Acerca de NGINX

- Proyecto iniciado en el 2004
- Compañia fundada en el 2011
- NGINX Plus primer release en 2013
- 1000 commercial customers
- 160 employees

NGINX

- Nginx es un servidor web/proxy incerso ligero de alto rendimiento y un proxy de protocolos de correo electrónico (IMAP/POP3)
- Es software libre y de codigo abierto, licenciado bajo la licencia BSD simplificada, tambien existe una version comercial distribuida bajo el nombre Nginx Plus
- Mayor descarga de software en Docker Hub
- Más utilizada en contenedores según DataDog y Sysdig

250 million

total sites
running on NGINX

60%

of the Top 10,000
most visited websites

40%

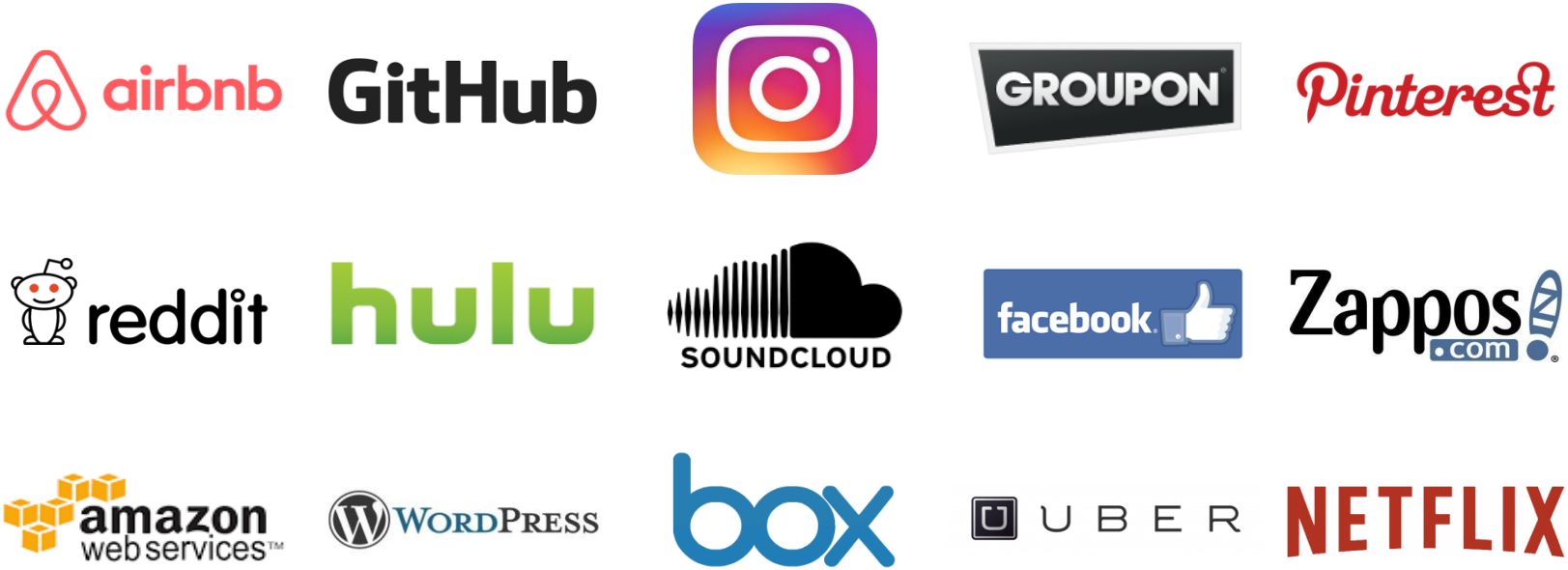
Of applications on
Amazon Web Services

Source: Netcraft July 2016 Web Server Survey

NGINX

- Originalmente, Nginx fue desarrollado para satisfacer las necesidades de varios sitios web, que recibian unos 500 millones de peticiones al día.
- Empresas que utilizan Nginx:
  - ✓ DropBox
  - ✓ CloudFare
  - ✓ Instagram
  - ✓ Netflix
  - ✓ GitHub

The First Web Apps



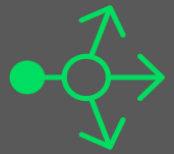
MORE INFORMATION AT
NGINX.COM

Now, Every App Is A Web App

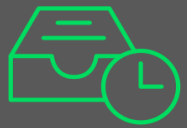
MORE INFORMATION AT
NGINX.COM

High Performance Webserver

NGINX+



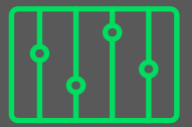
Load Balancer



Content Cache

Web Server

Security Controls

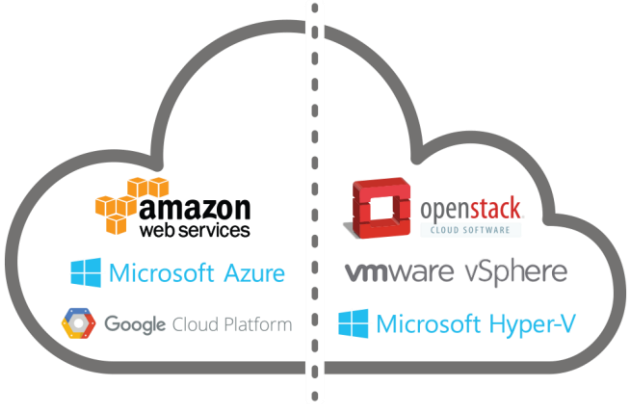


Monitoring & Management

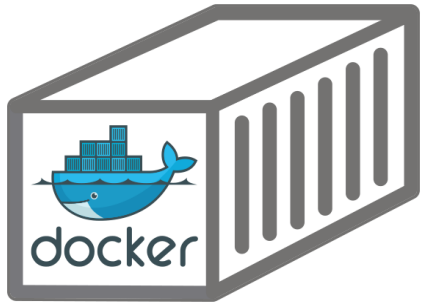
Impecable entrega de aplicaciones para la Web moderna

Funcionan sobre cualquier entorno

Bare Metal

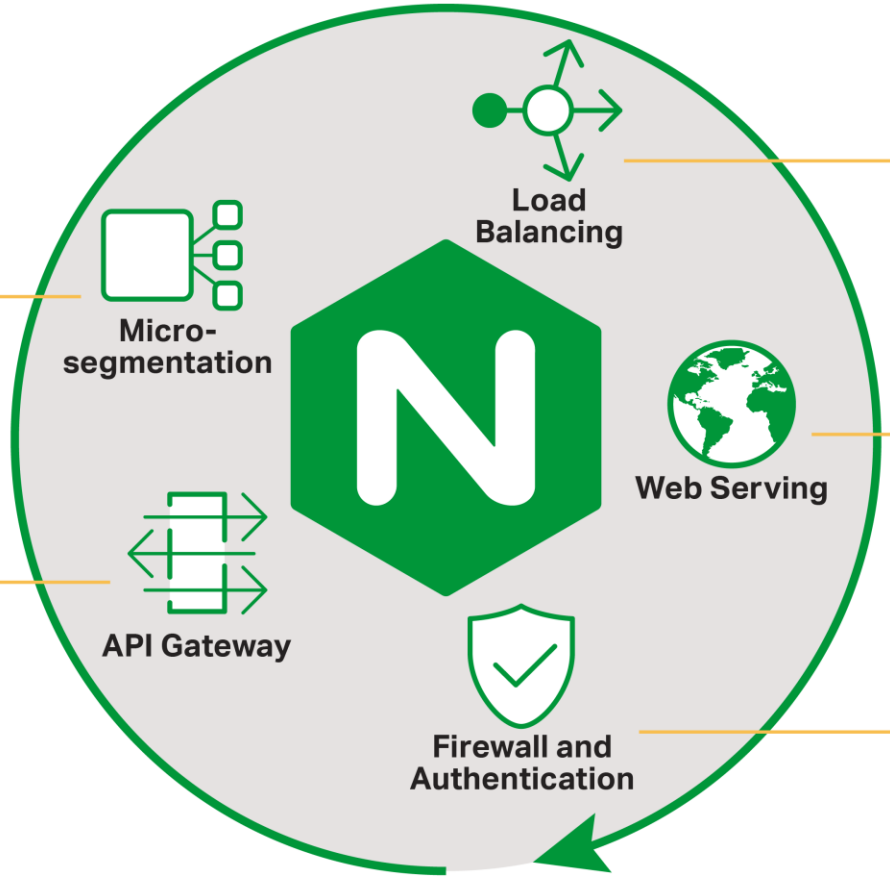


Public/Private/Hybrid Cloud



Containers

Enterprise Application Delivery Capabilities



Mejor rendimiento y mayor tiempo de actividad

Tecnología Alternativa:
- HAProxy (open source)
- F5 /Citrix (commercial)

Política de Control detallado

Tecnologías Alternativas technologies:
- HAProxy (open source)
- Manual de Enfoque DevOps

Entrega de Contenido y gateway a Aplicaciones

Tecnología Alternativa:
- Apache, IIS

Control de Acceso y Entrega de APIs

Tecnologías Alternativas:
- NGINX open source
- API Gateways built on NGINX (Mashape)
- API management solutions (3Scale)

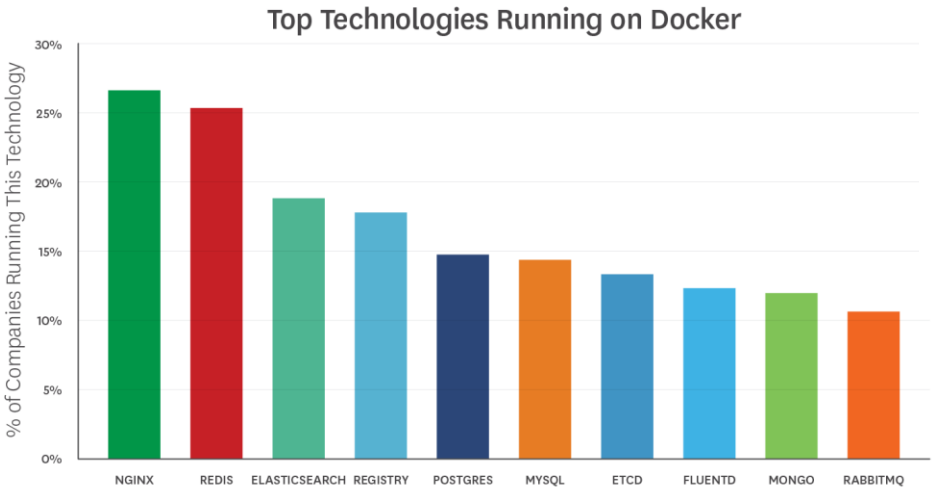
Proteja y asegura sus aplicaciones

Tecnologia Alternativa:
- Imperva, ADC vendors (App firewall)
- CA SSO, PingIdentity (Authentication)
- Check Point, Palo Alto (Firewall)

NGINX Plus

MORE INFORMATION AT NGINX.COM

#1 Tool for Containers



NGINX fluentd APACHE PostgreSQL elastic

CONSUL redis mongoDB HAPROXY

etcd JVM php fpm

"Above is a list of the twelve most common application checks we see running. Of interest to us was that Nginx was the most popular."

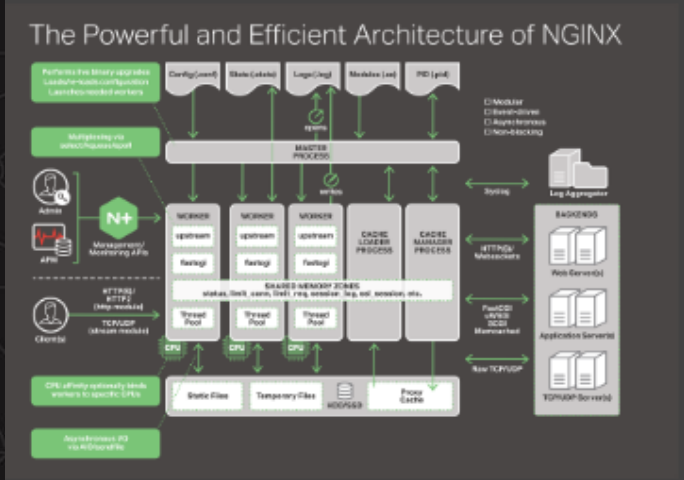
Sysdig

MORE INFORMATION AT NGINX.COM

NGINX

- Nginx, tiene un proceso Master y Workers.
- El objetivo del proceso Master es leer y evaluar el archivo de configuración y mantener el proceso worker.
- El proceso Worker, hace el procesamiento real de las solicitudes, en función del Sistema operativo para distribuir de manera eficiente.
- La directive worker_processes define la cantidad de procesos de trabajo y esto es en el archivo de configuración nginx.conf

NGINX Architecture



- Master Process:
  - PID
  - Userspace
- Worker Processes
  - Request
  - Response
- Shared Memory:
  - Limit Rates
  - Sessions
  - Status

NGINX – Instalación Lab1

- Descarga e instalación de requisitos
- Descarga del certificado de Nginx
- Instalación de Nginx
- Configuración de Nginx
- Validar la ejecución del servicio Nginx

Comandos Básicos NGINX

```
#reloads config
$ nginx -s reload

#graceful shutdown
$ nginx -s quit

#terminates NGINX process
$ nginx -s stop

#config syntax check (pre-reload)
$ nginx -t

#displays currently running configs
$ nginx -T

#checks NGINX version
$ nginx -v
```

Comandos Básicos NGINX

Activity/OS	CentOS / RedHat 6	CentOS / RedHat 7	Ubuntu 14.04 / Debian 8
Start NGINX	`service nginx start`	`systemctl start nginx`	`service nginx start`
Stop NGINX	`service nginx stop`	`systemctl stop nginx`	`service nginx stop`
Restart NGINX	`service nginx restart`	`systemctl restart nginx`	`service nginx restart`
Reload NGINX	`service nginx reload`	N/A	N/A

Sirviendo Contenido Estático

Running Processes

To check running processes, run the following command:

```
$ ps aux | grep nginx
```

```
[root@serverins conf.d]# ps -ef|grep nginx
root       3167      1  0 11:34 ?        00:00:00 nginx: master process /usr/sbin/nginx -c /etc/nginx/nginx.conf
nginx      3168   3167  0 11:34 ?        00:00:00 nginx: worker process
```

Sirviendo Contenido Estático

Path to Files

Executable Path

```
$ /usr/sbin/nginx
```

Log Path

```
$ /var/log/nginx
```

Sirviendo Contenido Estático

Configuration File

Global Configuration Path

```
$ /etc/nginx/nginx.conf
```

Additional Configuration(s) Path

```
$ /etc/nginx/conf.d/*.conf
```

Documentation: nginx.conf Example

Sirviendo Contenido Estático

Include Directive

The following line in nginx.conf allows NGINX to search for additional configurations

```
include /etc/nginx/conf.d/*.conf;
```

Sirviendo Contenido Estático

- El archive de configuración consta de “Directivas”, “Bloques” y “Contextos”
- Las directivas simples de una sola linea terminan con un punto y coma.
- Los bloques se agrupan con varias directivas encerrandolas con llaves.

```
user              nobody;
error_log         logs/error.log notice;
worker_processes  1;
```

- Algunas directivas de alto nivel las cuales se agrupan a diferentes tipos de trafico se denominan “Contextos”

`events` – General connection processing
`http` – HTTP traffic
`mail` – Mail traffic
`stream` – TCP and UDP traffic

Configuration File Structure

- Directives
- Blocks
- Contexts

Sirviendo Contenido Estático

Directives

Configuration statement that controls NGINX Modules

```
listen 80;
root /usr/share/nginx/html;
index index.html index.htm index.php;
```

Sirviendo Contenido Estático

Blocks

Contains mixture of directives and data—begins and ends with curly brackets.

```
server {
      listen 80;
      root /usr/share/nginx/html;
      index index.html index.htm index.php;
  }
```

Sirviendo Contenido Estático

Contexts

Nested Blocks implying a hierarchy. Colloquially, 'Block' and 'Context' are interchangeable.

```
http {
        include       /etc/nginx/mime.types;
        default_type  application/octet-stream;
        gzip on;
        log_format

        server {
                listen 80;
                root /usr/share/nginx/html;
                index index.html index.htm index.php;

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

Comandos Básicos NGINX

Serving Content

Requirements:

- `http` - high level processing (logging, compression, caching etc.)
- `server` - virtual server that handles the request
- `location` - processing based on request URI

Comandos Básicos NGINX

`server` Block Example

- Defines virtual `server` ("VirtualHost" in Apache)
- Always nested inside either `http` or `stream` context
- Binds to TCP sockets with `server_name` and `listen`

```
server {
    listen 80;
    server_name localhost;
    root /home/ubuntu/public_html;
}
```

Comandos Básicos NGINX

`listen` Directive

- Defines IP / Port that `server` responds to
- Default is `0.0.0.0:80` (`:8080` for non-root)
- Can be: IP, IP:Port, Port, Unix Socket

Comandos Básicos NGINX

`listen` Example

If example.com is hosted on port 80 of 192.168.1.10, the first block serves the response

```
server {
    listen 192.168.1.10;

}

server {
    listen 80;
    server_name example.com;

}
```

Lab 2. Configuración Nginx

- Requiere privilegios “sudo” o “root”

1. Navegar a la configuración de NGINX

   $ cd /etc/nginx/conf.d

2. Hacer un backup al archivo de configuración “default” y “ssl” de NGINX

   $ sudo mv default.conf default.conf.bak

   $ sudo mv example_ssl.conf example_ssl.conf.bak

3. Crear un Nuevo archive de configuración llamado: server_example.conf:

   $ sudo vim server_example.conf

Lab 2. Configuración Nginx

4. Crear un bloque “server” que escuche en el <Private_IP>:80

5. Crear un bloque adicional “server” que escuche por el Puerto 80, referenciado por el hostname sobre la directiva “server_name”

6. Adicionar lo siguiente en cada bloque “server”

return 200 “this is server 1”;

return 200 “this is server 2”;

7. Validar con el commando “curl” para la IP privada, hostname y localhost, anota los resultados.

Nota:

$ sudo nginx -t

$ sudo nginx -s reload

$ curl IP

Lab 2. Configuración NGINX

```
server {
        listen 192.168.1.100;
        return 200 "this is server 1";
}
server {
        server_name localhost;
        return 200 "this is server 2";
}
```

Directiva Server_Name

`server_name` Example 1

If "Host" value matches "host1.example.com" exactly, second block serves response

```
server {
    listen 80;
    server_name *.example.com;

}

server {
    listen 80;
    server_name host1.example.com;

}
```

Bloque Location

```
location /application1 {

}
```

Two most common types:

- **Prefix**
- **Regex**

Bloque Location

- Checked first, then longest match serves response
- Nested inside `server` context

```
location /application1 {

}

location /application1/images {
        alias /media/data;
}
```

Second prefix serves response if request is:

```
$ curl http://somedomain.com/application1/images/?img2
```

Bloque Location

Location Modifiers

Modifier	Usage
`=`	Literal String Match
`~*`	Case Insensitive Regex
`~`	Case Sensitive Regex
`^~`	Prevent Regex Location Processing
`@`	Named Location Routing (redirects, error pages etc.)

Bloque Location

Regex Location

Matched sequentially and only after prefix locations.

```
location /application1 {

}

location ~* ^\.(gif|jpg|jpeg|png)$ {
        alias /media/data;
}
```

Bloque Location

Location Order

Configuration Example

```
server {
    listen 80 default_server;
    root /usr/share/nginx/html;

    location = / {
    }

    location ~* ^\.(png|jpg)$ {
    }

    location ^~ /app1 {
    }
}
```

Given Request:

```
http://example.com/app1/logo.png
```

Process Order:

- Location 1
- Location 3
- Location 2

Lab 3. Server Pages

1. Hacer un backup a “server_example.conf”
2. Crear un archivo nuevo “server1.conf”, con las siguientes lineas:

```
server {
        listen 80;
        root /var/public_html;
}
```

3. Adicionar el bloque “location”:

```
location /application1 {
}
location /application2 {
}
location /images {
        root /data;
}
```

Lab 3. Server Pages

4. Usar “curl”, o un browser para validar las URIs:

   /application1/
   /application2/
   /images/logo.png

5. Si devuelve un error 403, modificar con la siguiente información:

```
location /application1 {
        index app1.html;
}
location /application2 {
        index app2.html;
}
location /images {
        root /data;
}
```

Lab 3. Server Pages

4. La configuración debe queda de la siguiente forma:

```
server {
    listen 80;
    root /var/public_html;

    location /application1 {
        index app1.html;
    }

    location /application2 {
        index app2.html;
    }

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

Lab 4. Habilitar el monitoreo Nginx

1. En el archive de configuración server1.conf, adicionar las siguientes lineas:

```
server {
    listen 8188;
    status_zone status-page;
    allow 192.168.1.0/24;
    deny all;
    root /usr/share/nginx/html;
    location = /status.html { }
    location = / {
        return 301 /status.html;
   }
   location /status {
       status;
       status_format json;
   }

}
```

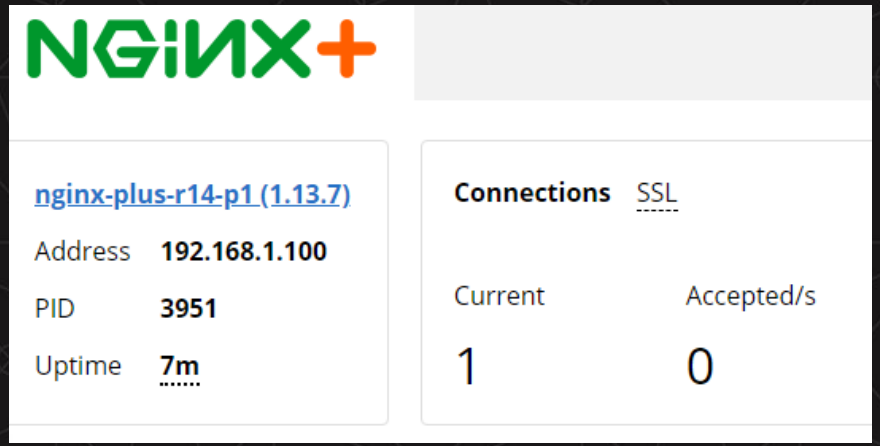
Lab 4. Habilitar el monitoreo Nginx

2. Validar la configuración realizada:

   $ nginx -t

3. Ingresar a la URL:

   http://192.168.1.100:8188/status.html



Questions and Next Steps

NGINX