How To LEMP su CentOS 7

How To: Installare Linux, Nginx, MySQL, PHP 7 (LEMP) su CentOS 7

Un ambiente LEMP a differenza dell'ambiente LAMP (Linux, Apache, MySQL, PHP) differisce per il web server ENGINX.

In questo HOW TO andremo a vedere come installare PHP 7 FPM con Nginx per avere le massime prestazioni.

Il primo passo da fare è disabilitare il SELINUX, la cosa migliore sarebbe configurarlo ad hoc, ma questa operazione va eseguite prima della messa in esercizio del server.

Primo step installare Nginx

Come primo passaggio installiamo il repository EPEL e IUS. Utilizziamo il comodissimo script messo a disposizione sul sito ius.io per fare prima. Il curl ci aiuta a scaricare il setup.

```
curl 'https://setup.ius.io/' -o setup-ius.sh
```

ora eseguiamo lo script:

```
bash setup-ius.sh
```

Ora possiamo installare NGINX

```
yum install nginx
```

avviamo il webserver con il comando systemctl

```
systemctl start nginx
```

ora possiamo provare puntando nel nostro browser

```
http://indirizzo_ip_del_server/
```

Se vedi questa pagina allora possiamo abilitare NGINX all'avvio del sistema,

al BOOT:

```
systemctl enable nginx
```

Secondo Step installare MySQL

Abbiamo due possibilità, installare MySQL o MariaDB, sono la stessa cosa nati dallo stesso adre **Ulf Michael Widenius** noto anche come **Monty.**

In questa guida opteremo per MySQL, il comando è il seguente:

```
yum install mysql-server mysql
```

Ora passiamo alla prima configurazione del nostro RDBMS: start del demone:

```
service mysqld status
```

e poi messa in sicurezza di base:

```
mysql_secure_installation
```

Siamo pronti per abilitare anche MySQL al boot:

```
systemctl enable mysqld
```

Terzo Step installazione di PHP-FPM 7

Ora passiamo all'installazione dei PHP-FPM (FastCGI Process Manager) 7, l'ultima versione del php disponibile ad oggi eseguito sulla porta 9000:

```
yum install php70u-fpm-nginx php70u-cli php70u-mysqlnd
```

installato apriamo il file di configurazione e sostituiamo l'utente e i l gruppo d'esecuzione:

```
vim /etc/php-fpm.d/www.conf
```

```
; When POSIX Access Control Lists are supported you can set them using
; these options, value is a comma separated list of user/group names.
; When set, listen.owner and listen.group are ignored
;listen.acl_users = apache,nginx
;listen.acl_users = apache
listen.acl_users = nginx
;listen.acl_groups =
```

a questo punto riavviamo creiamo un vhosts, per prima cosa per tenere in ordine il nostro ambiente posizioniamo i file dei virtual hosts in una directory:

```
mkdir /etc/nginx/sites-available
```

passiamo al file nginx.conf l’istruzione di leggere il contenuto della nuova direcotry

```
vim /etc/nginx/nginx.conf
```

aggiungendo la riga

```
include /etc/nginx/sites-enabled/*;
```

```
;server {
listen 81.127.13.234:80;
server_name stat.lbit-solution.it;
location / {
try_files $uri $uri/ =404;
root /var/www/vhosts/lbit-solution.it/stat.lbit-solution.it/;
index index.php index.html index.htm;
}

error_page 404 /404.html;
error_page 500 502 503 504 /50x.html;
location = /50x.html {
root /var/www/vhosts/lbit-solution.it/stat.lbit-solution.it/;
}

location ~ \.php$ {
root /var/www/vhosts/lbit-solution.it/stat.lbit-solution.it/;
fastcgi_pass 127.0.0.1:9000;
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME /var/www/vhosts/lbit-solution.it/stat.lbit-
solution.it$fastcgi_script_name;
include fastcgi_params;
}
}

server {
listen 81.127.13.234:443 ssl;
server_name stat.lbit-solution.it;

### SSL cert files ###
```

```
ssl_certificate /var/www/vhosts/lbit-solution.it/ssl/stat.lbit-
solution.it.crt;
ssl_certificate_key /var/www/vhosts/lbit-solution.it/ssl/stat.lbit-
solution.it.key;

### Add SSL specific settings here ###

ssl_protocols SSLv3 TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers RC4:HIGH:!aNULL:!MD5;
ssl_prefer_server_ciphers on;
keepalive_timeout 60;
ssl_session_cache shared:SSL:10m;
ssl_session_timeout 10m;

### SSL log files ###
access_log /var/www/vhosts/lbit-solution.it/logs/stat.lbit-solution.it.ssl-
access.log;
error_log /var/www/vhosts/lbit-solution.it/logs/stat.lbit-solution.it.ssl-
error.log;

location / {
try_files $uri $uri/ =404;
root /var/www/vhosts/lbit-solution.it/stat.lbit-solution.it/;
index index.php index.html index.htm;
}

error_page 404 /404.html;
error_page 500 502 503 504 /50x.html;
location = /50x.html {
root /var/www/vhosts/lbit-solution.it/stat.lbit-solution.it/;
}

location ~ \.php$ {
root /var/www/vhosts/lbit-solution.it/stat.lbit-solution.it/;
fastcgi_pass 127.0.0.1:9000;
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME /var/www/vhosts/lbit-solution.it/stat.lbit-
solution.it$fastcgi_script_name;
include fastcgi_params;
}
}
```

Prepariamo le direcotry del virtual hosts:

```
mkdir -p /var/www/vhosts/lbit-solution.it/{ssl,logs,httpdocs,stat.lbit-
solution.it}
```

E ora riavviamo php-fpm e Nginx

```
sudo systemctl restart php-fpm
sudo systemctl restart nginx
```

Vulnerabilità WordPress SEO by Yoast

Vulnerabilità SEO, ma come risolvere i danni fatti?

Come scritto nel blog ufficiale (LINK) il blasonato plugin SEO by Yoast soffre di una gravissima vulnerabilità: **Blind SQL Injection,** file interessato sarebbe il ***class-bulk-editor-list-table.php***.

Cos'è una **Blind SQL Injection** e come possiamo sfruttarla?

Un hacker inserisce una query SQL non valida in un'applicazione, nel nostro caso **WordPress** che avendo un autore, un admin o un editor già autenticati che visitano un URL malformato, il malintenzionato riesce ad accedere e modificare il database **WordPress.**

Vediamo cosa è successo proprio a noi che scriviamo questo articolo.

Il furbo di turno ha sfruttato la vulnerabilità per modificare il database, creare un nuovo utente, concedergli i privilegi amministrativi ed aggiungere delle widget con codice javascript.

Tale codice servire a modificare i link del sito per rimandare a pubblicità, il modo più veloce per monetizzare. Fortunatamente l'hacker aveva un suo scopo ben preciso, quello di monetizzare, per questo motivo non ha fatto danni.

Questo serve a riflettere su quanti usano WordPress per scopi professionali senza affidarsi ad aziende o professionisti del settore.

Come suggerito da hostingtalk.it:

> In casi come questi, il consiglio è di **aggiornare immediatamente** il plugin WordPress SEO by Yoast all'ultima versione disponibile o di **affidarsi a servizi di hosting gestiti**, che eseguono in automatico per l'utenza gli upgrade di sicurezza necessaria. Altra alternativa è **l'autoaggiornamento di WordPress**, sempre che non sia stato disabilitato.

In alternativa un contratto di manutenzione può salvare il proprio business.

Size of MySQL database

Vogliamo sapere lo spazio occupato da ogni singolo database usando la command line, una semplice query restituisce a video l'informazione richiesta.

Prestate attenzione perché questa query potrebbe richiedere molto tempo per DB di grandi dimensioni.

```
mysql> SELECT table_schema "DB Name",
Round(Sum(data_length + index_length) / 1024 / 1024, 1) "DB Size in MB"
FROM information_schema.tables
GROUP BY table_schema;
```

Ecco un esempio dell'output:

```
+--------------------------------+---------------+
| DB Name | DB Size in MB |
+--------------------------------+---------------+
| monitoraggio | 1505.0 |
| mysql | 0.7 |
| pcparts | 0.4 |
| performance_schema | 0.0 |
| photogulp | 193.3 |
| photogulp_webalbum | 0.5 |
| phplistdb | 33.4 |
| pixellone_artistika | 7.9 |
| pixellone_enter | 0.4 |
| rc-bazar_oc | 4.8 |
| wordpress_9 | 1.1 |
+--------------------------------+---------------+
```

Visualizzare la struttura di un database mysql da php

Se avete bisogno di "stampare" la struttura di un database mysql esistente e non volete ricorrere a soluzioni come mysql workbench o similari, potete utilizzare un comodissimo script php di David Walsh.

Si tratta di un file .php, lo copiate sotto la DocumentRoot del vostro webserver, lo editate con db_name, user e password e lo aprite da web. Il risultato sarà del tipo:

admin_roles

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI		auto_increment
nome	varchar(255)	NO			

admin_user

Field	Type	Null	Key	Default	Extra
iduser	int(11)	NO	PRI		auto_increment
username	varchar(255)	NO			
password	varchar(255)	NO			
salt	varchar(255)	NO			
nome	varchar(50)	NO			
cognome	varchar(50)	NO			

Lo script lo potete consultare in questo post del blog di David Walsh oppure scaricarlo da questa pagina su Mr.Webmaster

---

MySQL UDF Perl Regular Expression

Nel realizzare nuovi scraper per g4play.it Emanuele si è reso conto che la nostra istanza MySQL non supporta le espressioni regolari, a lui non servono solo query di ricerca ma manipolazioni di dati complesse. Con estrema semplicità mi chiede di installare la libreria lib_mysqludf_preg, non è complicato, ma neanche così banale. Iniziamo subito con l'installazione dei pacchetti che ci serviranno:

```
[root@mysqllbit lib_mysqludf_preg]# yum install pcre pcre-devel
[root@mysqllbit lib_mysqludf_preg]# yum install make gcc gcc-c++
[root@mysqllbit lib_mysqludf_preg]# yum install mysql-devel
```

Questo per evitare tutti gli errori relativi al compilatore, a pcre e mysql. Scarichiamo il paccheto da GitHub:

```
[root@mysqllbit lib_mysqludf_preg]# wget
https://github.com/mysqludf/lib_mysqludf_preg/archive/testing.zip
```

```
[root@mysqllbit lib_mysqludf_preg]# unzip testing.zip
```

ora lanciamo il configuratore

```
[root@mysqllbit lib_mysqludf_preg]# ./configure
```

ci siamo risparmiati gli errori avendo installato preventivamente i pacchetti, l'unico messaggio a video con la parola ERROR è

```
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using
password: NO)
```

Possiamo rilassarci, avendo settato la password di root è normale che non riesca ad accedere. Ora installiamo:

```
[root@mysqllbit lib_mysqludf_preg]# make
[root@mysqllbit lib_mysqludf_preg]# make install
[root@mysqllbit lib_mysqludf_preg]# make installdb

ERROR 1548 (HY000) at line 5: Cannot load from mysql.proc. The table is
probably corrupted
make: *** [uninstalldb] Error 1
```

Sull'ultimo passaggio ho ricevuto errore di tabella corrotta, per questo ho dovuto prima "sistemare" le tabelle MySQL e poi rilanciare il make installdb

```
[root@mysqllbit lib_mysqludf_preg]# make installdb
/usr/bin/mysql -p <./uninstalldb.sql
Enter password:
cat installdb.sql | sed 's/\.so/.dll/g' >installdb_win.sql
if test -f .libs/lib_mysqludf_preg.dll; then  \
                /usr/bin/mysql -p <./installdb_win.sql; \
        else \
                /usr/bin/mysql -p <./installdb.sql;\
        fi
Enter password:
[root@mysqllbit lib_mysqludf_preg]# make test
cd test; make test
make[1]: Entering directory `/usr/local/lib/lib_mysqludf_preg/test'
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
```

```
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-fi...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
make[1]: Leaving directory `/usr/local/lib/lib_mysqludf_preg/test'
```

Finito, ora Emanuele potrà usare le espressioni regolari per manipolare i dati di g4play.it.
