NGINX Web Server

POuL

have a lot of fun...

Tommaso Sardelli
sardelli.tommaso[at]gmail.com

Corsi GNU/Linux Avanzati 2014

10 Aprile 2014

Di cosa parliamo oggi?

(Compra una vocale)

Indice Generale

Outline

Cos'è un Web Server

- È uno degli elementi fondamentali della rete internet ed entra in gioco, senza che noi ce ne accorgiamo, ogni volta che navighiamo in un sito web.

Cos'è un Web Server

- È uno degli elementi fondamentali della rete internet ed entra in gioco, senza che noi ce ne accorgiamo, ogni volta che navighiamo in un sito web.
- Wordpress, Drupal, Joomla necessitano tutti di avere un software alle spalle (il web server appunto) per poter funzionare.

Cos'è un Web Server

- È uno degli elementi fondamentali della rete internet ed entra in gioco, senza che noi ce ne accorgiamo, ogni volta che navighiamo in un sito web.
- Wordpress, Drupal, Joomla necessitano tutti di avere un software alle spalle (il web server appunto) per poter funzionare.
- In parole (molto) povere, è il software che **invia pagine web** ai browser, nel momento in cui le **richiedono**.

Web Server più famosi

I web server più utilizzati al giorno d'oggi sono:

- Apache
- nginx
- lighttpd
- IIS (Internet Information Services)
- GWS (Google Web Server)
- Molti altri...

Outline

In principio era Apache

- Nato nel lontano **1995** dalle ceneri del demone HTTP.

In principio era Apache

- Nato nel lontano **1995** dalle ceneri del demone HTTP.
- Ha visto negli anni un enorme diffusione che lo portò, nel 2006, ad essere utilizzato per il **69,32%** dei siti web (secondo un' indagine Netcraft).

In principio era Apache

- Nato nel lontano **1995** dalle ceneri del demone HTTP.
- Ha visto negli anni un enorme diffusione che lo portò, nel 2006, ad essere utilizzato per il **69,32%** dei siti web (secondo un' indagine Netcraft).
- Ancora oggi è il web server più utilizzato e quello che offre maggiore compatibilità.

In principio era Apache

- Nato nel lontano **1995** dalle ceneri del demone HTTP.
- Ha visto negli anni un enorme diffusione che lo portò, nel 2006, ad essere utilizzato per il **69,32%** dei siti web (secondo un' indagine Netcraft).
- Ancora oggi è il web server più utilizzato e quello che offre maggiore compatibilità.
- Costituisce uno degli elementi principali dello stack **LAMP** (Linux, Apache, MySQL, PHP).

Poi Igor Sysoev creò NGINX

- Nato nel 2002 e progettato per servire le richieste dirette al sito www.rambler.ru (**500 milioni di visite al giorno** nel 2008) che avevano messo in crisi Apache.

Poi Igor Sysoev creò NGINX

- Nato nel 2002 e progettato per servire le richieste dirette al sito www.rambler.ru (**500 milioni di visite al giorno** nel 2008) che avevano messo in crisi Apache.
- È usato da siti come **Facebook**, **Dropbox**, **Wordpress**, **Netflix** e tanti altri.

Poi Igor Sysoev creò NGINX

- Nato nel 2002 e progettato per servire le richieste dirette al sito www.rambler.ru (**500 milioni di visite al giorno** nel 2008) che avevano messo in crisi Apache.
- È usato da siti come **Facebook**, **Dropbox**, **Wordpress**, **Netflix** e tanti altri.
- Scritto in **C**

Poi Igor Sysoev creò NGINX

- Nato nel 2002 e progettato per servire le richieste dirette al sito www.rambler.ru (**500 milioni di visite al giorno** nel 2008) che avevano messo in crisi Apache.
- È usato da siti come **Facebook**, **Dropbox**, **Wordpress**, **Netflix** e tanti altri.
- Scritto in **C**
- Obiettivi iniziali:
  - Garantire ottime performance anche sotto carichi elevati
  - Sfruttare al meglio le risorse hardware
  - Garantire un buon livello di sicurezza
  - Semplicità di configurazione

E vide che era cosa buona

- Programma **leggerissimo** sia nel consumo di memoria che nell'utilizzo della CPU.

E vide che era cosa buona

- Programma **leggerissimo** sia nel consumo di memoria che nell'utilizzo della CPU.
- **Performance elevate** grazie a una mirata gestione dei processi/thread.

E vide che era cosa buona

- Programma **leggerissimo** sia nel consumo di memoria che nell'utilizzo della CPU.
- **Performance elevate** grazie a una mirata gestione dei processi/thread.
- Perfetto per il **load balancing**

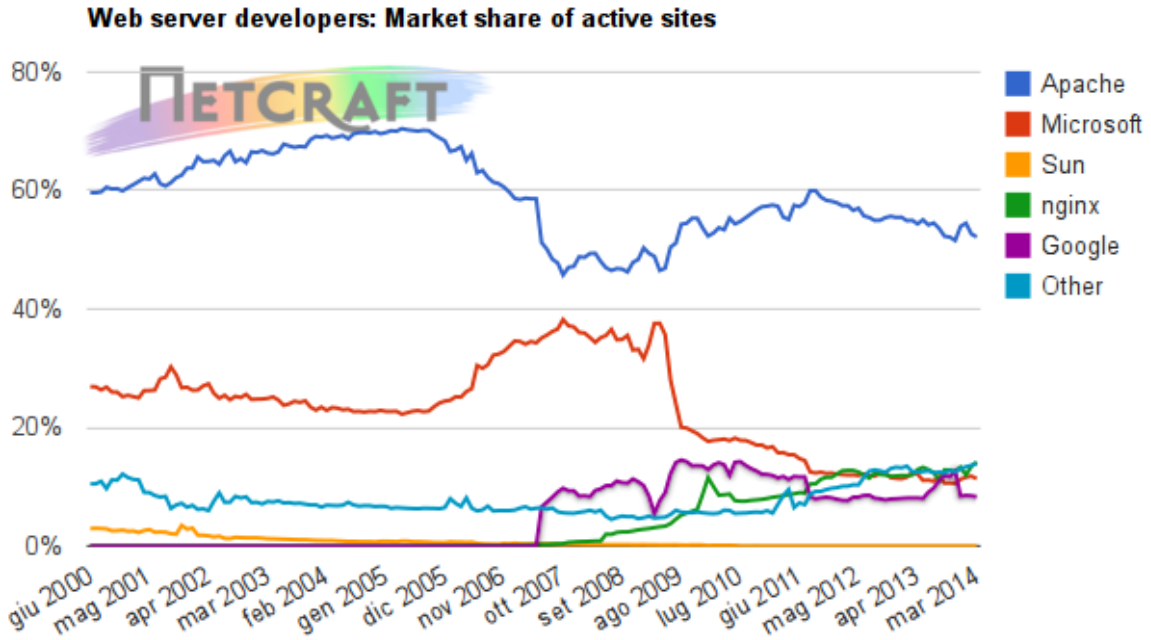
E vide che era cosa buona

- Programma **leggerissimo** sia nel consumo di memoria che nell'utilizzo della CPU.
- **Performance elevate** grazie a una mirata gestione dei processi/thread.
- Perfetto per il **load balancing**
- **Altamente configurabile** anche a basso livello (numero di file aperti per ogni processo, utilizzo dei core della CPU, ...)

E vide che era cosa buona

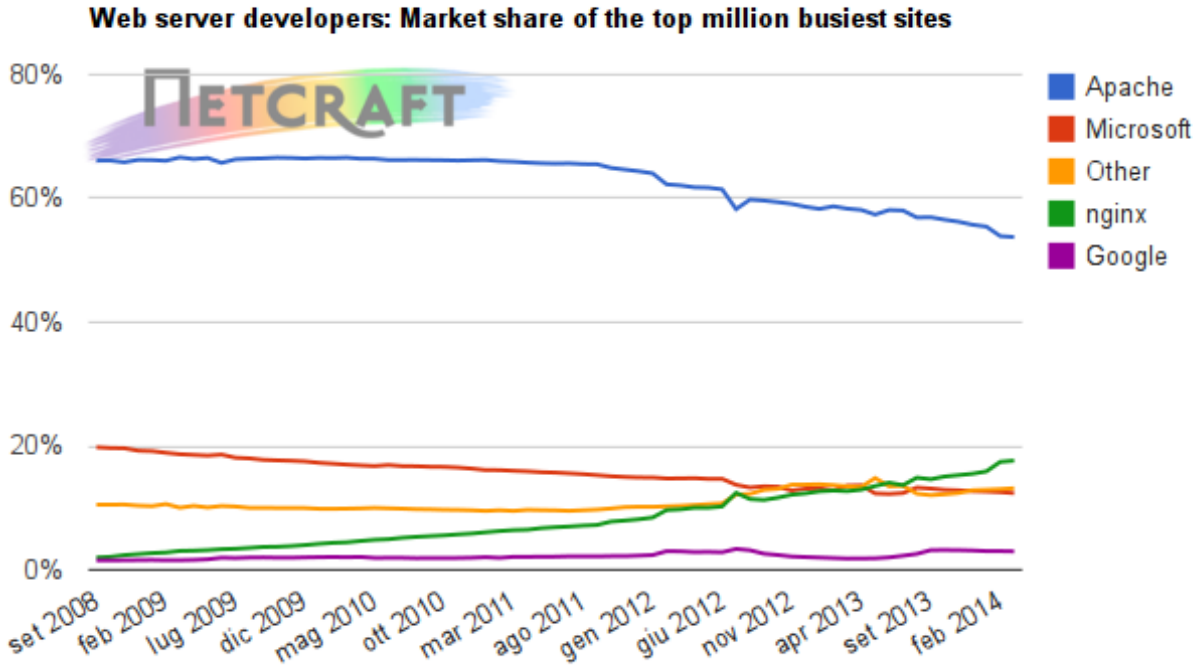
- Programma **leggerissimo** sia nel consumo di memoria che nell'utilizzo della CPU.
- **Performance elevate** grazie a una mirata gestione dei processi/thread.
- Perfetto per il **load balancing**
- **Altamente configurabile** anche a basso livello (numero di file aperti per ogni processo, utilizzo dei core della CPU, ...)
- **Modulare** grazie ai moduli integrati più molti altri moduli aggiuntivi che possono essere abilitati in fase di compilazione

E fu sera e fu mattina



Developer	February 2014	Percent	March 2014	Percent	Change
Apache	94,741,928	52.68%	93,759,928	52.18%	-0.50
nginx	24,206,737	13.46%	25,497,586	14.19%	0.73
Microsoft	21,196,966	11.79%	20,436,280	11.37%	-0.41
Google	15,245,912	8.48%	14,967,579	8.33%	-0.15

E fu sera e fu mattina



Developer	February 2014	Percent	March 2014	Percent	Change
Apache	539,129	53.91%	537,714	53.77%	-0.14
nginx	174,552	17.46%	176,507	17.65%	0.20
Microsoft	125,595	12.56%	123,981	12.40%	-0.16
Google	30,314	3.03%	29,937	2.99%	-0.04

Outline

Architettura di NGINX

NGINX è suddiviso in due processi:

- **Master process**: viene lanciato da root ed è il processo principale. Ha il compito di leggere i file di configurazione e di aprire il socket che gli permetterà di comunicare con il secondo componente di NGINX

Architettura di NGINX

NGINX è suddiviso in due processi:

- **Master process**: viene lanciato da root ed è il processo principale. Ha il compito di leggere i file di configurazione e di aprire il socket che gli permetterà di comunicare con il secondo componente di NGINX
- I **Worker processes**: una serie di processi lanciati come utente non privilegiato (ad esempio www-data) che hanno il compito di servire le richieste HTTP

Architettura di NGINX

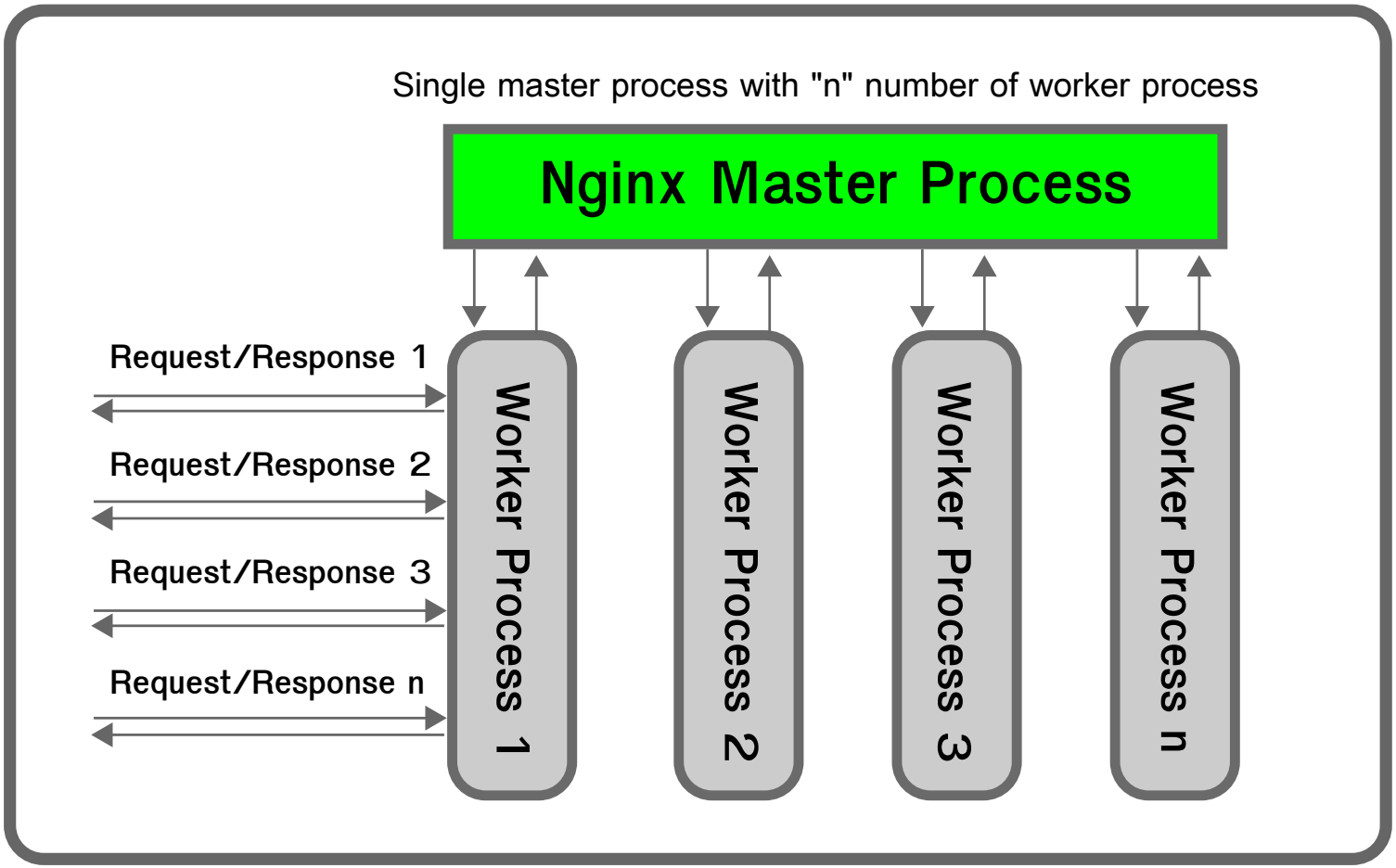


Figura: Architettura di NGINX

Outline

Repository

- **Debian/Ubuntu**

```
sudo apt-get install nginx
```

- **CentOS/RHEL**

```
sudo su -c 'rpm -Uvh http://dl.fedoraproject.org/
    pub/epel/6/x86_64/epel-release-6-8.noarch.rpm'

sudo yum install nginx
```

- **Arch Linux**

```
sudo pacman -S nginx
```

Nota per Debian/Ubuntu

A partire da **Debian 7 Wheezy** vengono forniti 4 diversi pacchetti di NGINX

- **nginx-light**: contiene soltanto un minimo numero di core modules
- **nginx-full**: contiente tutti i core modules (è quello che viene installato quando si installa il metapacchetto nginx)
- **nginx-extras**: contiente tutti i core modules più alcuni moduli extra
- nginx-naxsi: contiene il modulo per naxsi Web Application Firewall

`https://wiki.debian.org/Nginx`

Sorgenti

È possibile installare installare NGINX anche **compilandolo dai sorgenti** che possono essere scaricati dal sito:
`http://nginx.org/en/download.html`

Tale metodo di installazione è utile se:

Sorgenti

È possibile installare installare NGINX anche **compilandolo dai sorgenti** che possono essere scaricati dal sito:
`http://nginx.org/en/download.html`

Tale metodo di installazione è utile se:

- Si vuole installare NGINX abilitando moduli che non sono presenti nella versione fornita dalla distribuzione in uso

Sorgenti

È possibile installare installare NGINX anche **compilandolo dai sorgenti** che possono essere scaricati dal sito:
`http://nginx.org/en/download.html`

Tale metodo di installazione è utile se:

- Si vuole installare NGINX abilitando moduli che non sono presenti nella versione fornita dalla distribuzione in uso
- La versione precompilata fornita dalla distribuzione è **troppo datata**

Sorgenti

È possibile installare installare NGINX anche **compilandolo dai sorgenti** che possono essere scaricati dal sito:
`http://nginx.org/en/download.html`

Tale metodo di installazione è utile se:

- Si vuole installare NGINX abilitando moduli che non sono presenti nella versione fornita dalla distribuzione in uso
- La versione precompilata fornita dalla distribuzione è **troppo datata**
- Si vogliono disabilitare moduli che sappiamo non verrebbero usati, riducendo ulteriormente dimensioni e consumi di NGINX

Info sulla versione installata

Se volessimo conoscere la versione di NGINX installata oppure i moduli che sono stati abilitati in fase di compilazione (utile se non siamo stati noi a compilare) possiamo usare i seguenti comandi:

Info sulla versione installata

Se volessimo conoscere la versione di NGINX installata oppure i moduli che sono stati abilitati in fase di compilazione (utile se non siamo stati noi a compilare) possiamo usare i seguenti comandi:

- Per visualizzare la **versione installata**

```
sudo nginx -v
```

Info sulla versione installata

Se volessimo conoscere la versione di NGINX installata oppure i moduli che sono stati abilitati in fase di compilazione (utile se non siamo stati noi a compilare) possiamo usare i seguenti comandi:

- Per visualizzare la **versione installata**

```
sudo nginx -v
```

- Per avere la lista completa dei **moduli abilitati**

```
sudo nginx -V
```

Info sulla versione installata

Se volessimo conoscere la versione di NGINX installata oppure i moduli che sono stati abilitati in fase di compilazione (utile se non siamo stati noi a compilare) possiamo usare i seguenti comandi:

- Per visualizzare la **versione installata**

```
sudo nginx -v
```

- Per avere la lista completa dei **moduli abilitati**

```
sudo nginx -V
```

- Per **cercare un modulo** in particolare tra quelli abilitati

```
sudo nginx -V 2>&1 | grep --color nome_modulo
```

Outline

nginx.conf

Come abbiamo detto nell'introduzione, la semplicità nella configurazione è stato da sempre uno degli obiettivi di NGINX.

nginx.conf

Come abbiamo detto nell'introduzione, la semplicità nella configurazione è stato da sempre uno degli obiettivi di NGINX.

- Interamente configurabile mediante un unico file

nginx.conf

Come abbiamo detto nell'introduzione, la semplicità nella configurazione è stato da sempre uno degli obiettivi di NGINX.

- Interamente configurabile mediante un unico file
- In Debian il file si trova in **/etc/nginx/nginx.conf**

nginx.conf

Come abbiamo detto nell'introduzione, la semplicità nella configurazione è stato da sempre uno degli obiettivi di NGINX.

- Interamente configurabile mediante un unico file
- In Debian il file si trova in **/etc/nginx/nginx.conf**
- È comunque possibile **spezzare il file** in tanti file diversi richiamabili con "include"

nginx.conf

Come abbiamo detto nell'introduzione, la semplicità nella configurazione è stato da sempre uno degli obiettivi di NGINX.

- Interamente configurabile mediante un unico file
- In Debian il file si trova in **/etc/nginx/nginx.conf**
- È comunque possibile **spezzare il file** in tanti file diversi richiamabili con "include"
- Sintassi in stile programmativo (riga termina con ";" , blocchi di impostazioni inclusi tra graffe {})

nginx.conf

Come abbiamo detto nell'introduzione, la semplicità nella configurazione è stato da sempre uno degli obiettivi di NGINX.

- Interamente configurabile mediante un unico file
- In Debian il file si trova in **/etc/nginx/nginx.conf**
- È comunque possibile **spezzare il file** in tanti file diversi richiamabili con "include"
- Sintassi in stile programmativo (riga termina con ";" , blocchi di impostazioni inclusi tra graffe {})
- Al termine della configurazione diamo **sudo nginx -t && sudo service nginx reload** per controllare errori sintattici e attivare le modifiche

Context

- Il file di configurazione di NGINX è diviso in **context** che sarebbero blocchi di direttive racchiusi tra parentesi graffe

Context

- Il file di configurazione di NGINX è diviso in **context** che sarebbero blocchi di direttive racchiusi tra parentesi graffe
- Esistono 5 context e sono: **main, events, http, server e location**

Context

- Il file di configurazione di NGINX è diviso in **context** che sarebbero blocchi di direttive racchiusi tra parentesi graffe
- Esistono 5 context e sono: **main, events, http, server e location**
- Nel file di configurazione possiamo avere **un solo blocco** events e http ma **molteplici** blocchi server e location

Context

- Il file di configurazione di NGINX è diviso in **context** che sarebbero blocchi di direttive racchiusi tra parentesi graffe
- Esistono 5 context e sono: **main, events, http, server e location**
- Nel file di configurazione possiamo avere **un solo blocco** events e http ma **molteplici** blocchi server e location
- I context location si trovano sempre all'interno di context server che a loro volta risiedono nel context http

Context

- Il file di configurazione di NGINX è diviso in **context** che sarebbero blocchi di direttive racchiusi tra parentesi graffe
- Esistono 5 context e sono: **main, events, http, server e location**
- Nel file di configurazione possiamo avere **un solo blocco** events e http ma **molteplici** blocchi server e location
- I context location si trovano sempre all'interno di context server che a loro volta risiedono nel context http
- I context sono **ereditari,** cioè ogni context eredita le impostazioni del context in cui viene inserito.

Struttura di nginx.conf

```
user www-data;
worker_processes 1;
pid /run/nginx.pid;

events {
worker_connections 128;
}

http {
...
...
    server {
    ...
      location {
          ...
          }
    }
}
```

main e events

- **user www-data:** specifichiamo da quale utente saranno lanciati i processi workers (es. www-data)
- **worker_processes:** con questa direttiva decidiamo quanti processi worker avviare, solitamente si consiglia **uno per core**
- **pid /run/nginx.pid:** diciamo a NGINX in quale file salvare il proprio PID
- **worker_connections 128:** Impostiamo il numero di connessioni che può gestire ciascun worker (dobbiamo trovare il valore ideale facendo dei test ad esempio con httperf)

Blocco http

- È il blocco di configurazione **globale**, le direttive inserite in questo blocco hanno effetto su tutti i siti serviti da NGINX

Blocco http

- È il blocco di configurazione **globale**, le direttive inserite in questo blocco hanno effetto su tutti i siti serviti da NGINX
- Generalmente le impostazioni di default non pregiudicano il funzionamento di NGINX ma è sempre bene **modificarle in base alle nostre necessità**

Blocco http

- È il blocco di configurazione **globale**, le direttive inserite in questo blocco hanno effetto su tutti i siti serviti da NGINX
- Generalmente le impostazioni di default non pregiudicano il funzionamento di NGINX ma è sempre bene **modificarle in base alle nostre necessità**
- Contiene i blocchi server

Blocchi server

- È il blocco usato per configuare i **Virtual Domains** (VirtualHosts per chi è pratico di Apache)

Blocchi server

- È il blocco usato per configuare i **Virtual Domains** (VirtualHosts per chi è pratico di Apache)
- Avremo un blocco server per ogni sito (dominio o sottodominio) hostato sul server

Blocchi server

- È il blocco usato per configuare i **Virtual Domains** (VirtualHosts per chi è pratico di Apache)
- Avremo un blocco server per ogni sito (dominio o sottodominio) hostato sul server
- Insieme al blocco location è quello che andremo a configurare manualmente ogni volta che vogliamo far girare qualcosa su NGINX

VirtualHosts

```
http{
    server {
        listen 80;
        server_name example.org www.example.org;
        ...
    }

    server {
        listen 80;
        server_name poul.org www.poul.org;
        ...
    }

    server {
        listen 80;
        server_name antani.org www.antani.org;
        ...
    }
}
```

Blocchi location

- In questi blocchi troviamo le direttive che operano su file e cartelle

Blocchi location

- In questi blocchi troviamo le direttive che operano su file e cartelle
- Queste direttive ci permettono di decidere come NGINX debba comportarsi quando viene richiesta una specifica risorsa

Blocchi location

- In questi blocchi troviamo le direttive che operano su file e cartelle
- Queste direttive ci permettono di decidere come NGINX debba comportarsi quando viene richiesta una specifica risorsa
- È possibile utilizzare un percorso specifico per indicare una risorsa
  - In questo caso useremo il prefisso “=”

Blocchi location

- In questi blocchi troviamo le direttive che operano su file e cartelle
- Queste direttive ci permettono di decidere come NGINX debba comportarsi quando viene richiesta una specifica risorsa
- È possibile utilizzare un percorso specifico per indicare una risorsa
  - In questo caso useremo il prefisso “=”
- In alternativa possiamo eseguire il matching utilizzando le **regex**
  - Prefisso **"~"** se vogliamo che il matching sia case sensitive
  - Prefisso **"~*"** se vogliamo che il matching sia case insensitive

```
Syntax: location [ = | ~ | ~* | ^~ ] uri { ... }
```

Alcuni esempi

```
location = / {    # this matches only the / query.
        .........
}
```

Alcuni esempi

```
location = / {   # this matches only the / query.
    .........
}
```

- Anti-hotlinking

```
location ~ .(gif|png|jpe?g)$ {
    valid_referers none blocked mywebsite.com *.mywebsite.com;
    if ($invalid_referer){
        return   403;
    }
}
```

Alcuni esempi

```
location  = / {   # this matches only the / query.
      .........
}
```

- Anti-hotlinking

```
location ~ .(gif|png|jpe?g)$ {
    valid_referers none blocked mywebsite.com *.mywebsite.com;
    if ($invalid_referer){
        return   403;
    }
}
```

- Impedire l'accesso a script eseguibili all'interno delle cartelle con permessi di scrittura

```
location ~* /(images|cache|media|logs|tmp)/.*.(php|pl|py|jsp|asp|sh
    return 403;
    error_page 403 /403_error.html;
}
```

Outline

CGI

- CGI o Common Gateway Interface, è un meccanismo ideato per permettere di realizzare e far girare applicazioni lato server, in **qualunque lingauggio** (C, PHP, Python, Perl, ...)

CGI

- CGI o Common Gateway Interface, è un meccanismo ideato per permettere di realizzare e far girare applicazioni lato server, in **qualunque lingauggio** (C, PHP, Python, Perl, ...)
- Le richieste generalmente vengono servite in questo modo:
  - Il web server riceve una richiesta per servire una certa pagina (relativa all'applicazione)

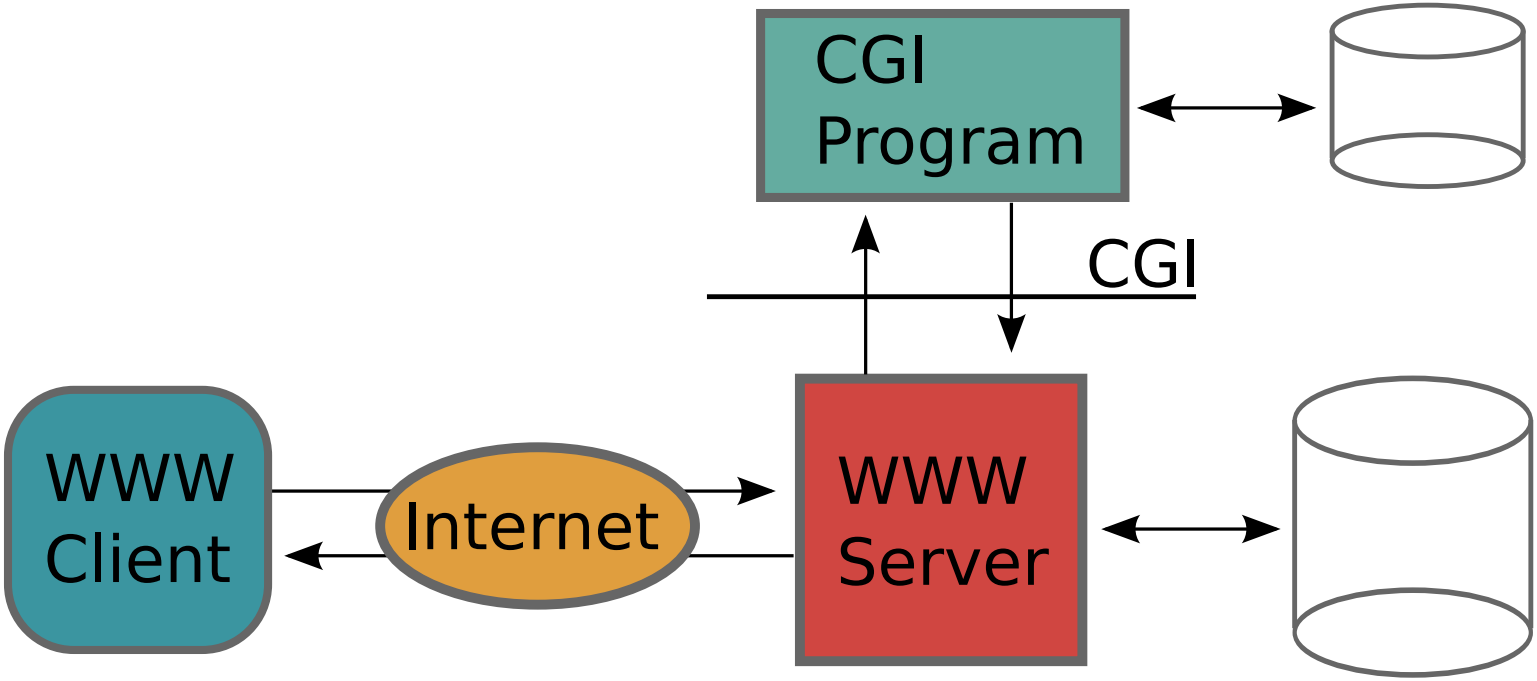
CGI

- CGI o Common Gateway Interface, è un meccanismo ideato per permettere di realizzare e far girare applicazioni lato server, in **qualunque lingauggio** (C, PHP, Python, Perl, ...)
- Le richieste generalmente vengono servite in questo modo:
  - Il web server riceve una richiesta per servire una certa pagina (relativa all'applicazione)
  - **Lancia un processo** tramite CGI per eseguire il programma e gli passa come parametri da linea di comando le varie variabili passate dalla GET/POST

CGI

- CGI o Common Gateway Interface, è un meccanismo ideato per permettere di realizzare e far girare applicazioni lato server, in **qualunque lingauggio** (C, PHP, Python, Perl, ...)
- Le richieste generalmente vengono servite in questo modo:
  - Il web server riceve una richiesta per servire una certa pagina (relativa all'applicazione)
  - **Lancia un processo** tramite CGI per eseguire il programma e gli passa come parametri da linea di comando le varie variabili passate dalla GET/POST
  - Il programma esegue il codice e restituisce l'output al web server che inoltra la risposta al client

Architettura di CGI



FastCGI - l'evoluzione

FastCGI è la diretta evoluzione di CGI, riprende alcune delle sue funzionalità migliorando però due aspetti importanti:

FastCGI - l'evoluzione

FastCGI è la diretta evoluzione di CGI, riprende alcune delle sue funzionalità migliorando però due aspetti importanti:

- **Performance:** I processi lanciati da FastCGI sono persistenti e vengono riutilizzati per servire diverse richieste. Viene creato un socket per scambiare le richieste tra web server e applicazione. Si occupa poi l'applicazione di generare eventualmente altri processi.

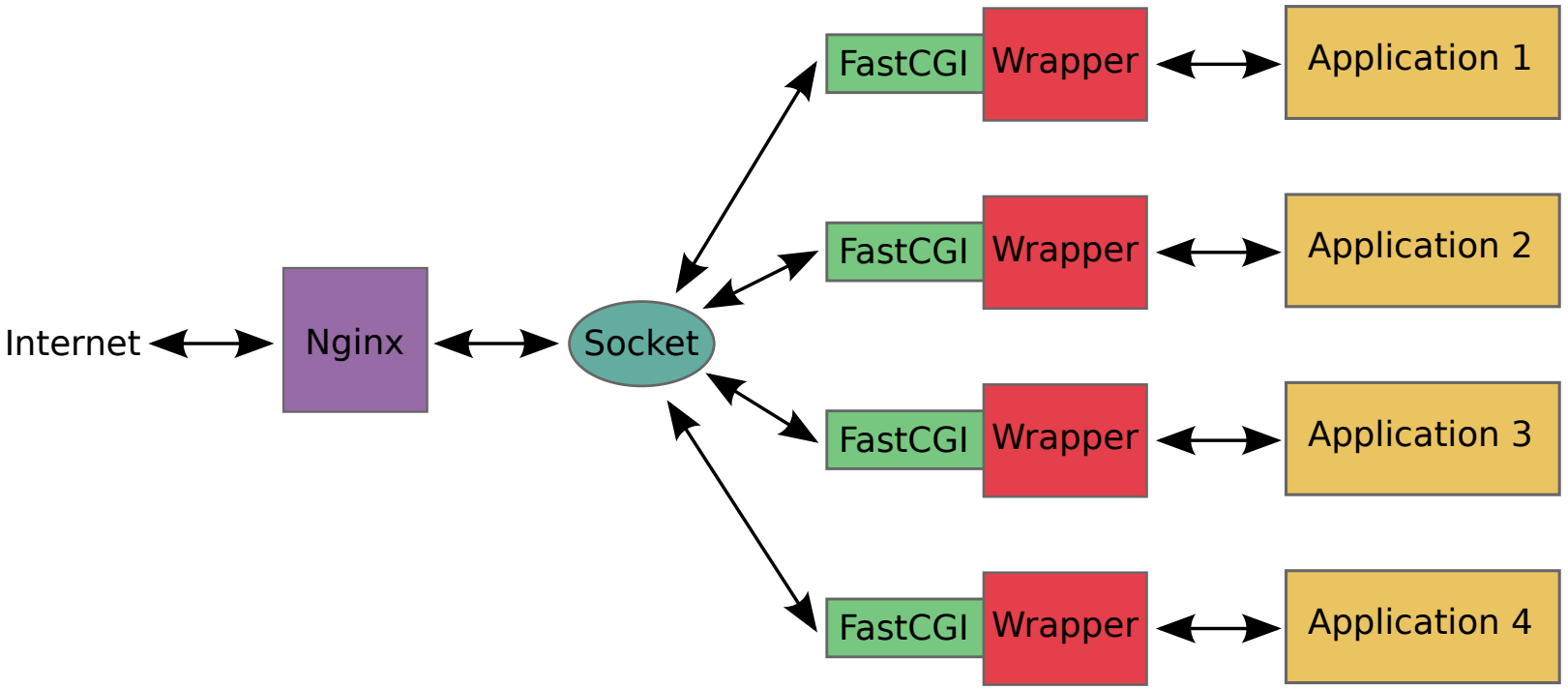
FastCGI - l'evoluzione

FastCGI è la diretta evoluzione di CGI, riprende alcune delle sue funzionalità migliorando però due aspetti importanti:

- **Performance:** I processi lanciati da FastCGI sono persistenti e vengono riutilizzati per servire diverse richieste. Viene creato un socket per scambiare le richieste tra web server e applicazione. Si occupa poi l'applicazione di generare eventualmente altri processi.

- **Scalabiltà:** FastCGI infatti offre la possibilità di eseguire le applicazioni da remoto permettendo la distribuzione del carico di lavoro su server differenti.

Architettura di FastCGI



PHP-FPM

L'implementazione più famosa di fastCGI per PHP è **PHP-FPM** ed è quella utilizzeremo per NGINX. È presente nei repo di tutte le principali distribuzioni

PHP-FPM

L'implementazione più famosa di fastCGI per PHP è **PHP-FPM** ed è quella utilizzeremo per NGINX. È presente nei repo di tutte le principali distribuzioni

- **Debian/Ubuntu**

```
sudo apt-get install php5-fpm
```

- **Centos/RHEL**

```
sudo yum --enablerepo=remi install php-fpm
```

- **Arch Linux**

```
sudo pacman -S php-fpm
```

php-fpm.conf

- In Debian si trova in “**/etc/php5/fpm/php-fpm.conf**”

php-fpm.conf

- In Debian si trova in "**/etc/php5/fpm/php-fpm.conf**"
- Richiama però i file presenti in "**/etc/php5/fpm/pool.d/*.conf**" (di default troviamo solo **www.conf**)

php-fpm.conf

- In Debian si trova in “**/etc/php5/fpm/php-fpm.conf**”
- Richiama però i file presenti in “**/etc/php5/fpm/pool.d/*.conf**” (di default troviamo solo **www.conf**)
- All’interno di questo file ci interessa la stringa
  - “**listen = /var/run/php5-fpm.sock**”

Passiamo gli script PHP a FastCGI

Per abilitare l'esecuzione di codice PHP all'interno di un sito è sufficiente creare un apposito **blocco location** all'interno del context server interessato.

```
location ~ \.php$ {
    try_files $uri =404;
    fastcgi_pass unix:/var/run/php5-fpm.sock;
    fastcgi_index index.php;
    include fastcgi_params;
}
```

Analizziamo il blocco

- **“location ~ \.php$”:** Cerca tutti i file con estensione .php
- **“try_files $uri =404”:** Controlla che il file esista veramente per evitare l’esecuziona arbitraria di codice, se così non fosse restituisce un errore 404.
- **“fastcgi_pass unix:/var/run/php5-fpm.sock”:** Indichiamo il full path del socket di php-fpm
- **“fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name“:** In Debian dobbiamo aggiungere questa riga per definire l’absolute path dello script

Outline

Don't try this at home

The next time you visit a cafe to sip coffee and surf on some free Wi-Fi, try an experiment: Log in to some of your usual sites. Then, with a smile, hand the keyboard over to a stranger. Now walk away for 20 minutes. Remember to pick up your laptop before you leave.

Mike Shema

Cos'è HTTPS?

- HyperText Transfer Protocol over Secure Socket Layer.

Cos'è HTTPS?

- HyperText Transfer Protocol over Secure Socket Layer.
- È il risultato dell'applicazione di un protocollo di **crittografia asimmetrica e simmetrica** al protocollo di trasferimento di ipertesti HTTP.

Cos'è HTTPS?

- HyperText Transfer Protocol over Secure Socket Layer.
- È il risultato dell'applicazione di un protocollo di **crittografia asimmetrica e simmetrica** al protocollo di trasferimento di ipertesti HTTP.
- Viene utilizzato per garantire trasferimenti riservati di dati nel web, in modo da impedire intercettazioni dei contenuti che potrebbero essere effettuati tramite tecniche di attacco **man in the middle**.
  - Password
  - Dati sensibili
  - Furti di identità

Come funziona?

- Interpone tra il protocollo TCP e HTTP, un livello di crittografia/autenticazione come il Secure Sockets Layer (SSL) o il **Transport Layer Security (TLS)**.

Come funziona?

- Interpone tra il protocollo TCP e HTTP, un livello di crittografia/autenticazione come il Secure Sockets Layer (SSL) o il **Transport Layer Security (TLS)**.
- Crea quindi un **canale di comunicazione** criptato tra il client e il server attraverso uno scambio sicuro di **chiavi di cifratura**

Come funziona?

- Interpone tra il protocollo TCP e HTTP, un livello di crittografia/autenticazione come il Secure Sockets Layer (SSL) o il **Transport Layer Security (TLS)**.
- Crea quindi un **canale di comunicazione** criptato tra il client e il server attraverso uno scambio sicuro di **chiavi di cifratura**
- una volta stabilito questo canale al suo interno viene utilizzato il protocollo **HTTP** per la comunicazione

Come funziona?

- Interpone tra il protocollo TCP e HTTP, un livello di crittografia/autenticazione come il Secure Sockets Layer (SSL) o il **Transport Layer Security (TLS)**.
- Crea quindi un **canale di comunicazione** criptato tra il client e il server attraverso uno scambio sicuro di **chiavi di cifratura**
- una volta stabilito questo canale al suo interno viene utilizzato il protocollo **HTTP** per la comunicazione
- Impiega come porta di default la **443** e non la 80 come in HTTP (trasparentemente all'utente)

Bello...Lo voglio!

Possiamo facilmente creare dei cosiddetti **self-signed certificates** per proteggere il traffico da e verso i nostri siti.

- Creiamo la **cartella di destinazione** ed entriamoci

```
sudo mkdir -p /etc/nginx/ssl && cd /etc/nginx/ssl
```

- Creiamo la **private key SSL**

```
sudo openssl genrsa -out dominio.key 2048
```

- Creiamo il **certificate signing request (CSR)**

```
sudo openssl req -new -key dominio.key -out dominio.csr
```

- Creiamo il **certificato**

```
sudo openssl x509 -req -days 365 -in dominio.csr -signkey
    dominio.key -out dominio.crt
```

Abilitiamolo in nginx.conf

Reindirizziamo tutte le richieste HTTP sul protocollo HTTPS in questo modo:

```
server {
    listen 80;
    server_name dominio.com;
    return 301 https://$server_name$request_uri;
}


server {
    listen   443 ssl default_server;
    server_name dominio.com;

    ssl_certificate      /etc/nginx/ssl/dominio.crt;
    ssl_certificate_key /etc/nginx/ssl/dominio.key;
    ssl_protocols        TLSv1 TLSv1.1 TLSv1.2;
}
```

Un piccolo inconveniente

Il certificato di sicurezza del sito non è affidabile!

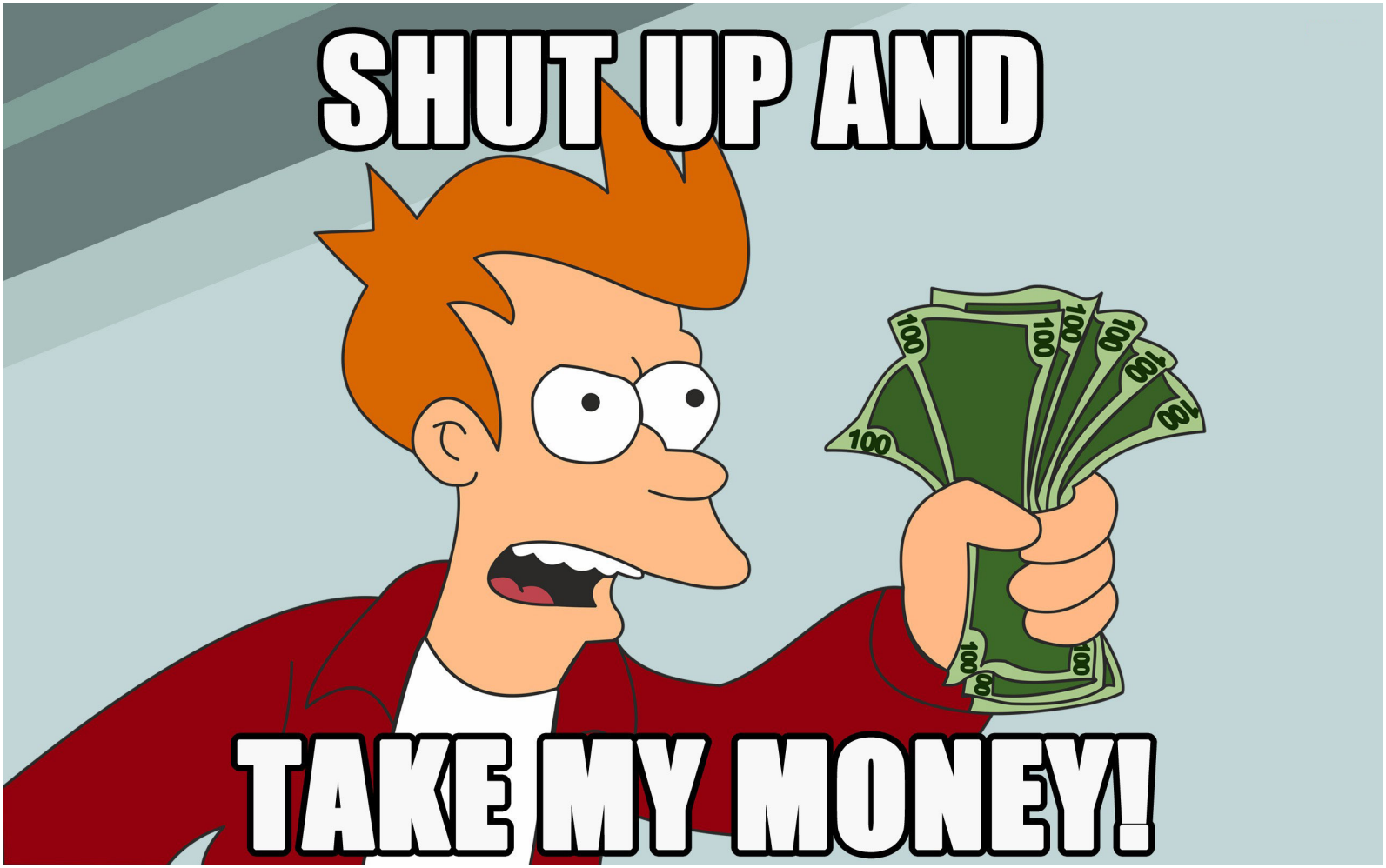
Hai tentato di accedere a **bmk.cippaciong.tk**, ma il server ha presentato un certificato emesso da un'autorità ritenuta non attendibile dal sistema operativo del computer. Questo potrebbe significare che il server ha generato le proprie credenziali di sicurezza, che Chrome non ritiene attendibili ai fini del riconoscimento dell'identità, o che un utente malintenzionato sta cercando di intercettare le tue comunicazioni.

Non procedere oltre, **soprattutto** se questo avviso non è mai stato visualizzato per questo sito.

Procedi comunque | Torna nell'area protetta

Due Soluzioni

Due Soluzioni



Link

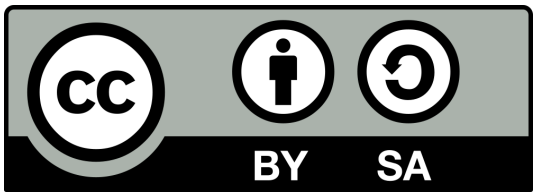
- **Getting Started**

NGINX Primer
NGINX Admin Guide
NGINX Secure Web Server Examples
NGINX Tips

- **SSL**

BetterCrypto
Why You Should Always Use HTTPS
StartSSL Free Certificates
Hardening NGINX SSL/TSL Configuration
Hardening Your Web Server's SSL Ciphers
5 easy tips to accelerate SSL

License



`http://www.poul.org/`