

Installazione e configurazione Web Server su Ubuntu Linux



Fabio Dainese
Matricola 857661



Università
Ca' Foscari
Venezia

Indice dei contenuti

- 1) Installazione Apache e configurazione Firewall
- 2) Installazione MySQL
- 3) Installazione PHP
- 4) Test PHP
- 5) Impostazione Virtual Host
- 6) Impostazione SSL
- 7) Impostazione sistema di Monitoring
- 8) Progettazione e realizzazione form PHP

LAMP

1) Installazione Apache e configurazione Firewall

PREREQUISITI:

Per questioni di sicurezza è sempre bene utilizzare un account non-root con privilegi sudo per eseguire le successive operazioni nel nostro server.

```
$ adduser fabio  
$ usermod -aG sudo fabio
```

1) Installazione Apache e configurazione Firewall

Il web server Apache è il più popolare al mondo; possiamo facilmente installarlo su Ubuntu utilizzando il suo packet manager apt:

```
$ sudo apt-get update  
$ sudo apt-get install apache2
```

Utilizziamo update per re-sincronizzare i file degli indici dei packages dalle loro fonti.

1) Installazione Apache e configurazione Firewall

Assegniamo un ServerName globale per evitare warning aggiungendo una nuova linea al file:

/etc/apache2/apache2.conf

```
$ sudo nano /etc/apache2/apache2.conf
```

```
...
```

```
ServerName server_domain_or_IP
```

Verifichiamo gli errori di sintassi:

```
$ sudo apache2ctl configtest
```

1) Installazione Apache e configurazione Firewall

Se l'output del precedente comando ha riportato:

Syntax OK

Possiamo procedere riavviando Apache:

```
$ sudo systemctl restart apache2
```

1) Installazione Apache e configurazione Firewall

Procediamo quindi a regolare il Firewall (UFW) per consentire il traffico web HTTP e HTTPS. Verifichiamo se UFW ha un profilo per l'applicazione di Apache:

```
$ sudo ufw app list
```

Available applications:

Apache

Apache Full

Apache Secure

OpenSSH

1) Installazione Apache e configurazione Firewall

Se controlliamo il profilo Apache Full dovremmo ottenere un'abilitazione del traffico per la porta 80 e 443:

```
$ sudo ufw app info "Apache Full"
```

Abilitiamo quindi il traffico in entrata per questo profilo:

```
$ sudo ufw allow in "Apache Full"
```

2) Installazione MySQL

Ora che abbiamo un Web Server avviato, installiamo un database management system come MySQL:

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

Al termine dell'installazione avviamo un semplice script che rimuoverà alcuni valori di default pericolosi:

```
$ sudo mysql_secure_installation
```

Seguirà una semplice ed intuitiva procedura di configurazione guidata volta ad aumentare la sicurezza del nostro sistema.

3) Installazione PHP

Durante l'installazione di PHP includiamo degli ulteriori pacchetti «d'aiuto» che ci permetteranno di eseguire il nostro codice sotto server Apache e di comunicare con il nostro database MySQL:

```
$ sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql
```

- *libapache2-mod-php* → per Apache
- *php-mcrypt* → (sostituisce Unix crypt) per la sicurezza
- *php-mysql* → per il database MySQL

3) Installazione PHP

Di default Apache cerca un index file chiamato *index.html*, ora che abbiamo installato PHP potremmo preferire un index con estensione *.php* rispetto a *.html*. Modifichiamo nel seguente modo:

```
$ sudo nano /etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl
    index.xhtml index.htm
</IfModule>
```

Il file *dir.conf* spostando al primo posto *index.php*

4) Test PHP

Riavviamo un'ulteriore volta Apache:

```
$ sudo systemctl restart apache2
```

Verifichiamo inoltre lo stato di Apache utilizzando:

```
$ sudo systemctl status apache2
```

Se tutto risulta correttamente installato, sarà possibile installare ulteriori moduli PHP (come php-cli).

4) Test PHP

Per testare se il nostro sistema è configurato correttamente per PHP, scriviamo un semplice script *info.php* che inseriremo nella nostra «web root», contenente:

```
$ sudo nano /var/www/html/info.php
<?php
phpinfo();
?>
```

Tramite browser verifichiamo:

http://indirizzo_ip_server/info.php

5) Impostazione Virtual Host

Uno dei componenti più rilevanti di Apache è il Virtual Host, che permette all'amministratore di un server di «hostare» domini multipli attraverso una singola interfaccia od un singolo IP utilizzando un meccanismo di confronto.

Ogni dominio configurato (tramite DNS) sarà quindi in grado di indirizzare i visitatori in una specifica cartella contenente le informazioni del sito.

5) Impostazione Virtual Host

Innanzitutto prepariamo la struttura di directory che dovrà ospitare i siti legati ai nostri domini web di esempio:

```
$ sudo mkdir -p /var/www/example.com/public_html  
$ sudo mkdir -p /var/www/test.com/public_html
```

5) Impostazione Virtual Host

Permettiamo agli utenti regolari di modificare i file nelle nostre web directory:

```
$ sudo chown -R $USER:$USER /var/www/example.com/  
public_html
```

```
$ sudo chown -R $USER:$USER /var/www/test.com/  
public_html
```

Dove \$USER è una variabile che assumerà il valore dell'utente attualmente loggato. Modifichiamo inoltre i permessi generali:

```
$ sudo chmod -R 755 /var/www
```

5) Impostazione Virtual Host

Per facilitarne il riconoscimento, creiamo una pagina di esempio per entrambi i nostri Virtual Host:

```
$ nano /var/www/example.com/public_html/index.html
```

```
...
```

```
$ nano /var/www/test.com/public_html/index.html
```

```
...
```

5) Impostazione Virtual Host

Passiamo finalmente alla creazione dei nostri virtual host files che conterranno appunto la configurazione dei nostri Virtual Host per Apache. Copiamo il file di configurazione *000-default.conf* di default fornito da Apache:

```
$ sudo cp /etc/apache2/sites-available/000-default.conf  
/etc/apache2/sites-available/example.com.conf  
$ sudo nano /etc/apache2/sites-available/  
example.com.conf
```

5) Impostazione Virtual Host

Il file di configurazione di default si presenterà nel seguente modo (senza i commenti):

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

5) Impostazione Virtual Host

Modifichiamo quindi il file per il nostro primo Virtual Host nel seguente modo:

```
<VirtualHost *:80>
    ServerAdmin admin@example.com
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example.com/public_html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

5) Impostazione Virtual Host

Eseguiamo quindi le stesse operazioni per il nostro secondo dominio, ottenendo:

```
<VirtualHost *:80>
    ServerAdmin admin@test.com
    ServerName test.com
    ServerAlias www.test.com
    DocumentRoot /var/www/test.com/public_html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

5) Impostazione Virtual Host

Non ci resta che abilitare i due file di configurazione appena creati, utilizzando lo strumento *a2ensite*:

```
$ sudo a2ensite example.com.conf  
$ sudo a2ensite test.com.conf
```

Riavviamo Apache:

```
$ sudo service apache2 restart
```

5) Impostazione Virtual Host

Verifichiamo quindi tramite browser se tutto funziona correttamente accedendo alle pagine di esempio precedentemente create ed ora raggiungibili attraverso i nostri indirizzi:

<http://example.com>

<http://test.com>

5) Impostazione Virtual Host

NOTA: Se nel nostro server avessimo installato un gestore per il nostro database MySQL, sarebbe ora necessario configurarlo per i nuovi Virtual Host.

Nel caso di *PhpMyAdmin*, procediamo aggiungendo un alias dopo l'istruzione *DocumentRoot*:

```
...  
Alias /phpmyadmin /usr/share/phpmyadmin
```

6) Impostazione SSL

Utilizziamo Let's Encrypt per ottenere un certificato TLS/SSL condiviso e gratuito per il nostro server:

```
$ sudo apt-get update  
$ sudo apt-get install python-letsencrypt-apache
```

Il client *letsencrypt* è ora pronto ad essere utilizzato.

6) Impostazione SSL

Generiamo un certificato SSL per Apache fornendo i seguenti parametri:

```
$ sudo letsencrypt --apache -d example.com
```

Se il sito presenta sottodomini, sarà opportuno specificarli modificando il precedente comando come segue:

```
$ sudo letsencrypt --apache -d example.com -d  
www.example.com
```

6) Impostazione SSL

Al termine dell'installazione il certificato sarà disponibile in `/etc/letsencrypt/live`

NOTA:

Per la gestione dei Virtual Host è consigliato creare un certificato separato per ogni nome dominio.

6) Impostazione SSL

Sarà in seguito vantaggioso creare uno script per il rinnovo dei certificati, oppure utilizzare l'apposita funzione:

```
$ sudo letsencrypt renew  
Processing /etc/letsencrypt/renewal/example.com.conf  
Processing /etc/letsencrypt/renewal/test.com.conf
```

The following certs are not due for renewal yet:

```
/etc/letsencrypt/live/example.com/fullchain.pem  
/etc/letsencrypt/live/test.com/fullchain.pem
```

...

6) Impostazione SSL



Automatically enable HTTPS on your website with EFF's Certbot,
deploying Let's Encrypt certificates.

I'm using

Apache

on

System

To get instructions for other software and systems, select from the dropdown menus above. Certbot can automatically choose the best method for your setup. If you prefer more control, select "I'm not sure" or "None of the above" if you want less automation and more control.

- Software
- Apache
- Nginx
- Haproxy
- Plesk
- None of the above



7) Impostazione sistema di Monitoring

Utilizziamo Nagios per monitorare i nostri hosts.

Procediamo all'installazione ed alla configurazione di base consultando le guide presenti nel sito ufficiale, nel caso di *Ubuntu* utilizziamo questa guida:

<https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/quickstart-ubuntu.html>

7) Impostazione sistema di Monitoring

Una volta installato e configurato Nagios, procediamo ad aggiungere i vari hosts da monitorare.

Creiamo un nuovo file di configurazione per ogni hosts remoto che vogliamo monitorare:

```
$ sudo vi /usr/local/nagios/etc/servers/mio_host1.cfg
```

7) Impostazione sistema di Monitoring

Modifichiamo quindi il file per ogni hosts, ad esempio:

```
define host {  
    use          linux-server  
    host_name   test.com  
    alias       test.com  
    address     www.test.com  
}  
...
```

7) Impostazione sistema di Monitoring

Aggiungiamo vari servizi, come «Ping»:

```
define service {  
    use generic-service  
    host_name test.com  
    service_description PING  
    check_command check_ping!100.0,20%!500.0,60%  
}  
...
```

7) Impostazione sistema di Monitoring

Altri servizi potrebbero essere HTTP ed SSH check:

```
define service {  
    use generic-service  
    host_name test.com  
    service_description SSH  
    check_command check_ssh  
    notifications_enabled 0  
}  
...
```

7) Impostazione sistema di Monitoring

Al termine delle configurazioni, riavviamo il servizio:

```
$ sudo service nagios reload
```

Ed acconsentiamo, attraverso browser, a Nagios di verificare la corretta installazione dei nuovi hosts e servizi.

8) Progettazione e realizzazione form PHP

index.php

```
1 <?php
2 // Includo lo script di login
3 include ('login.php');
4
5 if (isset($_SESSION['login_user']))
6 {
7 header("location: profile.php");
8 }
9
10 ?>
11 <!DOCTYPE html>
12 <html>
13
14 <head>
15 |...<title>Login Form</title>
16 </head>
17
18 <body>
19 |...<div id="main">
20 |...|...<h1>PHP Login Session</h1>
21 |...|...<div id="login">
22 |...|...|...<h2>Login Form</h2>
23 |...|...|...<form action="index.php" method="post">
24 |...|...|...|...<label>UserName: </label>
25 |...|...|...|...<input id="name" name="username" placeholder="username" type="text">
26 |...|...|...|...<label>Password: </label>
27 |...|...|...|...<input id="password" name="password" placeholder="*****" type="password">
28 |...|...|...|...<input name="submit" type="submit" value="Login">
29 |...|...|...|...<!-- Eventuali errori di login --&gt;
30 |...|...|...|...&lt;span&gt;&lt;?php echo $error; ?&gt;&lt;/span&gt;
31 |...|...|...&lt;/form&gt;
32 |...|...&lt;/div&gt;
33 |...&lt;/div&gt;
34 &lt;/body&gt;
35
36 &lt;/html&gt;</pre>
```

8) Progettazione e realizzazione form PHP

login.php

```
1 <?php
2 session_start(); // Avvio sezione
3 $error = ''; // Variabile per memorizzare errori di login
4
5 if (isset($_POST['submit']))
6 {
7 if (empty($_POST['username']) || empty($_POST['password']))
8 {
9 $error = "Username o Password non validi!";
10 }
11 else
12 {
13
14 // Definisco $username e $password
15 $username = $_POST['username'];
16 $password = $_POST['password'];
17
18 // Stabilisco una semplice connessione al server
19 $connection = mysql_connect("localhost", "root", "");
20
21 // Per proteggersi da MySQL injection
22 $username = stripslashes($username);
23 $password = stripslashes($password);
24 $username = mysql_real_escape_string($username);
25 $password = mysql_real_escape_string($password);
26
27 // Seleziono il database
28 $db = mysql_select_db("miodb", $connection);
29
30 // Eseguo la query
31 $query = mysql_query("select * from login where password='$password' AND username='$username'", $connection);
32 $rows = mysql_num_rows($query);
33 if ($rows == 1)
34 {
35 $_SESSION['login_user'] = $username; // Inizializzo sessione con username
36 header("location: profile.php"); // Faccio un redirect php alla pagina del profilo
37 }
38 else
39 {
40 $error = "Username o Password non validi!";
41 }
42
43 mysql_close($connection); // Chiudo la connessione al server
44 }
45 }
```

8) Progettazione e realizzazione form PHP

Hashing delle password:

```
1 <?php
2 require 'password.php';
3
4 $passwordHash = password_hash('secret-password', PASSWORD_DEFAULT);
5
6 if (password_verify('bad-password', $passwordHash)) {
7 |...| // password corretta
8 } else {
9 |...| // password sbagliata
10 }
11 |
```

```
$password = $_POST['password'];
$hashed_password = password_hash($password, PASSWORD_DEFAULT);
var_dump($hashed_password);
if(password_verify($password, $hashed_password)) {
    // ...
}
```

Approfondimento:

http://www.phptherightway.com/#password_hashing