhost:8080/otherapp;
    }

    location /images/ {
        root /data;
    }
}
server {
    listen 8080;
    root /data/proxy;
}
```

Logging

- The **error_log** directive can be used to configure the logging settings
- Syntax:
  `error_log <file> <log level>;`
- Can be used in the main, server, http and location contexts
- The Log level specifies how detailed the log output will be

Example

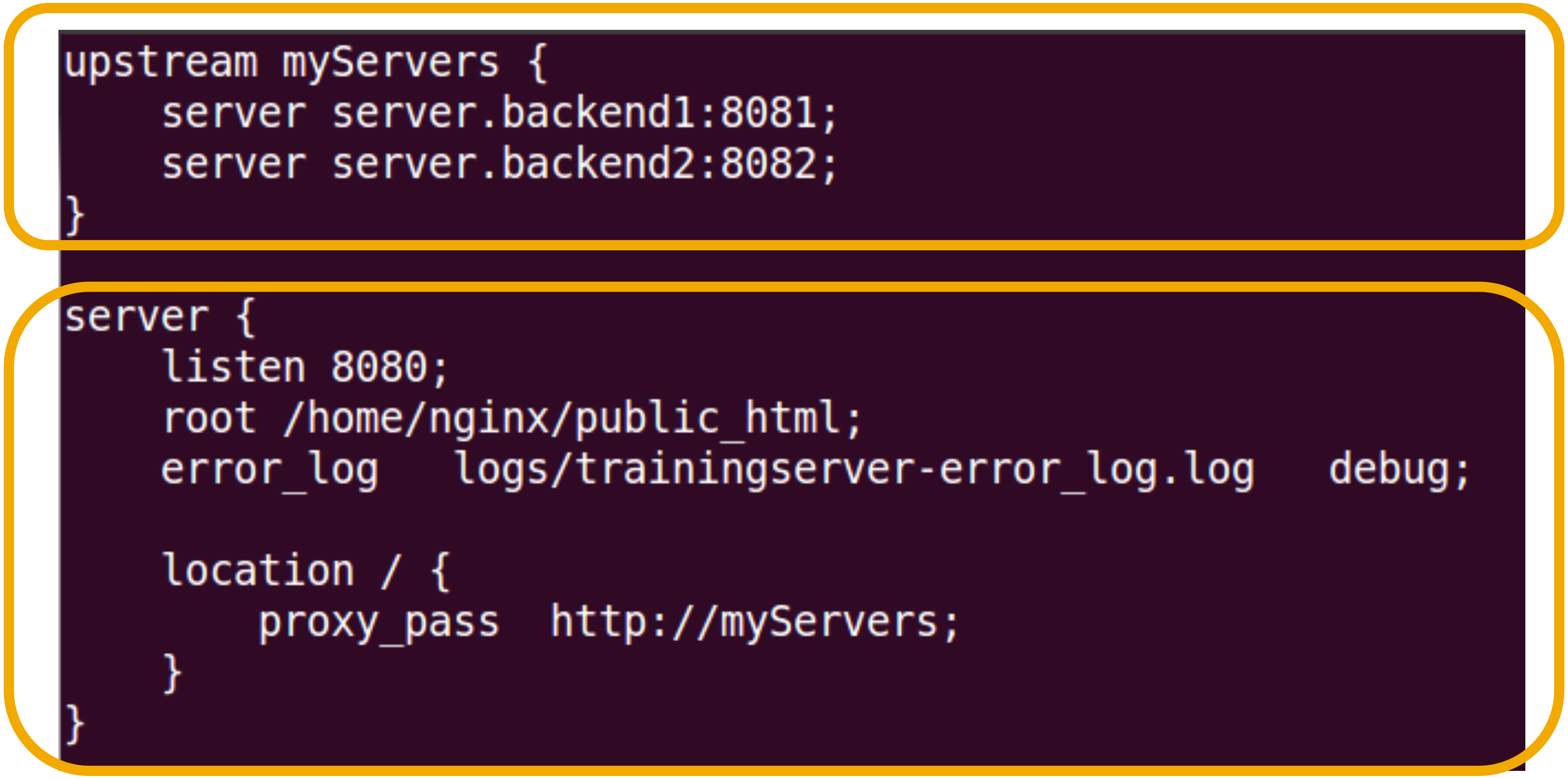
```
error_log   logs/error.log info;
```

Logging best practices

- Should keep a separate error log file for each server
- Helps to reduce size of each log file and makes troubleshooting easier

```
server {
     server_name server1.com;
     root /data/server1.com;
     error_log logs/server1.error.log     info;
}

server {
     server_name server2.com
     root /data/server2.com;
     error_log logs/server2.error.log     info;
}
```

Proxying to the upstream block



```
upstream myServers {
    server server.backend1:8081;
    server server.backend2:8082;
}

server {
    listen 8080;
    root /home/nginx/public_html;
    error_log   logs/trainingserver-error_log.log   debug;

    location / {
        proxy_pass  http://myServers;
    }
}
```

Specifying server priorities

- By default, all servers defined in the upstream block are treated with equal priority
- Use the **weight** parameter to indicate a higher or lower weighting for a particular server

```
upstream myServers {
   server backend.server1 weight=5
   server backend.server2 weight=3
   server backend.server3 weight=2
}
```

Reverse proxy and caching

- It's common to use NGINX in front of another web or application server
- NGINX can handle serving all the static content, while requests for dynamic content such as php are proxied to the application server
- Static content can then be cached to improve performance

Defining the cache path

```
http {
      proxy_cache_path /var/cache/nginx levels=1:2
keys_zone=server-cache:8m max_size=1000m
inactive=600m;
      proxy_temp_path /tmp/nginx;
```

- **proxy_cache_path** directive to set where to store cached content
- **proxy_temp_path** directive tells NGINX where to store temporary data which is used to build the cache
- Both directives must be placed in HTTP context

Defining the cache path

- **proxy_cache_path** parameters
  - **keys_zone** parameter specifies the name and size of the cache
  - **max_size** parameter specifies the maximum size of the cache
  - **Inactive** parameter specifies how long cached data is kept for if not accessed

Configuring the proxy cache

- **proxy_cache_key** directive specifies to use the hostname/subdomain/domain and request URI as the key
- **proxy_cache** directive defines the shared memory zone used for caching.
  - Name specified must match the name of the cache defined in the proxy_cache_path directive

```
Location / {
    proxy_pass http://application.com:8080;
    proxy_cache_key "$scheme$host$request_uri";
    proxy_cache server-cache;
    proxy_chache_valid 1m;
    proxy_cache_valid 404 1m;
]
```

Passing headers

- Use **proxy_set_header** directive to redefine the request header fields that are passed to the proxied server
- Use this to pass on the hostname and IP address of the request machine
- Without setting the headers, the server you proxy to will simply see your reverse proxy server's host and IP

```
proxy_set_header   Host    $host;
proxy_set_header   X-Real-IP   $remote_addr;
proxy_set_header   X-Forwarded-For    $proxy_add_x_forwarded_for;
```

Configuring a HTTPS server

- Enable SSL by specifying the SSL parameter on the listen directive
- Specify the path of your SSL server certificate and private key

```
server {
    listen   443 ssl;
    server_name  training.secure.com;

    error_log    logs/secure.error.log;
    ssl_certificate  /etc/nginx/certs/nginxtraining.crt
    ssl_certificate_key /etc/nginx/certs/nginxtraining.key
]
```

SSL session cache

- SSL sessions can be stored in a cache and reused in order to avoid having to perform a “handshake” as part of subsequent connections
- Reduces the amount of CPU intensive operations on the server
- The session cache can be shared between workers
- Cache will timeout after 5 minutes by default, but this can be configured with the **ssl_session_timeout** directive

Session cache example

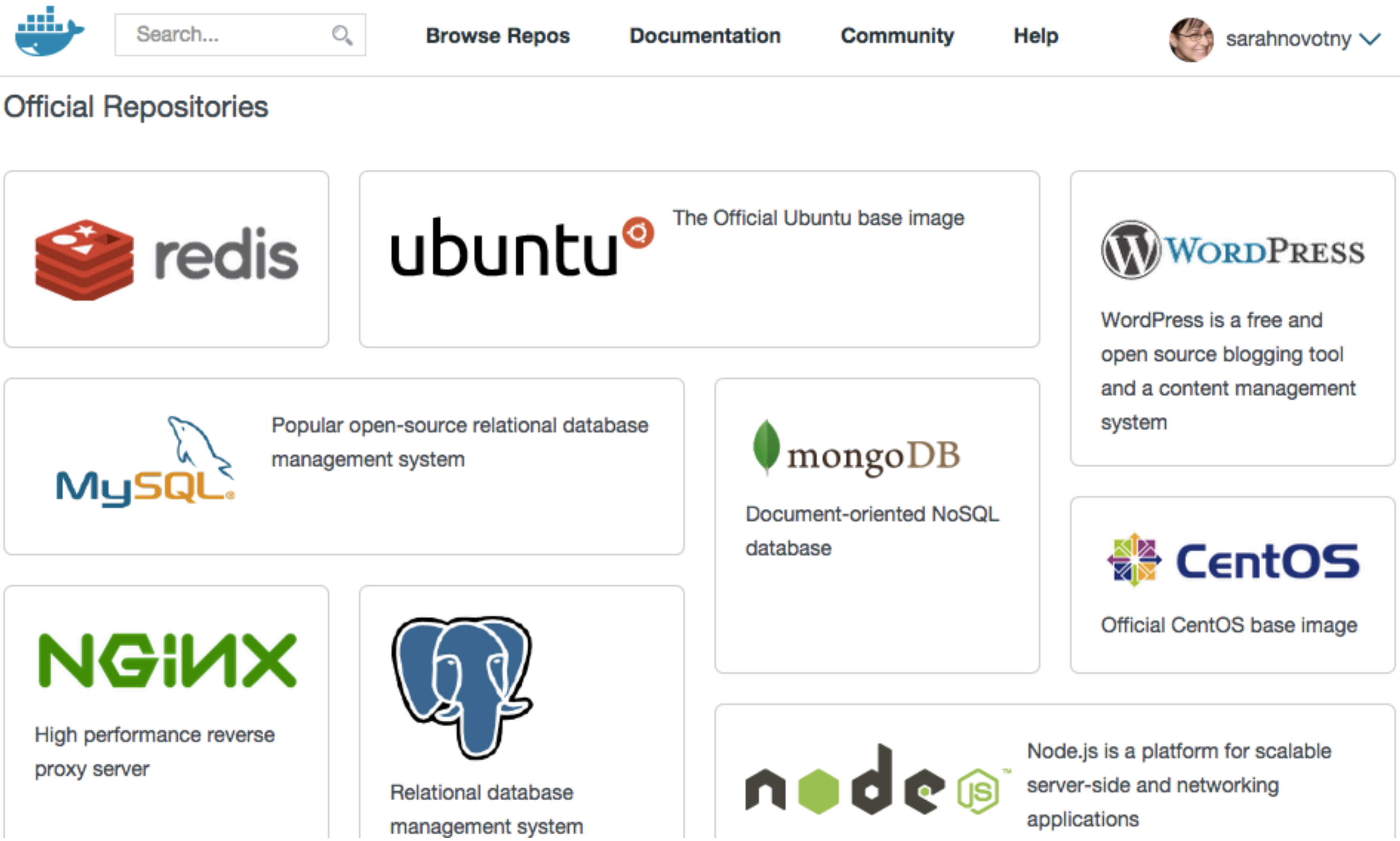
- Syntax
  `ssl_session_cache shared:<name>:size;`
- Size is specified in bytes or megabytes
- 1 MB can store around 4000 sessions
- Can specified in the http or server context

Example

```
http {
    ssl_session_cache shared:ssl:10m;
    ssl_session_timeout 10m;

    server {
        listen 443 ssl;
        ...
```

Now with
more Docker

registry.hub.docker.com



Official Repositories

redis

ubuntu — The Official Ubuntu base image

WordPress — WordPress is a free and open source blogging tool and a content management system

MySQL — Popular open-source relational database management system

mongoDB — Document-oriented NoSQL database

CentOS — Official CentOS base image

NGINX — High performance reverse proxy server

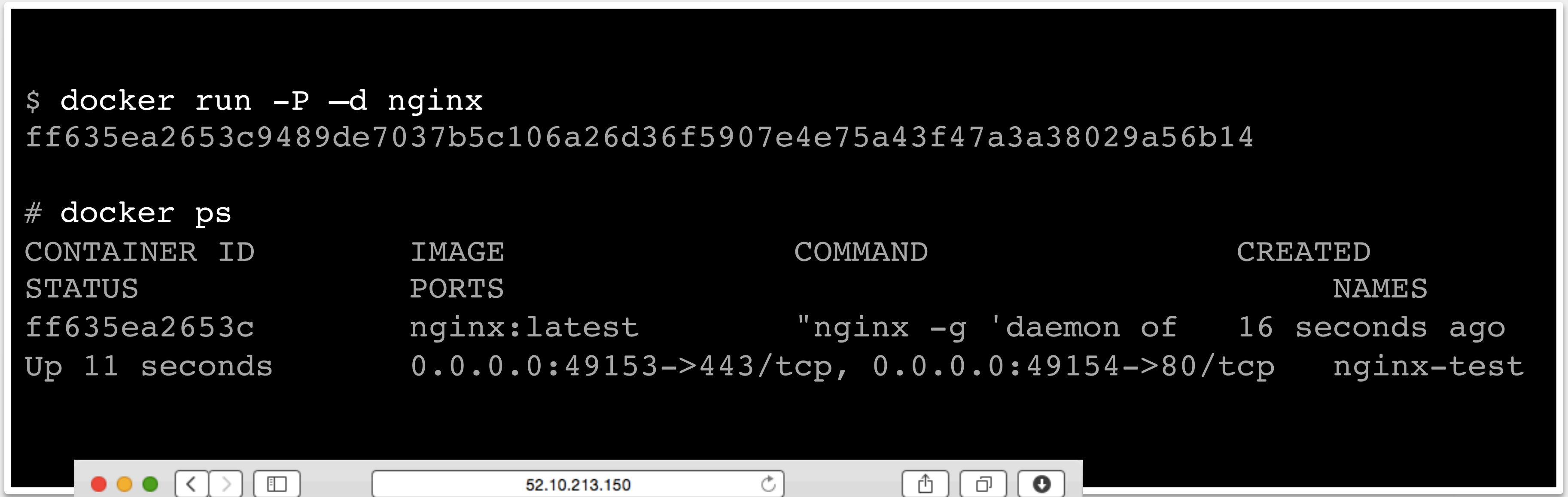
Relational database management system

node js — Node.js is a platform for scalable server-side and networking applications

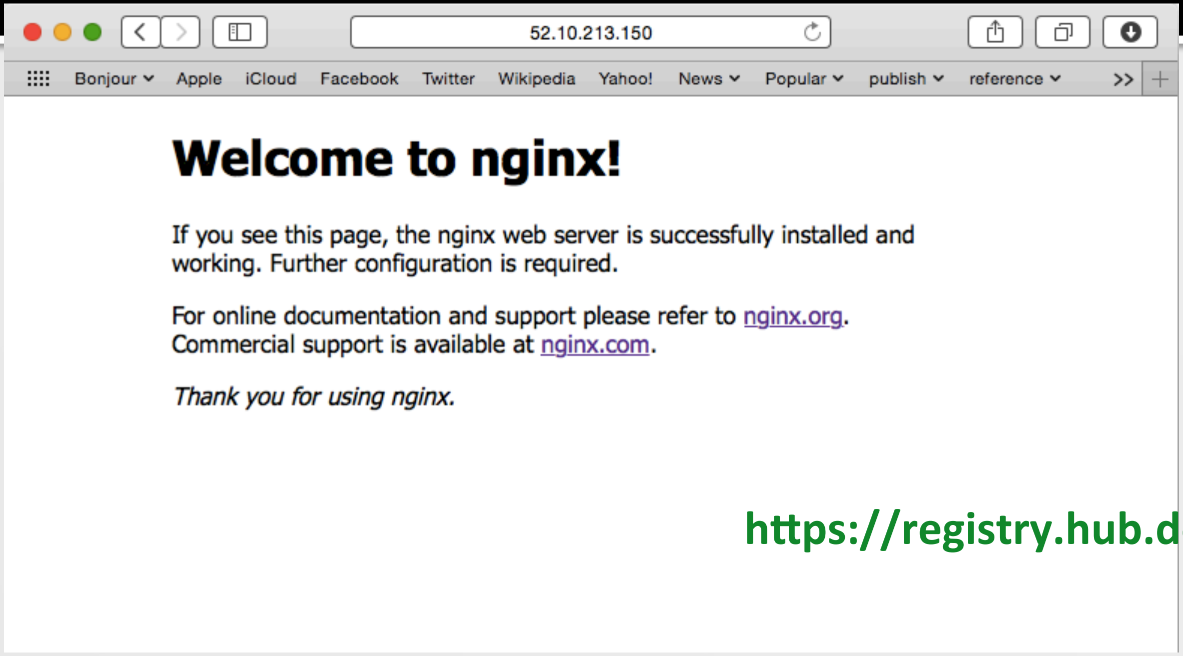
Dockerfile

```
FROM debian:wheezy

MAINTAINER NGINX Docker Maintainers "docker-maint@nginx.com"

RUN apt-key adv --keyserver pgp.mit.edu --recv-keys
573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62
RUN echo "deb http://nginx.org/packages/mainline/debian/ wheezy nginx" >> /etc/
apt/sources.list

ENV NGINX_VERSION 1.7.10-1~wheezy

RUN apt-get update && \
    apt-get install -y ca-certificates nginx=${NGINX_VERSION} && \
    rm -rf /var/lib/apt/lists/*

# forward request and error logs to docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log
RUN ln -sf /dev/stderr /var/log/nginx/error.log

VOLUME ["/var/cache/nginx"]

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]
```

Run our Docker container



```
$ docker run -P –d nginx
ff635ea2653c9489de7037b5c106a26d36f5907e4e75a43f47a3a38029a56b14

# docker ps
CONTAINER ID        IMAGE               COMMAND                 CREATED
STATUS              PORTS                                            NAMES
ff635ea2653c        nginx:latest        "nginx -g 'daemon of    16 seconds ago
Up 11 seconds       0.0.0.0:49153->443/tcp, 0.0.0.0:49154->80/tcp    nginx-test
```



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.

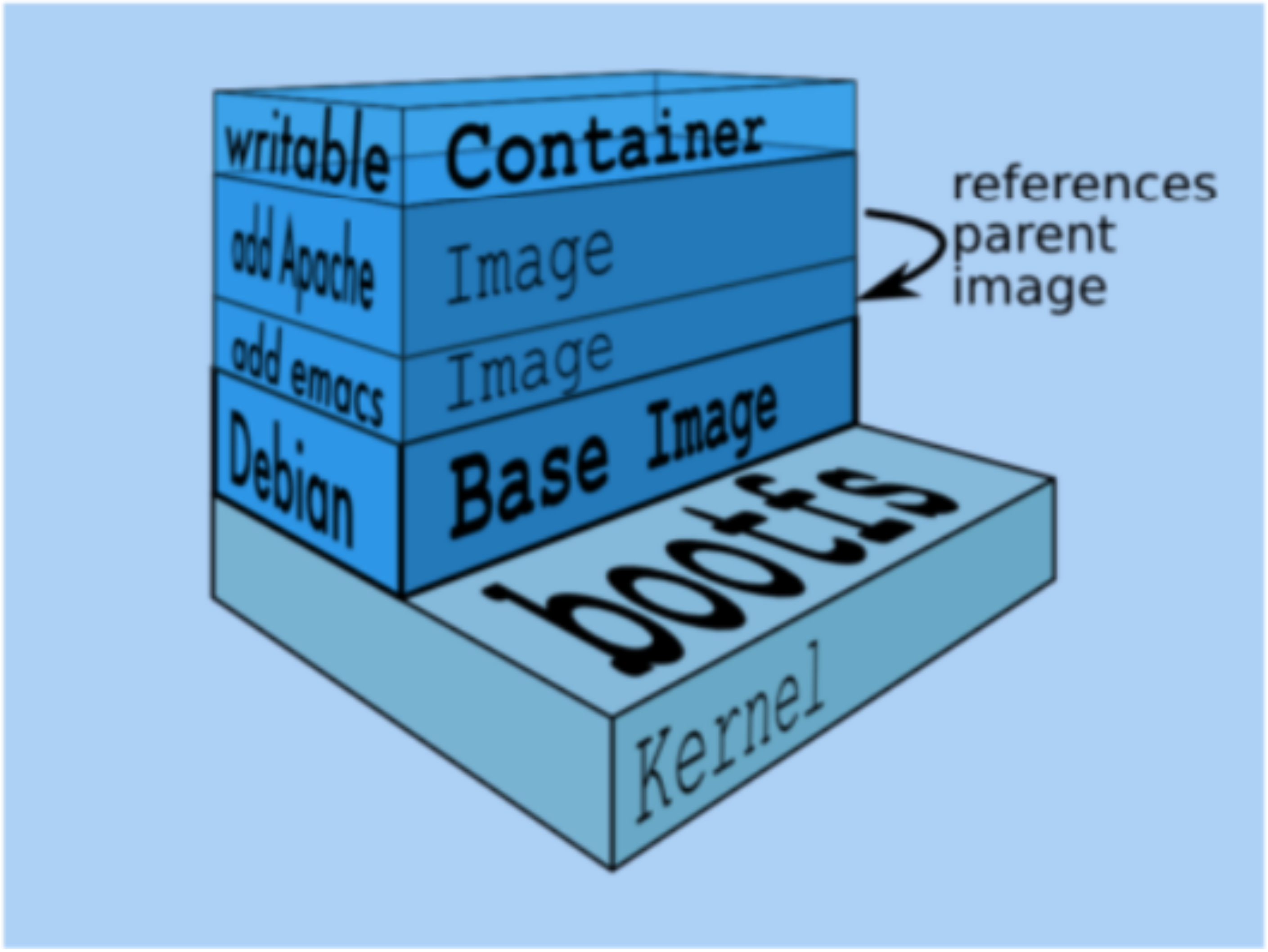
Thank you for using nginx.

https://registry.hub.docker.com/_/nginx/

Exploring our Docker container

```
$ docker@52.10.213.150 ~: docker run  -it nginx /bin/bash
root@74d2a7e93244:/# more /etc/nginx/nginx.conf

user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"’;


…
```

Extending base images in your Dockerfile



From @jpettazo's Docker talk 20150220 #SCaLE13x

Your NGINX Dockerfile

```
FROM nginx
RUN rm /etc/nginx/conf.d/default.conf
RUN rm /etc/nginx/conf.d/example_ssl.conf
COPY static-html-directory /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf
```

- Fancier options i.e**. more repeatable and scalable**
  - Defining VOLUMEs
  - Using helper containers
  - Linking containers

http://nginx.com/blog/deploying-nginx-nginx-plus-docker/

@sarahnovotny
Chief Evangelist, NGINX
Program Chair, OSCON

Thanks for your time!

http://sarah.is/ExcitedAboutMicroservices

HOW TO ADOPT
MICROSERVICES

NGINX

Download O'Reilly's new book on
designing fine-grained systems

FREE DOWNLOAD

NGINX

DISCOVER THE SECRET

to making your site and apps perform better.

secretheart.nginx.com

NGINX PLUS PROVIDES

a complete application delivery platform.

NGINX PLUS PROVIDES

a complete application delivery platform.

LOAD BALANCING Optimize the availability of apps, APIs, and services

NGINX PLUS PROVIDES
a complete application delivery platform.

LOAD BALANCING	Optimize the availability of apps, APIs, and services
WEB SERVER	Deliver assets with unparalleled speed and efficiency

NGINX PLUS PROVIDES
a complete application delivery platform.

LOAD BALANCING	Optimize the availability of apps, APIs, and services
WEB SERVER	Deliver assets with unparalleled speed and efficiency
CONTENT CACHING	Accelerate local origin servers and create edge servers

NGINX PLUS PROVIDES
a complete application delivery platform.

LOAD BALANCING	Optimize the availability of apps, APIs, and services
WEB SERVER	Deliver assets with unparalleled speed and efficiency
CONTENT CACHING	Accelerate local origin servers and create edge servers
STREAMING MEDIA	Stream high-quality video on demand to any device