

Stefano Bortolato



Web server e siti web. Manuale pratico in stile How To

Manuale 1

2 giugno 2024

La scienza senza Dio è tecnologia.

La scienza con Dio è carità.



Web server e siti web.

Manuale pratico in stile How To 1

Una guida prati per creare web server e siti web.

Questa primo manuale guida attraverso i fondamentali: l'installazione di un web server LAMP, la sua messa in funzione e la pubblicazione di un CMS fino alla creazione del certificato per l'HTTPS con Let's Encrypt.

Le parti ti teoria, oltre a offrire i primi elementi necessari di teoria, offrono una visione più ampia sulle altre tecnologie e sulle possibili configurazioni.

Una guida indispensabile per chi è agli inizi, per chi desidera fare da se e per chi intende imparare seriamente queste tecnologie.

Sommario

1. Premessa.....	2
2. Installare il webserver.....	8
3. Installare PHP.....	11
4. Installare MariaDB.....	14
5. Installiamo un CMS.....	25
6. Backup.....	38
7. Consultare-Analizzare i log.....	56
8. Provider.....	60
9. Certificati SSL.....	67
10. Compendio.....	73
11. Conclusione.....	81
12. Indice.....	82

1. Premessa

Il mondo dei web server e dei siti web è un'avventura affascinante e presente un po' in tutte le nostre esperienze: pensiamo a quanti servizi web usiamo ogni giorno.

In questo manuale iniziamo a esplorare l'allestimento di un server sito con l'obiettivo di esplorarlo a tutto campo.

Iniziamo proprio dall'installazione fisica del software Apache HTTP Server, il più noto e usato web server del mondo. A seguire vedremo subito come aggiungere e usare il PHP. Anche in questo caso è il linguaggio di scripting per siti web più usato al mondo. Completiamo l'ambiente con l'installazione del RDBMS MariaDB. Anche questo database è il più usato nelle applicazioni web. Con questo ambiente completo di web hosting (un servizio LAMP in tutta regola) passeremo a eseguire l'installazione dei due CMS più noti: Joomla e WordPress.

Da qui passiamo ad alcuni passi di teoria-pratica: il backup per salvare i nostri siti e le nostre web application in ambiente LAMP. Quindi capiamo dove sono i log generati dal web server per esplorarli e accenneremo alle soluzioni disponibili per analizzarli ed esplorarli.

Da questa esplorazione dei log saltiamo ai provider, per capire cosa sono e come acquistare i loro servizi per poi vedere i certificati HTTPS, capire cosa sono, come si acquistano dai provider e come si generano con "Let's Encrypt" in ambiente Linux.

Chiuderemo con una veloce rassegna delle compliance normative (GDPR, cookies, ecc...), le altre principali tecnologie web, java con gli Application Server. Quindi chiuderemo con un accenno alle tecnologie di casa Microsoft IIS e .NET.

Con questo primo manuale raggiungeremo una competenza base per l'allestimento (o l'acquisto) di un serve LAMP semplice e la pubblicazione di un CMS. Insomma: avremo una infarinatura su tutto quello che serve.

I manuali che seguiranno completeranno i temi qui accennati, dando una trattazione completa per raggiungere livelli di competenza da web master e da sistemista internet.

1.1. Panoramica

Nel procede sull'itinerario sopra indicato ci guiderà un obiettivo preciso:

- avere un ambiente LAMP con le seguenti caratteristiche:
 - OS: Ubuntu Server 22.04 LTS
 - hostname: www.example.com
 - IP: 10.10.10.200
 - web server: Apache 2.4
 - Linguaggio: PHP 8.1
 - Database: MariaDB 10.6
 - Connessione: SSH
- sito web:
 - fare una pagina manualmente
 - fare un'installazione base di PhpMyAdmin
 - fare un'installazione base di Joomla 5
 - fare un'installazione base di WordPress 6.x
- completamenti:
 - imparare a creare un certificato HTTPS con Let's Encrypt
 - capire come si acquista un dominio, un hosting e un server virtuale
 - avere una prima conoscenza delle cache e delle principali tecnologie alternative a LAMP.

1.2. Concetti base

I seguenti concetti li troveremo sviluppati nelle pagine di questa guida. Per un migliore apprendimento è opportuno acquisirne una prima comprensione da subito

- **CMS:** acronimo di Content Management System. Si tratta di una tipologia precisa di software impiegato per creare siti web. Spesso confuso con i sistemi di blog in realtà sono due soluzioni diverse, anche se simili.
- **FQDN:** acronimo di Fully Qualified Domain Name, fa riferimento ai nomi internet correttamente registrati su un DNS.
- **Hosting:** concetto generico che fa riferimento all'ospitare una risorsa per la rete. Potrebbe essere un server, un sito, un servizio...

Web server e siti web. Manuale pratico in stile How To 1

Spesso si intende il “web hosting”, ovvero un ambiente hardware-software in internet idoneo a contenere e far funzionare un sito web.

- **LAMP**: acronimo di Linux, Apache, MySQL o MariaDB, Perl PHP Python, indica una precisa architettura per ospitare siti web normalmente di tipo dinamico. È la tipologia di web hosting più usata.
- **LEMP**: acronimo di Linux, Nginx, MySQL o MariaDB, Perl PHP Python Uguale all'ambiente LAMP usa, come software di web server, Nginx al posto di Apache.
- **(R)DBMS**: famiglia di software che gestiscono database.

Gli DBMS gestiscono database non relazionali. Gli RDMS gestiscono database relazionali.

Programmi di queste famiglie sono: MySQL, MariaDB, PostgreSQL, SQL Server.

- **Server**: concetto generico che indica un computer o un server che fornisce servizi. Al nostro scopo accenniamo ai seguenti server
 - Apache HTTP Server: software di web server. Senza dubbio ancora il più usato ed il noto;
 - Nginx: software di web server. Molto più giovane di Apache vanta di essere molto veloce;
 - Web Server: normalmente si riferisce al computer che esplica questo servizio. Può indicare, però, anche il solo software che offre questo servizio.
- **TCP/IP**: si tratta del protocollo usato da internet per permettere la comunicazione. In realtà si tratta di un insieme di protocolli, ovvero di regole che permettono agli apparati di rete di funzionare automaticamente.

Al nostro scopo vanno menzionati HTTP e HTTPS: due protocolli che normalmente viaggiano sopra il TCP/IP e trasportano le pagine web ed i loro contenuti multimediali.

1.3. Architettura di un sistema Web

Il web è concepito come un sistema server-client:

- il device client inoltra la richiesta di una pagina ad un server;
- il server risponde restituendo quella pagina.

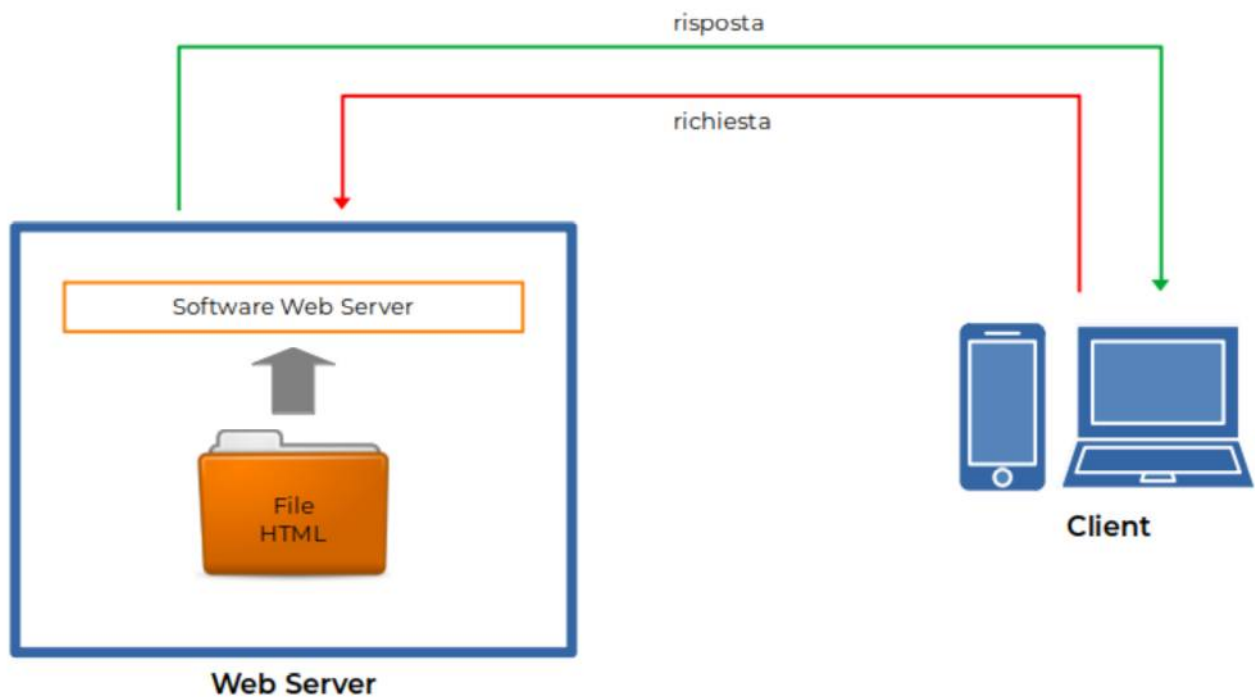


Figura 1: Architettura di principio di un web server semplice

La pagina coincide con un file HTML presente su una specifica directory. Se il file non esiste il server risponde con un errore convenzionale. Ad esempio se non esiste il path o il file richiesto otteniamo un errore 404 (la grafica varia molto a seconda delle impostazioni fatte dal web designer).

Web server e siti web. Manuale pratico in stile How To 1



Figura 2: Schermate di esempio per l'errore 404

Questa architettura di principio ha un limite molto importante: se usiamo dei file HTML per creare le pagine significa che queste sono statiche. Pertanto non è possibile creare form, interrogare un database, generare pagine dinamiche che restituiscono le previsioni del tempo, né creare web TV o radio TV.

Questi limiti si superano, restando sempre nell'architettura di principio, mettendo un linguaggio a monte del webserver. In questo modo il web server passa la pagina richiesta al linguaggio, il quale elabora quanto riceve e restituisce una risposta sotto forma di pagina HTML.

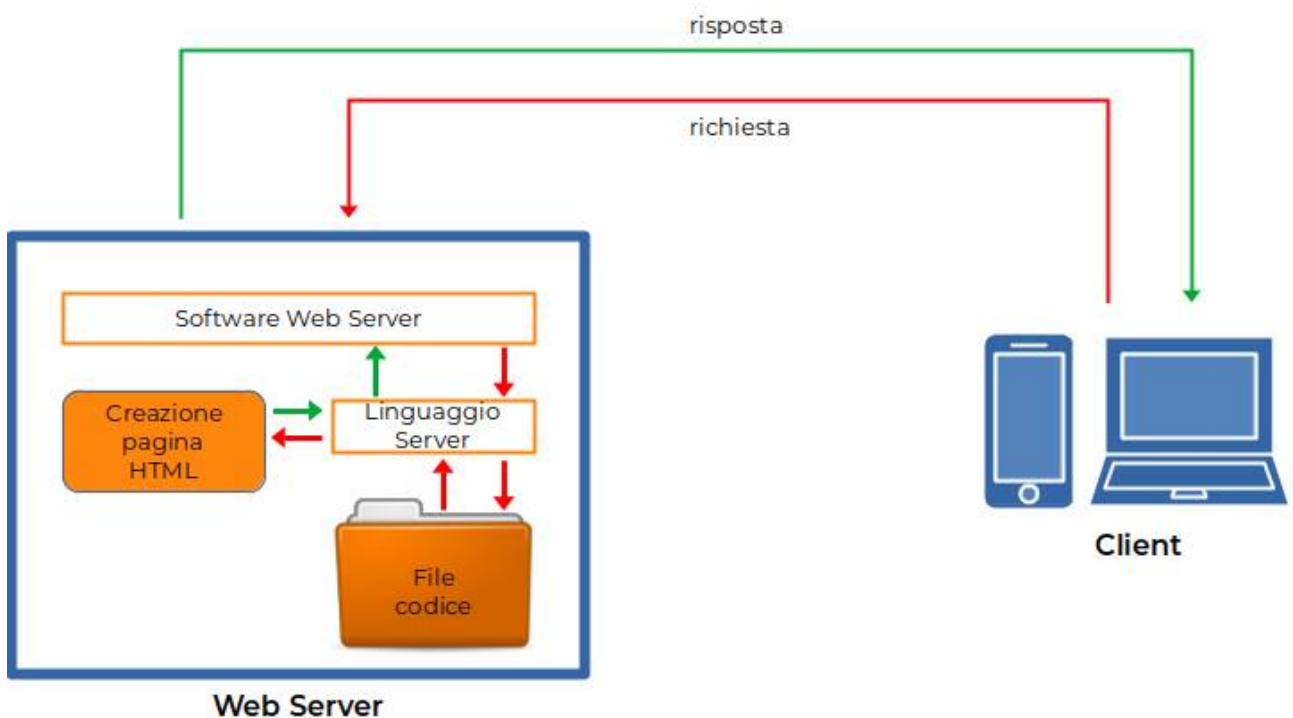


Figura 3: Architettura di principio di un web server con un linguaggio

In questa configurazione la soluzione più popolare del web è l'abbinata PHP con MySQL o MariaDB in ambiente Linux, nota anche come LAMP.

1.4. Prepariamo il server

I capitoli che seguono ipotizzano che operiamo su un server di tipo Ubuntu Server 22.04 LTS (cf.: <https://www.ubuntu-it.org>).

Non è scopo di questo manuale formare all'installazione e a una corretta configurazione. Per questo obiettivo si rimanda alle guide online o alla fruizione di un corso ad hoc.

Le indicazioni e le istruzioni che seguono mirano a dotare ciascuno dell'ambiente di riferimento usato dal capitolo 2 in avanti.

Per chi desidera approfondire l'installazione e la parte sistemistica di un web server Ubuntu può far riferimento alle guide:

- Installazione
<https://wiki.ubuntu-it.org/Installazione>
- Installazione > Installare Ubuntu > Nuovo Installer
<https://wiki.ubuntu-it.org/Installazione/InstallareUbuntu/NuovoInstaller>
- Ubuntu Server how-to guides
<https://ubuntu.com/server/docs/how-to>
- How to Install Ubuntu 24.04 (Noble Numbat) Minimal Server
<https://www.howtoforge.com/tutorial/ubuntu-24-04-minimal-server/>

Per avere un server atto allo scopo didattico di queste pagine possiamo scegliere tra:

- impiegare un vecchio computer dismesso;
- creare una VM (Macchina Virtuale) sul nostro computer tramite un hypervisor come
 - VirtualBox, <https://www.virtualbox.org>
 - VMware Workstation,
<https://www.vmware.com/products/desktop-hypervisor.html>
 - Parallels, <https://www.parallels.com>
- acquistare da un provider un server virtuale.

L'ultima ipotesi, per chi ha un primo approccio a questo mondo, può essere eccessiva anche se parliamo assolutamente di piccoli costi.

Un VM è una via percorribile per tutti, anche totalmente senza costi, gestibile da tutti gli hardware recenti, ma comporta un po' di complicazioni per l'utente.

Per chi recupera un vecchio computer (ma non troppo vecchio) è una nobile azione anche di riciclaggio.

L'installazione, in linea di principio, è semplice:

Web server e siti web. Manuale pratico in stile How To 1

- si scarica il file ISO di Ubuntu
- si scrive il file su un DVD o un pendrive.

Nel caso realizziamo un VM basta rendere disponibile il file all'hypervisor;

- si avvia la macchina fisica dal supporto creato dal file ISO.

Nel caso di macchina virtuale basta farla avviare dal file ISO;

- dopo l'avvio c'è una procedura guida che provvede prima a chiedere le informazioni necessarie, poi a installare Ubuntu.

Durante la prima fase di raccolta delle informazioni fondamentali è importante impostare l'IP 10.10.10.200, impostare la username `webadmin` e documentare subito la password impostata e selezionare l'installazione del server OpenSSH;

- al termine dell'installazione verrà chiesto il riavvio.

Finita la fase di installazione è importante fare subito un'operazione di aggiornamento:

- colleghiamoci al server appena creato o dalla console o via SSH;
- autenticiamoci con l'utente impostato durante l'installazione;
- dal terminale diamo il seguente comando di aggiornamento

```
sudo apt update
```

NB: il comando `sudo` richiede una password. Quella da inserire è la password dell'utente `webadmin`

- diamo il comando per applicare gli aggiornamenti disponibili

```
sudo apt full-upgrade
```

Simo pronti per procedere.

2. Installare il webserver

Partiamo lavorando sul server appena creato (un Ubuntu Server 22.04 LTS).

I parametri principali per operare sul server sono:

IP: 10.10.10.200

FQDN: `www.example.com`

User: `webadmin`

L'utente `webadmin` è amministratore del server.

Procediamo installando Apache HTTP Server e testeremo da un client una pagina creata appositamente.

Lavoriamo dal terminale testuale ipotizzando di collegarci tramite la rete via SSH. Se operiamo dalla console possiamo usare la stessa procedura omettendo il comando di connessione SSH.

Web server e siti web. Manuale pratico in stile How To 1

Infine ipotizziamo che il computer da cui lavoriamo è una workstation Linux (quella che usiamo durante la stesura di queste pagine). Se preferiamo Mac il terminale offre di default il servizio ssh. Con Windows dovremo installare un client SSH (come PuTTY) e lanciarlo per poi operare.

2.1. Installare Apache HTTP Server

1. Dalla nostra workstation apriamo un terminale e diamo il comando di connessione al webserver remoto:

```
ssh weadmin@www.example.com
```

2. appena stabilita la connessione viene chiesta la password. La digitazione è sempre cieca, cioè non vediamo nulla sullo schermo, né quello che digitiamo, né caratteri jolly come l'asterisco;
3. ora siamo finalmente nel server remoto tramite il terminale. Diamo il comando installazione di Apache HTTP Server:

```
sudo apt install apache2
```

A questo punto il software di webserver è installato e funzionante.

Nota: con questa installazione e configurazione è funzionante una versione base di webserver. Inoltre viene automaticamente avviato e configurato come servizio sempre attivo.

2.2. Testiamo Apache HTTP Server

Testiamo il funzionamento e poi creiamo una pagina test con il nome `test.html`.

1. Dalla workstation client apriamo un web browser, come FireFox o Edge, e accediamo all'indirizzo `http://www.example.com`.

Comparirà la seguente pagina!

Web server e siti web. Manuale pratico in stile How To 1



Figura 4: Home default dopo l'installazione del web server Apache in Ubuntu

2. Creiamo ora la pagina `test.html`.

Dal terminale sul server diamo il seguente comando

```
sudo nano /var/www/html/test.html
```

3. inseriamo il seguente codice e usciamo salvando

```
<!doctype html>
<html>
  <body>
    <h1>Pagina TEST</h1>
    <p>La mia prima pagina HTML da <b>Apache HTTP Server</b></p>
  </body>
</html>
```

4. dalla workstation client apriamo un web browser, come FireFox o Edge, e accediamo all'indirizzo `http://www.example.com/test.html`.

Comparirà la seguente pagina.



Figura 5: Visualizzazione della pagina index test

2.3. Prima conclusione per Apache

Installare e far funzionare un webserver è una cosa semplice e che richiede veramente pochi secondi.

Con le operazioni sopra descritte, però, non abbiamo nessuna sicurezza: il webserver funziona correttamente, ma non è sicuro.

Inoltre è privo di ogni funzione: non si possono fare form funzionanti, interrogare database, ecc...

Per poter avere tutte queste funzioni dobbiamo aggiungere un paio di strati software. In particolare: un linguaggio e un database.

Iniziamo installando PHP, il linguaggio più noto e più usato nelle web application.

Procederemo, poi, installando il database MariaDB.

3. Installare PHP

Una volta installato Apache l'aggiunta di PHP richiede la stessa procedura.

Tenete presente che PHP è un ecosistema ampio, con molte componenti, librerie, framework, acceleratori... e con altrettante configurazioni.

Tutta questa parte aggiuntiva la salteremo intenzionalmente.

3.1. Installiamo PHP

Sempre operando dal nostro terminale sul server `www.example.com` diamo il seguente comando:

```
sudo apt install php8.1 php8.1-common php8.1-gd php8.1-mysql php8.1-imap php8.1-  
cli php8.1-cgi php-pear php8.1-curl php8.1-intl php8.1-pspell php8.1-sqlite3  
php8.1-tidy php8.1-xmlrpc php8.1-xsl php8.1-zip php8.1-mbstring php-soap php8.1-  
soap php8.1-openssl php8.1-memcache php8.1-apcu php8.1-imagick libapache2-mod-  
php8.1
```

al termine dell'installazione riavviamo Apache per rendere esecutivo PHP:

```
sudo systemctl restart apache2
```

3.2. Verifichiamo PHP

Verifichiamo il PHP appena installato e caricato:

1. creiamo ora la pagina `info.php`.

Dal terminal sul server diamo il seguente comando

```
sudo nano /var/www/html/info.php
```

2. inseriamo il seguente codice e usciamo salvando

```
<?php  
phpinfo();  
?>
```

3. dalla workstation client apriamo un web browser, come FireFox o Edge, e accediamo all'indirizzo `http://www.example.com/info.php`.

Comparirà la seguente pagina!

PHP Version 8.1.2-1ubuntu2.17	
System	Linux examplecom 5.15.0-107-generic #117-Ubuntu SMP Fri Apr 26 12:26:49 UTC 2024 x86_64
Build Date	May 1 2024 10:10:07
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/apache2
Loaded Configuration File	/etc/php/8.1/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/apache2/conf.d
Additional .ini files parsed	/etc/php/8.1/apache2/conf.d/10-mysqld.ini, /etc/php/8.1/apache2/conf.d/10-opcache.ini, /etc/php/8.1/apache2/conf.d/10-pdo.ini, /etc/php/8.1/apache2/conf.d/15-xml.ini, /etc/php/8.1/apache2/conf.d/20-apcu.ini, /etc/php/8.1/apache2/conf.d/20-calendar.ini, /etc/php/8.1/apache2/conf.d/20-ctype.ini, /etc/php/8.1/apache2/conf.d/20-curl.ini, /etc/php/8.1/apache2/conf.d/20-dom.ini, /etc/php/8.1/apache2/conf.d/20-exif.ini, /etc/php/8.1/apache2/conf.d/20-fileinfo.ini, /etc/php/8.1/apache2/conf.d/20-ftp.ini, /etc/php/8.1/apache2/conf.d/20-gd.ini, /etc/php/8.1/apache2/conf.d/20-gettext.ini, /etc/php/8.1/apache2/conf.d/20-iconv.ini, /etc/php/8.1/apache2/conf.d/20-imagick.ini, /etc/php/8.1/apache2/conf.d/20-imap.ini, /etc/php/8.1/apache2/conf.d/20-intl.ini, /etc/php/8.1/apache2/conf.d/20-mbstring.ini, /etc/php/8.1/apache2/conf.d/20-memcache.ini, /etc/php/8.1/apache2/conf.d/20-mysqli.ini, /etc/php/8.1/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.1/apache2/conf.d/20-pdo_sqlite.ini, /etc/php/8.1/apache2/conf.d/20-pear.ini, /etc/php/8.1/apache2/conf.d/20-posix.ini, /etc/php/8.1/apache2/conf.d/20-readline.ini, /etc/php/8.1/apache2/conf.d/20-redis.ini, /etc/php/8.1/apache2/conf.d/20-sockets.ini, /etc/php/8.1/apache2/conf.d/20-ssh2.ini, /etc/php/8.1/apache2/conf.d/20-sysvshm.ini, /etc/php/8.1/apache2/conf.d/20-tokenizer.ini, /etc/php/8.1/apache2/conf.d/20-xmlreader.ini, /etc/php/8.1/apache2/conf.d/20-xmlwriter.ini, /etc/php/8.1/apache2/conf.d/20-xsl.ini, /etc/php/8.1/apache2/conf.d/20-zip.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	AP20210902.NTS
PHP Extension Build	AP20210902.NTS
Debug Build	no
Thread Safety	disabled

Figura 6: Pagina auto-generata da PHP

3.3. Prima conclusione per PHP

In questa fase abbiamo raggiunto con totale successo il nostro obiettivo: abbiamo un webserver con PHP installato e funzionante: complementi!

Sicuramente vedremo delle performance eccellenti. In realtà abbiamo installato la soluzione più lenta per eseguire PHP. Questo diventerà rapidamente evidente all'accesso contemporaneo di più navigatori e alla messa in produzione di applicazioni PHP (giusto per capirci: WordPress, Joomla, ecc...).

Vale la pena fermarci un paio di minuti per spiegare potenzialità, limiti e soluzioni.

- **PHP:** si tratta di un linguaggio di scripting interpretato. Malgrado lo sviluppo lo abbia portato ai livelli dei linguaggi moderni resta invariata la sua natura originale: gli vengono passate delle istruzioni, le compila al volo, esegue le istruzioni macchina, vengono passate al web server righe HTML ed il server le passa all'utente che le ha richieste.

Questa architettura è semplice, funziona, ma obbliga ad ogni richiesta di ricreare il codice macchina e il codice HTML;

- **vantaggi e limiti:** il vantaggio più rilevante è che non serve compilazione. PHP ri-crea (=interpreta) tutto al volo. Quindi ogni scripting funziona in tempo reale. Aspetto particolarmente apprezzabile in caso di fixing, bug, ecc... Contemporaneamente è anche il limite più significativo perché richiede molto calcolo per ogni singola richiesta;
- **soluzioni:** la prima soluzione è insita alla programmazione stessa. Le applicazioni PHP vanno progettate con sistemi di cache interne che creano solo una volta la parte fissa delle pagine web e ricalcolano ogni volta solo le parti dinamiche.

Questo meccanismo viene potenziato da componenti aggiuntive di cache per PHP come APCu, Memcached, ecc...

Un secondo importante acceleratore viene dall'uso di moduli di interconnessione tra il web server e PHP che elaborino parallelamente più richieste. Quindi, in alternativa al tradizionale mod_php (che abbiamo usato sopra) si impiegano Fast-CGI, PHP-FPM, ecc...

In conclusione ricordiamo il limite, per applicazioni piccole è trascurabile, per obiettivi più importanti abbiamo di serie diversi strumenti per mitigare e controllare il problema. Per obiettivi più ambiziosi abbiamo soluzioni che comportano un po' di complessità e costi.

Per servizi critici e di alto livello vale la pena guardare ad altre tecnologie, meno note, più complesse, ma pagano alla grande tutti i *rompicapo* di progettazione, installazione e realizzazione dei servizi.

4. Installare MariaDB

Passiamo a installare un database di backend.

Tradizionalmente MySQL era il database dedicato alle applicazioni e siti costruiti con PHP e Apache. Quando la Oracle comprò la Sun, modificando la licenza d'uso di MySQL (ed anche il suo sviluppo) nacque lo spin-off MariaDB. Da alcuni anni le distribuzioni Linux hanno di serie sia MySQL, sia MariaDB.

Senza entrare in spiegazioni (noiose) qui di seguito descriviamo l'installazione e l'uso di MariaDB. Successivamente aggiungeremo nel web server PhpMyAdmin, un programma per gestire il database direttamente dal web. Poi vedremo l'uso di DBeaver, un'app desktop per gestire MariaDB e ogni tipo di database.

Non vedremo nulla, invece, circa l'SQL, il linguaggio per parlare con MariaDB e circa la gestione di MariaDB anche se si tratta di conoscenze che è opportuno acquisire per gestire adeguatamente un web server. Esulano dagli scopi di questa guida.

4.1. Installiamo MariaDB

L'installazione viene fatta in due passi:

- installazione di MariaDB;
- securizzazione di MariaDB.

All'installazione MariaDB è aperto a tutti e non ha password. Per questo motivo è necessario securizzarlo.

Sempre operando dal nostro terminale sul server `www.example.com` diamo il seguente comando:

```
sudo apt install mariadb-server
```

l'istruzione installa sia la componente server, sia la componente client.

Al termine dell'installazione il database viene automaticamente avviato e impostato come servizio disponibile ad ogni riavvio.

Procediamo alla securizzazione tramite `mariadb-secure-installation`:

```
sudo mariadb-secure-installation
```

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB  
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!
```

```
In order to log into MariaDB to secure it, we'll need the current  
password for the root user. If you've just installed MariaDB, and  
haven't set the root password yet, you should just press enter here.
```

```
Enter current password for root (enter for none): RETURN  
OK, successfully used password, moving on...
```


Web server e siti web. Manuale pratico in stile How To 1

Setting the root password or using the `unix_socket` ensures that nobody can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

```
Switch to unix_socket authentication [Y/n] RETURN
Enabled successfully!
Reloading privilege tables..
... Success!
```

You already have your root account protected, so you can safely answer 'n'.

```
Change the root password? [Y/n] RETURN
New password: MiaPassword
Re-enter new password: MiaPassword
Password updated successfully!
Reloading privilege tables..
... Success!
```

By default, a MariaDB installation has an anonymous user, allowing anyone to log into MariaDB without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

```
Remove anonymous users? [Y/n] RETURN
... Success!
```

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

```
Disallow root login remotely? [Y/n] RETURN
... Success!
```

By default, MariaDB comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

```
Remove test database and access to it? [Y/n] RETURN
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
```

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

Web server e siti web. Manuale pratico in stile How To 1

```
Reload privilege tables now? [Y/n] RETURN
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Evidenziate le azioni:

- **RETURN**: premere il tasto invio della tastiera;
- **MiaPassword**: la password per l'utente `root` di MariaDB. Ovviamente mettete la vostra password sicura.

Nota 1: per usare l'utente `root` di MariaDB è necessaria una password.

Nota 2: l'utente di sistema `root` dal terminale è automaticamente trustato e non viene chiesta la password.

4.2. Verifichiamo MariaDB

Verifichiamo il nostro server con una richiesta di versione e una query base.

Vediamo la versione di MariaDB:

```
mariadb -V
mariadb Ver 10.6.16-MariaDB-0ubuntu0.22.04.1 for debian-linux-gnu on x86_64
(Ubuntu 22.04)
```

Accediamo alla command-line `mariadb`, chiediamo l'elenco dei database e usciamo:

```
sudo mariadb -h localhost
MariaDB [(none)]> show databases;
+-----+
| Database          |
+-----+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+-----+
4 rows in set (0.001 sec)

MariaDB [(none)]> \q
Bye
```

4.3. Gestiamo MariaDB con PhpMyAdmin

PhpMyAdmin è presente nei repo default di Ubuntu e lo possiamo installare direttamente con `apt`. Procederemo, invece, installandolo manualmente: operazione più laboriosa, ma utile allo scopo didattico. Sarà la metodologia che useremo anche per installare i CMS.

- Sempre operando dal nostro terminale sul server `www.example.com` installiamo le utility `wget` e `unzip`

```
sudo apt install wget unzip
```

- creiamo una directory di comodo e accediamoci

```
mkdir Temp && cd Temp
```

- scarichiamo PhpMyAdmin

```
wget https://files.phpmyadmin.net/phpMyAdmin/5.2.1/phpMyAdmin-5.2.1-all-languages.zip
```

- estraiamo il pacchetto

```
unzip phpMyAdmin-5.2.1-all-languages.zip
```

- spostiamo i file di PhpMyAdmin nella web root e rinominiamo la directory da `phpMyAdmin-5.2.1-all-languages` a `pma`

```
sudo mv -r phpMyAdmin-5.2.1-all-languages /var/www/html/pma
```

- creiamo il file di configurazione e lanciamo l'editor

```
sudo cp /var/www/html/pma/config.sample.inc.php /var/www/html/pma/config.inc.php
sudo nano /var/www/html/pma/config.inc.php
```

- inseriamo una password per il meccanismo blowfish: basta inserire una lunga stringa di caratteri casuali come nell'esempio qui di seguito.

Per uscire dall'editor cliccare CTRL+X, confermare con Y e premere invio

```
[ ... ]
$config['blowfish_secret'] = 'ihpeblòfgourhg9073290563phjasncòacdpo3ur94153476091kjhtra2'; /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */
[ ... ]
```

dalla workstation client apriamo un web browser, come FireFox o Edge, e accediamo all'indirizzo `http://www.example.com/pma`.

Comparirà la seguente pagina

Web server e siti web. Manuale pratico in stile How To 1



Figura 7: Login di PhpMyAdmin

Inseriamo le nostre credenziali (utente root e la password impostata prima con `mariadb-secure-installation`)



Figura 8: Login di PhpMyAdmin: inserimento delle credenziali

comparirà la pagina principale: MariaDB è funzionante, ma ancora vuoto.

Web server e siti web. Manuale pratico in stile How To 1

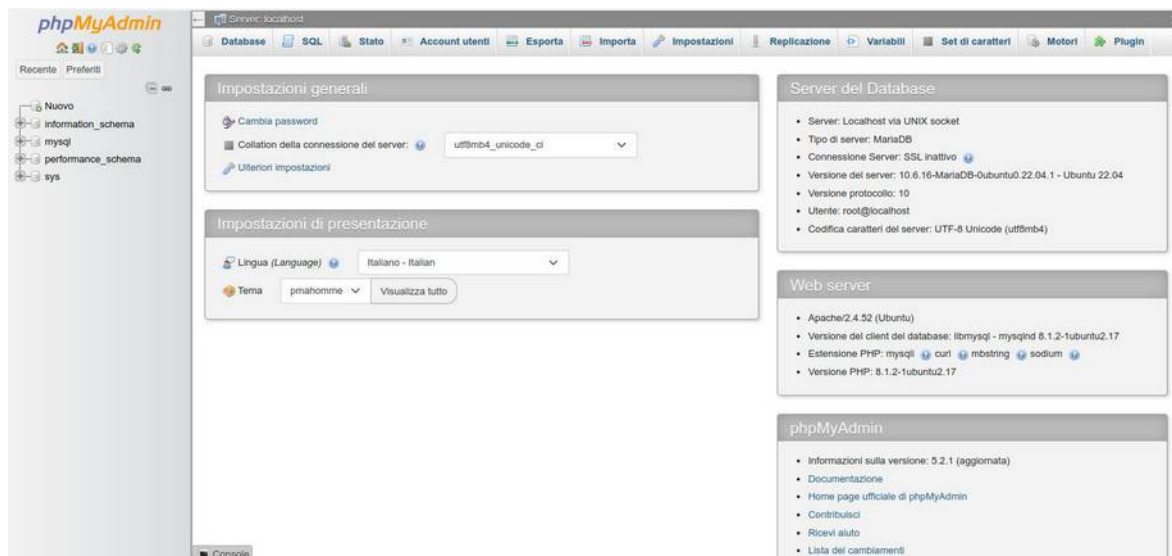


Figura 9: Home di PhpMyAdmin

Procediamo creando un utente remoto per la gestione totale di MariaDB che useremo dopo con DBeaver.

Nota: a titolo didattico creeremo un utente che può connettersi da ovunque: non è una buona politica di sicurezza! Va bene solo a scopo didattico e negli ambienti di sviluppo!

- Clicchiamo sul bottone “Account utenti”;
- clicchiamo su “Aggiungi account utente”

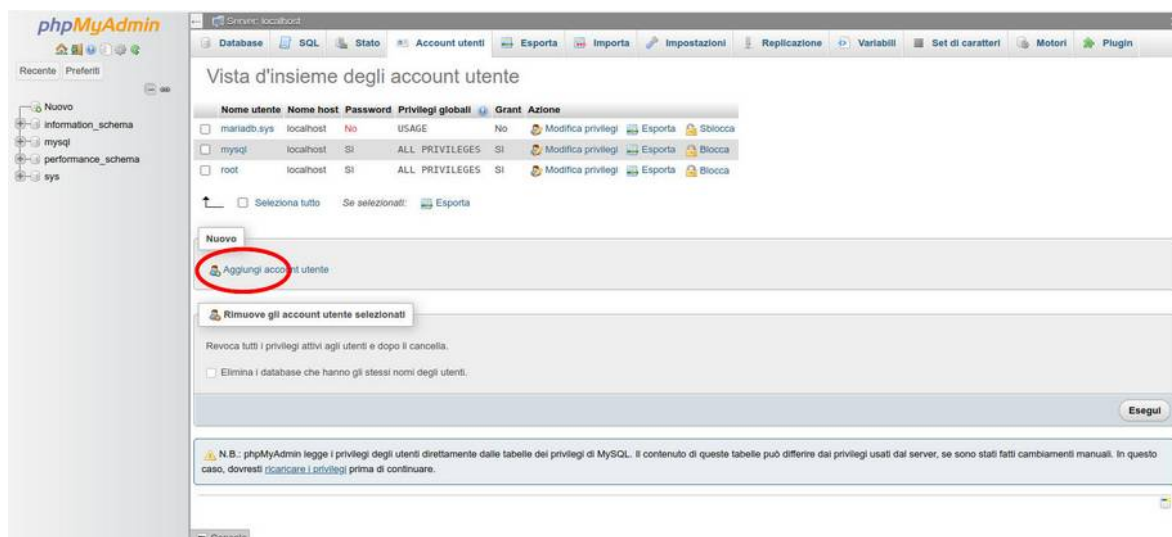


Figura 10: PhpMyAdmin: link per accedere alla creazione di un utente

- nella form che compare impostiamo
 - Nome utente: remoteroot
 - Nome host: %
 - Password: PasswordEsempio
 - Privilegi globali: Seleziona tutto

Web server e siti web. Manuale pratico in stile How To 1

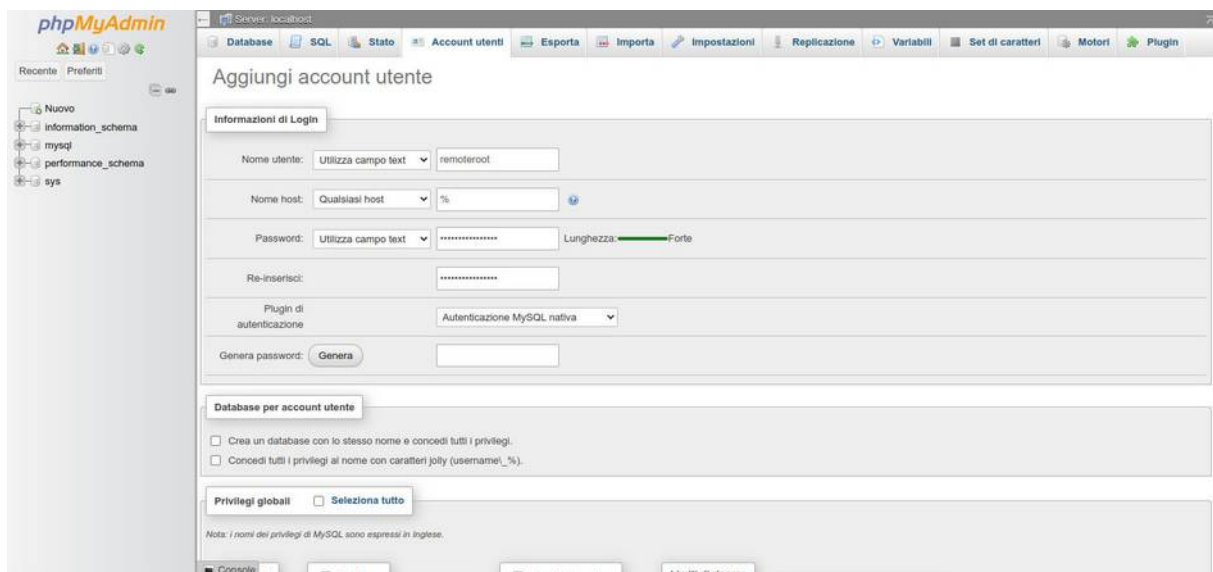


Figura 11: PhpMyAdmin. Creazione di un utente: inserimento dei dati

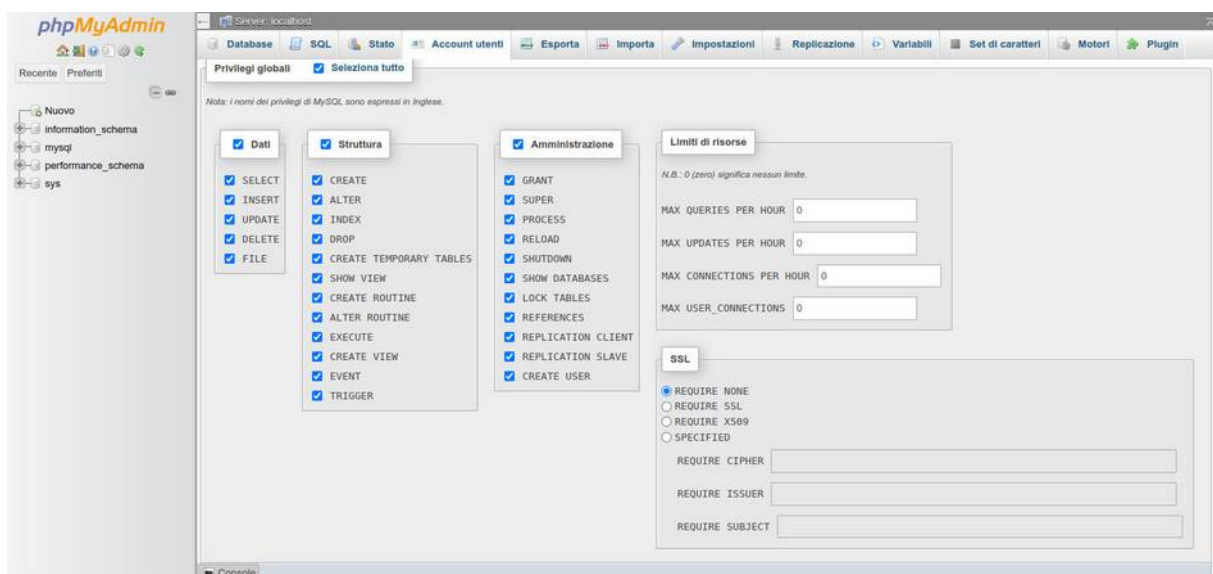


Figura 12: PhpMyAdmin. Creazione di un utente: impostazione dei permessi

4.4. Gestiamo MariaDB da DBeaver

PhpMyAdmin è uno stupendo strumento per gestire i nostri database da un'interfaccia web.

Per diverse ragioni è utile avere un software desktop per gestire i nostri database remoti e/o locali. Useremo DBeaver (<https://dbeaver.io>) una soluzione open source, compatibile con Windows, Linux e Mac e in grado di gestire svariati database tra cui MariaDB e PostgreSQL (di cui parleremo qui di seguito).

Scarichiamo e installiamo DBeaver sulla nostra workstation e lanciamolo. Comparirà una schermata simile a quella qui di seguito

Web server e siti web. Manuale pratico in stile How To 1

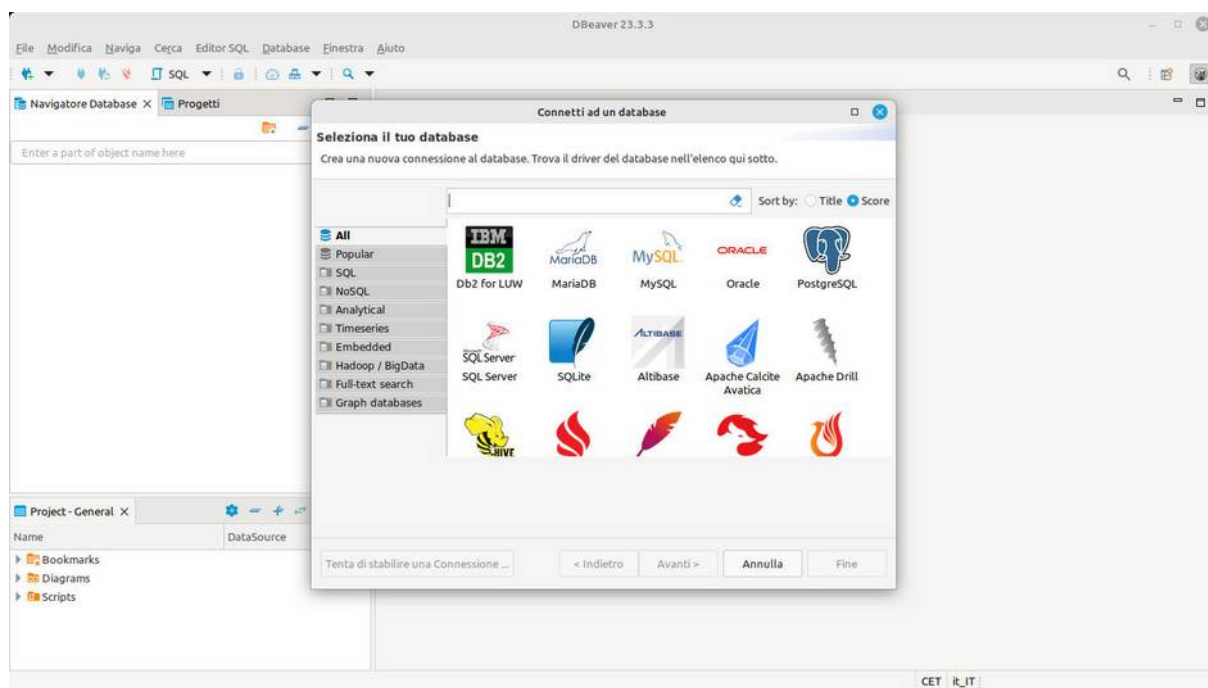


Figura 13: DBeaver: schermata al primo lancio

procediamo creando la prima connessione.

- Clicchiamo sull'icona di MariaDB e poi sul bottone in basso "Avanti";
- compiliamo la form di creazione collegamento come segue

Server HOST: localhost

Porta: 3306

Nome utente: remoteroot

Password: PasswordEsempio

Salva password: selezionato

Web server e siti web. Manuale pratico in stile How To 1

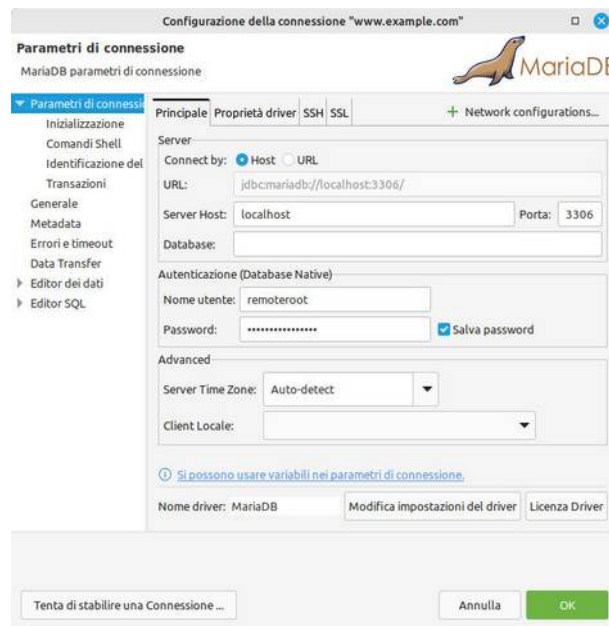


Figura 14: DBeaver: creazione di una connessione. Inserimento dati di accesso a MariaDB

- clicchiamo sul tab SSH e impostiamo i dati per l'accesso remoto ssh usati finora

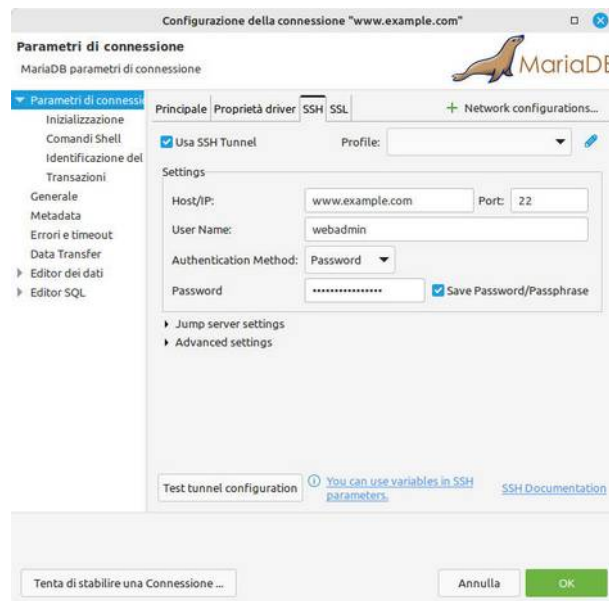


Figura 15: DBeaver: creazione di una connessione. Inserimento dati di accesso al server

- segue una richiesta una tantum per scaricare il driver specifico per creare la connessione al database. Confermare e proseguire

Web server e siti web. Manuale pratico in stile How To 1

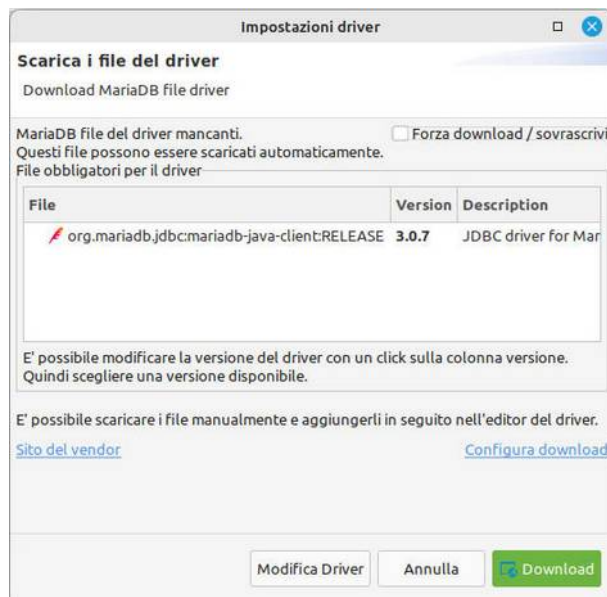


Figura 16: DBeaver: schermata informativa per lo scaricamento del connector per MariaDB

- a questo punto abbiamo disponibile una nuova connessione sulla colonna di sinistra
- doppio click sulla connessione per aprire il collegamento al server

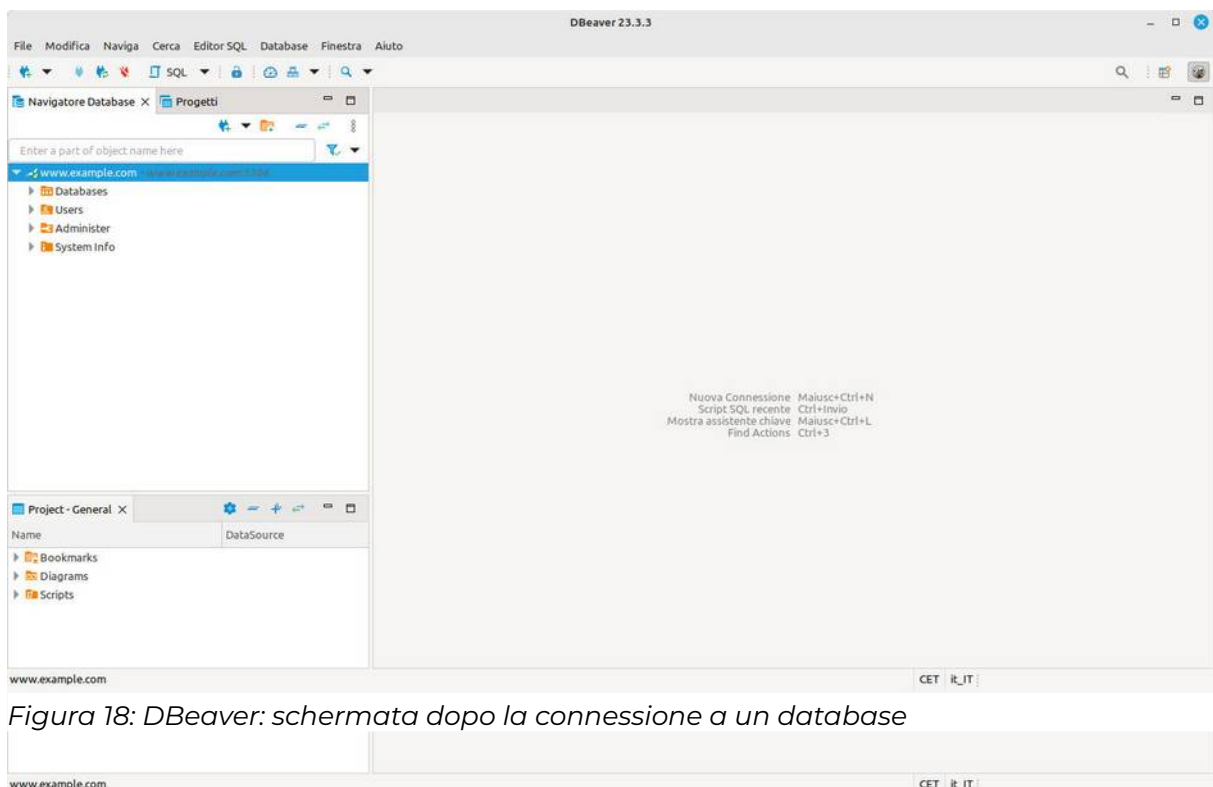


Figura 18: DBeaver: schermata dopo la connessione a un database

Figura 17: DBeaver: collocamento della connessione al database
Connessione riuscita!

A questo punto abbiamo i permessi per fare ogni cosa.

4.5. PostgreSQL

Una piccola digressione circa PostgreSQL, un database alternativo, sempre open source, costruito sul concetto software di oggetto. MySQL e MariaDB, invece, sono costruiti sul concetto di tabelle relazionate.

Tecnicamente è più potente di MariaDB, gestisce funzioni più avanzate, supporta linguaggi aggiuntivi e una più ampia tipologia di dati.

Accanto a queste ed altre caratteristiche specifiche, diverse applicazioni sono costruite su questo DBMS e l'uso si è molto diffuso.

La gestione è molto diversa da MariaDB, ma esistono, oltre a DBeaver, tre interessanti applicativi (web e desktop) che semplificano la gestione: pgAdmin, Admin4 e phpPgAdmin (fork di PhpMyAdmin).

Questo fantastico database va un po' studiato per usarlo, ma dopo una fatica iniziale, paga abbondantemente lo sforzo sostenuto.

Ricordiamolo e conosciamolo anche se non lo useremo.

5. Installiamo un CMS

Abbiamo installato Ubuntu creando il server `www.example.com`. Successivamente abbiamo installato il web server Apache HTTP a cui abbiamo aggiunto lo strato software PHP. Infine abbiamo aggiunto e inizializzato MariaDB e PhpMyAdmin per gestirlo.

Passiamo ora a installare due noti CMS: Joomla e WordPress. Le procedure sono simili e analoghe all'installazione di molti altri prodotti Web.

Passiamo, dunque, alla fase esecutiva. L'obiettivo è il seguente ambiente:

- Joomla:
 - URL: `www.example.com/joomla`
 - Database nome: `joomladb`
 - Database user: `joomlauser`
 - Database password: `joomlapass`
 - Joomla admin user: `jadminuser`
 - Joomla admin password: `jadminpass32FF`
- WordPress:
 - URL: `www.example.com/wordpress`
 - Database nome: `wordpressdb`

Web server e siti web. Manuale pratico in stile How To 1

- Database user: wordpressuser
- Database password: wordresspass
- Wordpress admin user: wpadminuser
- Wordpress admin password: wpadminpass

5.1. Preparazione per Joomla

Prepariamo il database per Joomla:

- colleghiamoci a MariaDB attraverso PhpMyAdmin accedendo all'URL <http://www.example.com/pma>
- autenticiamoci con l'utente `root` o con `remoteroot`
- nella home clicchiamo sul bottone "Database"

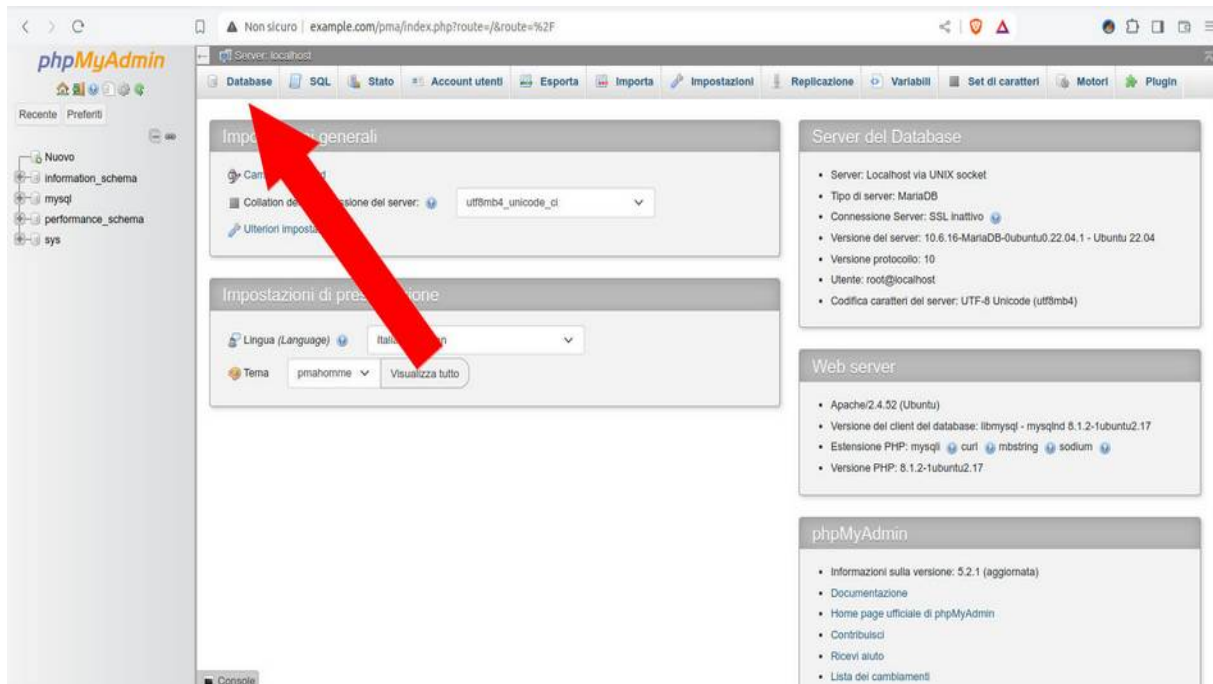


Figura 19: PhpMyAdmin: osizionamento del menu "Database"

- nella form di creazione database scriviamo il nome `joomlaadb` e clicchiamo sul bottone "Crea"

Web server e siti web. Manuale pratico in stile How To 1

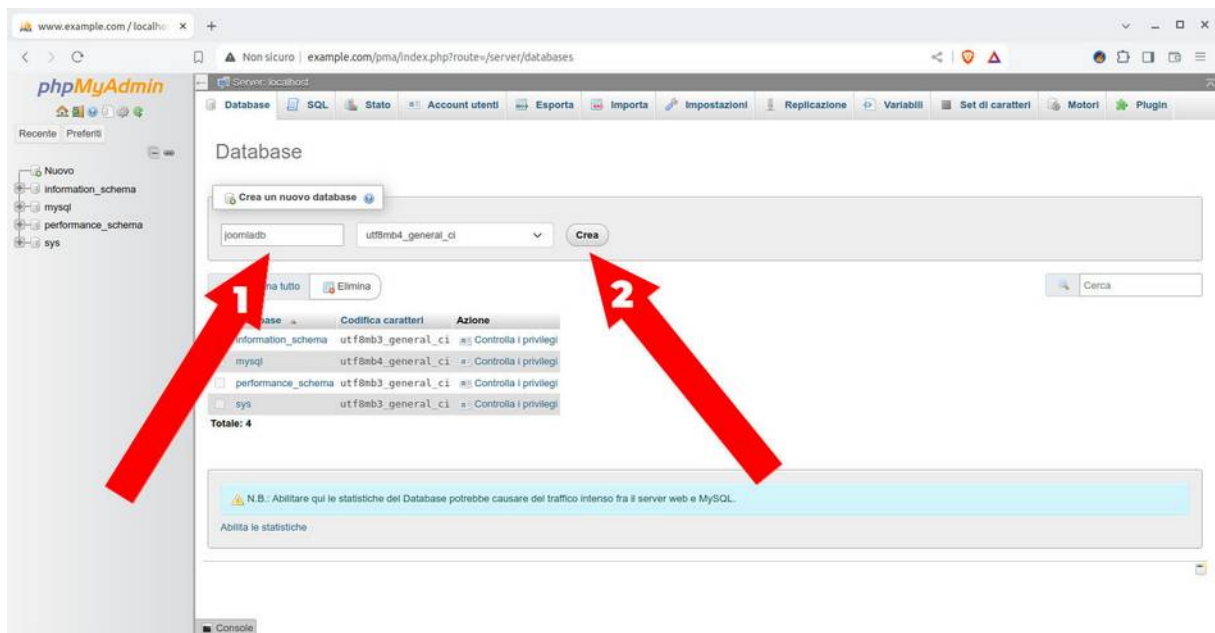


Figura 20: PhpMyAdmin: schermata per creare un database

- nella nuova form che comparirà clicchiamo prima su “Privilegi” e poi su “Aggiungi account utente”

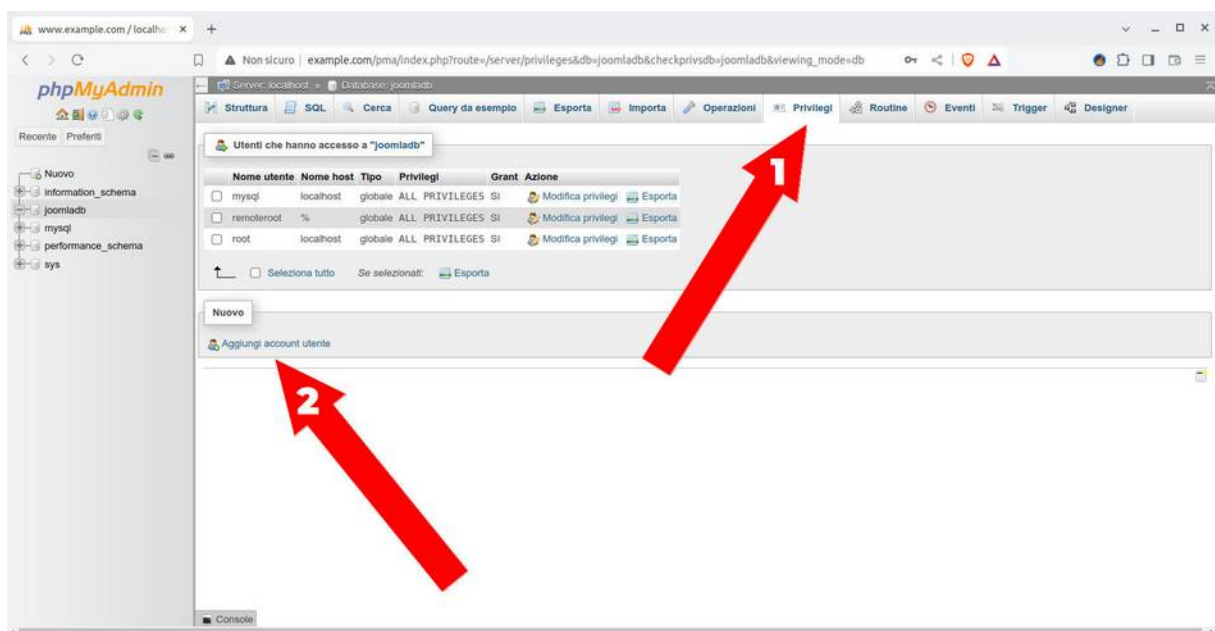


Figura 21: PhpMyAdmin: schermata per creare un utente

- nella form che segue impostiamo
Nome utente: joomlauser
Nome host: localhost
Password: joomlapass

Web server e siti web. Manuale pratico in stile How To 1

e clicchiamo sul bottone a fondo pagina “Esegui”

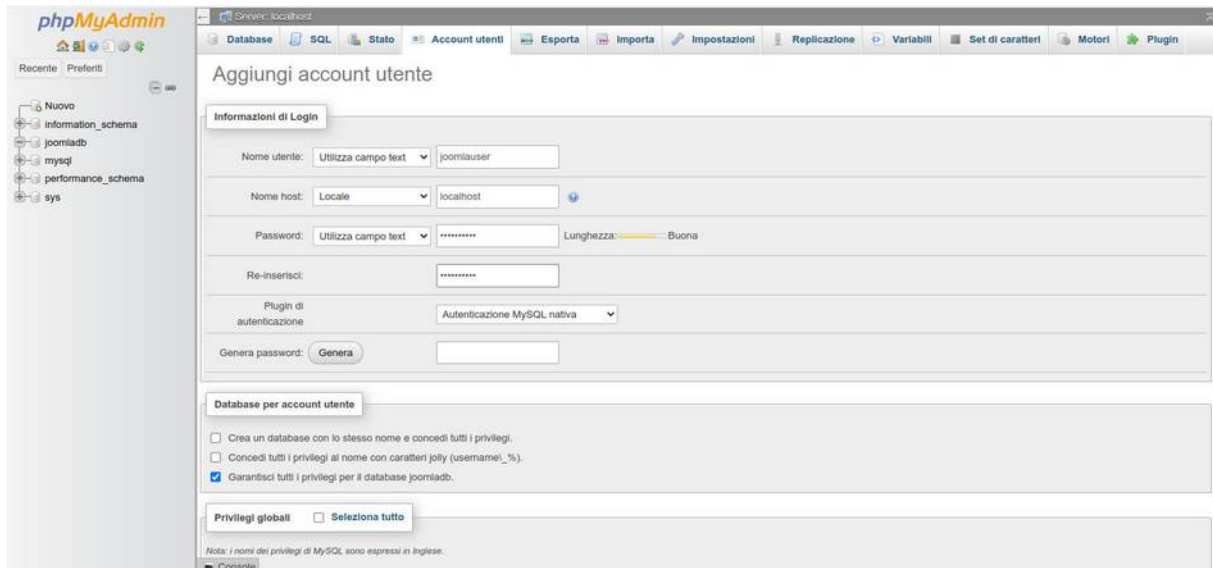


Figura 22: PhpMyAdmin: schermata per creare un account utente

- nella nuova pagina che compare clicchiamo su “Database”

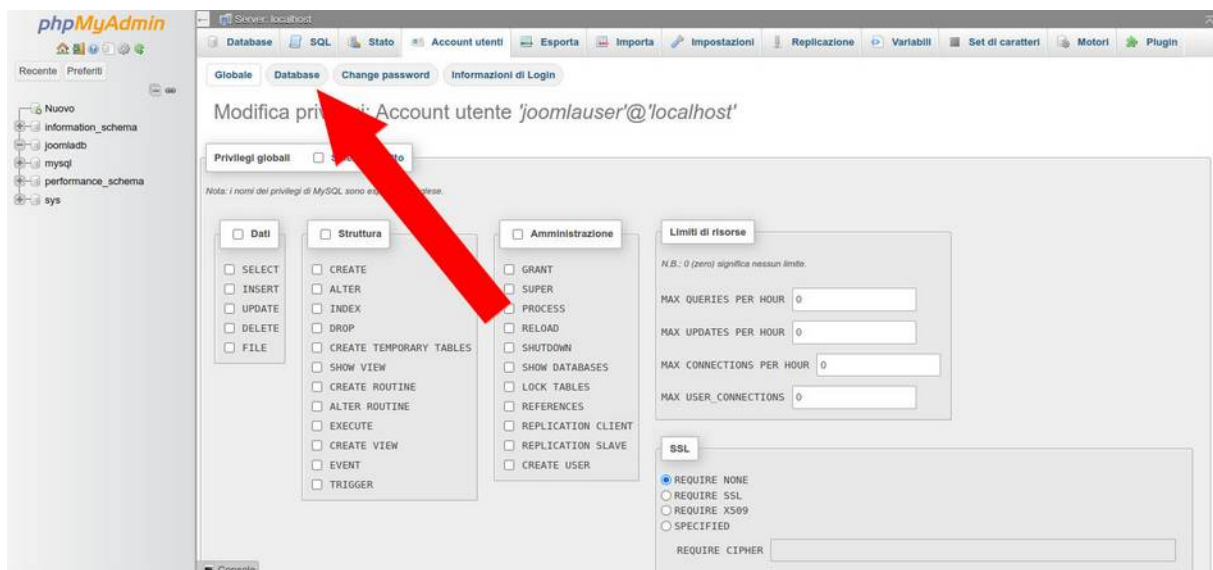


Figura 23: PhpMyAdmin: schermata principale per impostare i privilegi di un account

- nella pagina di selezione del database clicchiamo su “joomlabd” e poi sul bottone esegui

Web server e siti web. Manuale pratico in stile How To 1

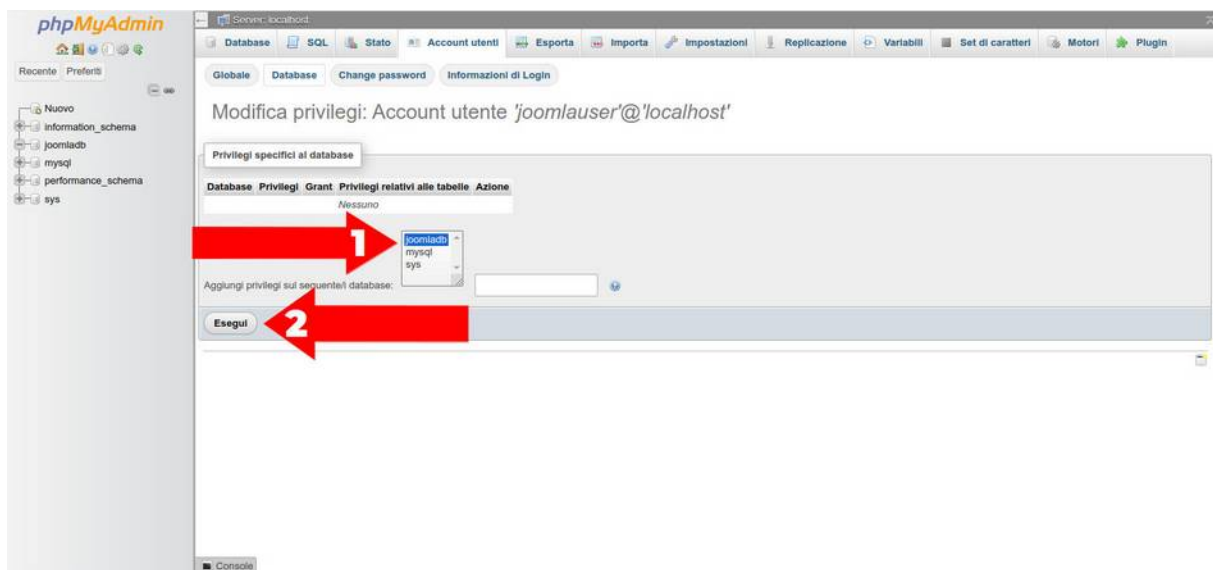


Figura 24: PhpMyAdmin: schermata per selezionare il database da associare ad un account

- selezioniamo tutte le autorizzazioni disponibili e clicchiamo sul bottone “Esegui”

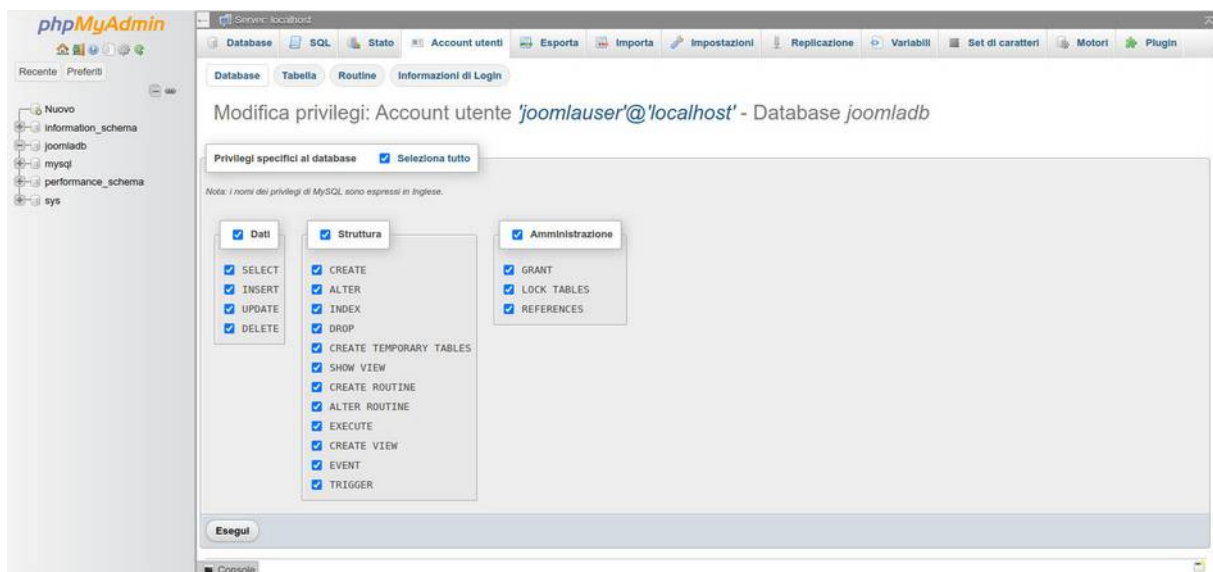


Figura 25: PhpMyAdmin: schermata per i privilegi account-database

A questo punto abbiamo creato un database e un utente con permessi ristretti solo al database “joomlaadb”. Siamo pronti ad eseguire l’installazione vera e propria di Joomla.

5.2. Installare Joomla

Iniziamo operando prima dal terminale e poi completeremo dal web browser dalla nostra workstation.

- Dal terminale spostiamoci nella directory temporanea che abbiamo creato prima

```
cd ~ && cd Temp
```

- creiamo subito la directory “joomla” che useremo in seguito

Web server e siti web. Manuale pratico in stile How To 1

```
mkdir joomla && cd joomla
```

- scarichiamo pacchetto di Joomla 5

```
wget https://downloads.joomla.org/cms/joomla5/5-1-0/Joomla_5-1-0-Stable-Full_Package.zip
```

- estraiamolo ed eliminiamo lo zip che non ci servirà

```
unzip Joomla_5-1-0-Stable-Full_Package.zip  
rm Joomla_5-1-0-Stable-Full_Package.zip
```

- spostiamo il tutto nella web root

```
cd ..  
sudo mv joomla /var/www/html/
```

- e infine sistemiamo permessi e predisponiamo l'attivazione del SEO

```
sudo chown www-data:www-data -R /var/www/html/joomla  
sudo cp -p /var/www/html/joomla/htaccess.txt /var/www/html/joomla/.htaccess
```

A questo punto tutto è pronto per passare alla finalizzazione.

- Ci spostiamo sulla nostra workstation, apriamo il nostro web browser e accediamo all'URL <http://www.example.com/joomla>

Ci troveremo con la schermata qui di seguito

Figura 26: Joomla: prima schermata di installazione

- inseriamo il nome che vogliamo dare al sito e clicchiamo sul bottone “Configura dati di accesso”

NB: il nome lo possiamo cambiare in seguito se cambiamo idea

Web server e siti web. Manuale pratico in stile How To 1



The screenshot shows the Joomla! 5.1.0 installation interface. At the top, there is a dark blue header with the Joomla! logo on the left, the text "Installazione di Joomla" in the center, and a question mark icon on the right. Below the header, the main content area is white. It features two sections: "Seleziona la Lingua di Installazione" and "Imposta il Nome del Sito". The first section has a dropdown menu labeled "Seleziona la lingua" with "Italiano (it-IT) | Italiano (it-IT)" selected. The second section has a text input field labeled "Inserisci il nome del tuo sito Joomla. *" and a blue button labeled "Configura dati di accesso >". At the bottom of the page, there is a small line of text: "Joomla! is Free Software released under the GNU General Public License."

Figura 27: Joomla installazione: impostazione nome del sito

- la nuova pagina ci chiede i dati per creare l'utente amministratore generale del sito. Dopo la compilazione clicchiamo sul bottone "Configura la Connessione del Database". Impostiamo i seguenti dati

nome del tuo Super User:	Mario Rossi
Nome utente per il tuo account Super Amministratore:	jadminuser
Password del tuo account Super User:	jadminpass32FFi
email del Super User del sito:	mario.rossi@example.com

NB: scrivetevi subito i dati che inserite perché non si potranno recuperare. Pertanto se vi dimenticate la password, ad esempio, rimarrete chiusi fuori dal sito e dovrete reinstallarlo. Una volta entrati potete creare un secondo super user, un terzo, ecc...

Web server e siti web. Manuale pratico in stile How To 1



The screenshot shows the Joomla! 5.1.0 installation interface. The header includes the Joomla! logo and the title 'Installazione di Joomla!'. The main content area is titled 'Dati di accesso' (Access Data). It contains several input fields: 'Inserisci il vero nome del tuo Super User.' (filled with 'Mario Rossi'), 'Imposta il nome utente per il tuo account Super Amministratore.' (filled with 'jadminuser'), 'Imposta la password del tuo account Super User.' (masked with asterisks), and 'Inserisci l'indirizzo email del Super User del sito.' (filled with 'mario.rossi@example.com'). A green progress bar indicates 'Password accettata' (Password accepted) with the note 'Inserisci almeno 12 caratteri.' (Enter at least 12 characters). At the bottom is a blue button labeled 'Configura la Connessione del Database >'. A footer note states 'Joomla! is Free Software released under the GNU General Public License.'

Figura 28: Joomla installazione: impostazione dati di accesso a Joomla

- nella nuova pagina inseriamo i dati del database che abbiamo creato prima e clicchiamo sul bottone a fondo pagina "Installa Joomla". A titolo di promemoria i dati da inserire sono

Nome dell'host:	localhost
Nome utente del database:	joomlauser
Password del database:	joomlapass
Nome del database:	joomladb



The screenshot shows the Joomla! 5.1.0 installation interface for database configuration. The header is the same as Figure 28. The main content area is titled 'Configurazione database' (Database Configuration). It contains a dropdown menu for 'Seleziona il tipo di database.' (selected 'MySQL'), an input field for 'Immetti il nome dell'host, solitamente "localhost" o un nome fornito dal tuo host.' (filled with 'localhost'), an input field for 'Inserisci il nome utente del database che hai creato o un nome utente fornito dal tuo host.' (filled with 'joomlauser'), a masked password field for 'Inserisci la password del database che hai creato o una password fornita dal tuo host.', and an input field for 'Immetti il nome del database.' (filled with 'joomladb'). A final note says 'Immetti un prefisso delle tabelle o utilizza quello generato casualmente.' (Enter a table prefix or use the one generated randomly). The footer note is the same as Figure 28.

Figura 29: Joomla installazione: impostazione dati di accesso al database

- una barra di progressione ci terrà aggiornati sullo stato di installazione che durerà pochi secondi.

Web server e siti web. Manuale pratico in stile How To 1

Al termine ci verrà presentata la schermata qui di seguito che ci comunica che tutto è terminato ed è andato a buon porto.



Joomla! is Free Software released under the GNU General Public License.

Figura 30: Joomla installazione: pagine di fine installazione

L'installazione è terminata!

Joomla è completamente funzionante, ma è come una scatola vuota e disadorna.

5.3. Verifichiamo Joomla

Per verificare il nostro CMS:

- dal browser accediamo all'URL `http://www.example.com/joomla`
Comparirà la pagina di home vuota come detto sopra;
- ora accediamo all'URL `http://www.example.com/joomla/administrator`
- comparirà la pagina di login per la gestione del sito.

Inseriamo la username e la password definite prima durante l'installazione.

Accederemo alla home di gestione. Nel primo accesso ci verrà proposto un tour esplorativo di Joomla.

5.4. Preparazione per WordPress

Passiamo a installare il più noto CMS open source WordPress.

L'operazione è molto simile a quella che abbiamo fatto per Joomla.

Iniziamo dalla creazione del database e dell'utente per il DB. Faremo tutto, però, dal terminale:

- accediamo alla command-line di MariaDB

```
sudo mariadb -h localhost
```

- creiamo il database

```
CREATE DATABASE wordpressdb;
```

- creiamo l'utente per accedere al database con autorizzazioni solo per questo database

```
CREATE USER "wordpressuser"@"localhost" IDENTIFIED BY "wordpresspass";  
GRANT ALL PRIVILEGES ON wordpressdb.* TO "wordpressuser"@"localhost";  
FLUSH PRIVILEGES;
```

- e usciamo dalla command-line

```
\q
```

A questo punto abbiamo creato un database e un utente con permessi ristretti solo al database "wordpressdb". Siamo pronti ad eseguire l'installazione vera e propria di WordPress.

5.5. Installare WordPress

Iniziamo operando prima dal terminale e poi completeremo dal web browser della nostra workstation.

- Dal terminale spostiamoci nella directory temporanea che abbiamo prima creato

```
cd ~ && cd Temp
```

Web server e siti web. Manuale pratico in stile How To 1

- scarichiamo il pacchetto di WordPress 6.5.3 (la versione disponibile mentre scriviamo)

```
wget -O wordpress-6.5.3.zip https://wordpress.org/latest.zip
```

- estraiamo ed eliminiamo lo zip che non ci servirà

```
unzip wordpress-6.5.3.zip && rm wordpress-6.5.3.zip
```

- spostiamo il tutto nella web root

```
sudo mv wordpress /var/www/html/
```

- e infine sistemiamo i permessi

```
sudo chown www-data:www-data -R /var/www/html/wordpress
```

A questo punto tutto è pronto per passare alla finalizzazione:

- ci spostiamo sulla nostra workstation, apriamo il nostro browser web e accediamo all'URL <http://www.example.com/wordpress>
- nella pagina che si apre selezioniamo la lingua e clicchiamo sul bottone "Continua" a fondo pagina

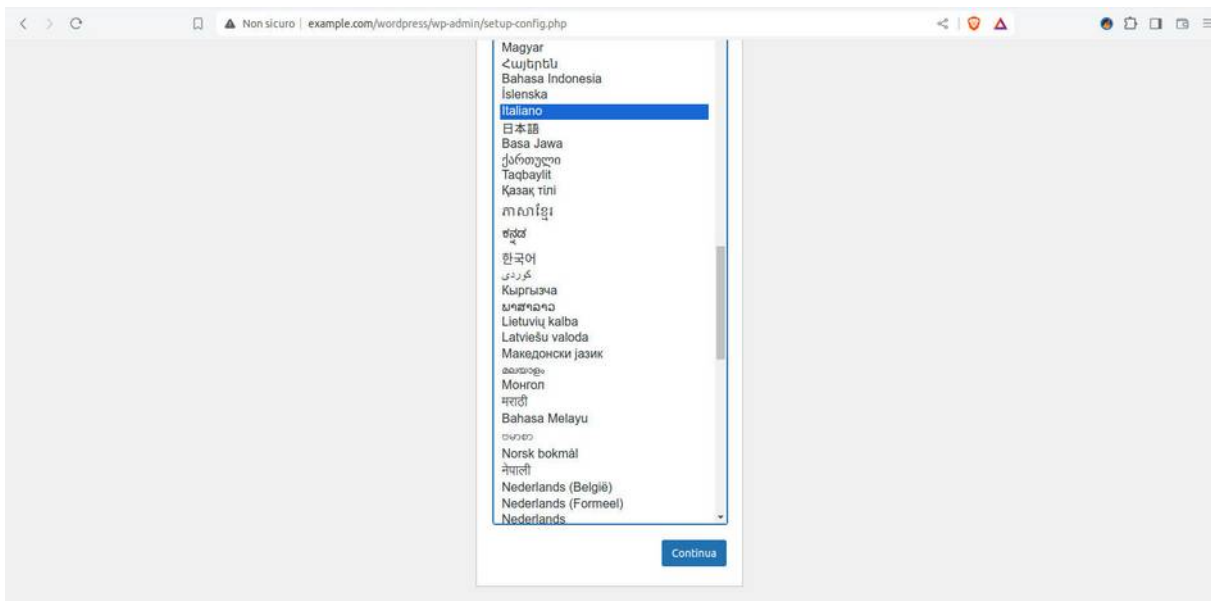


Figura 31: WordPress installazione: selezione della lingua delle pagine

- dopo qualche istante si apre la schermata che comunica l'inizio installazione; clicchiamo sul bottone "Iniziamo"

Web server e siti web. Manuale pratico in stile How To 1



Figura 32: WordPress installazione: prima informativa

- la nuova pagina ci chiede i dati per accedere al database. Dopo la compilazione clicchiamo sul bottone "Invia". Impostiamo i seguenti dati

Nome del database: wordpressdb
Nome utente: wordpressuser
Password: wordpresspass
Host del database: localhost



Figura 33: WordPress installazione: dati per l'accesso al database

- si apre una schermata di conferma. Clicchiamo sul bottone "Avvia l'installazione"

Web server e siti web. Manuale pratico in stile How To 1



Figura 34: WordPress installazione: seconda informativa

- nella pagina che segue inseriamo i dati per creare l'utente amministratore. Dopo aver completato la form clicchiamo sul bottone "Installa WordPress".

A titolo di promemoria i dati da inserire sono:

Titolo del sito: Mio sito WP

Nome utente: wpadminuser

Password: wpadminpass

La tua email: mario.rossi@example.com

NB: come per Joomla scriviamoci subito i dati di accesso e conserviamoli accuratamente;

Figura 35: WordPress installazione: creazione della account di accesso

- l'ultima pagina che appare è un promemoria delle credenziali inserite ed una conferma che tutto è andato a buon fine



Figura 36: WordPress installazione: informativa di riepilogo

L'installazione è terminata!

WordPress è completamente funzionante e popolato con qualche immagine e testo di esempio.

5.6. Verifichiamo WordPress

Per verificare il nostro CMS:

- dal browser accediamo all'URL `http://www.example.com/wordpress`
Comparirà la pagina default con una foto d'effetto ed un testo.
- Ora accediamo all'URL `http://www.example.com/wordpress/wp-login.php`
- inseriamo le credenziali inserite durante l'installazione e ci troveremo nelle pagine di gestione e impostazione del sito e dei contenuti.

5.7. Conclusione sui CMS

Esistono tanti diversi CMS. Ciascuno ha i suoi punti di forza specifici.

Abbiamo visto che l'installazione, dopo una prima formazione necessaria, è semplice. Impegnativa e decisamente più lunga è la parte di personalizzazione dell'aspetto del CMS (=template) e il caricamento dei contenuti.

Aspetto qui intenzionalmente non trattato, ma solo accennato, è la sicurezza: in parte dipende dal CMS stesso (e quindi si rimanda la questione alle aziende che progettano il CMS che non sempre brillano per buon codice), in parte dai web master (a volte l'assecondare il cliente si scontra con insufficienti competenze e soluzioni semplici e d'effetto, ma con intrinseche vulnerabilità) e dal sistemista che configura il server.

Quest'ultimo aspetto andrà affrontato. Servono diverse conoscenze su varie tecnologie e alcune specifiche sul server e sull'architettura su cui operiamo.

Queste prime pagine vanno lette come una prima formazione per capire come si fa e come funziona. Il risultato è funzionante e anche robusto, ma non ancora adatto per essere un reale web server internet.

6. Backup

Passiamo ad una componente fondamentale nella gestione del ciclo di vita del web server.

L'argomento è più complesso di quello che il nome evoca. Pertanto lo dividiamo in una prima parte dove vediamo di capire cos'è il backup e cosa si mette sotto backup. Successivamente vediamo quali tipi di backup esistono con i pregi e i limiti di ciascun approccio. Quindi vediamo un paio di soluzioni semplici di backup. Successivamente affronteremo la complessa questione della conservazione dei backup e chiuderemo con un piccolo excursus di confronto tra backup e disaster recovery.

6.1. Backup: cos'è

Il backup è semplicemente la copia di un file per riaverlo qualora l'originale scompaia per un qualsiasi motivo.

Il concetto semplice soffre diverse complicazioni quando guardiamo alla realtà dei nostri dispositivi di lavoro e ai server. Saltiamo a piè pari la questione dei dispositivi individuali e guardiamo ai server e ai siti web. Iniziamo pensando al lavoro che abbiamo fatto prima e diamoci l'obiettivo di fare il backup di PhpMyAdmin, Joomla e WordPress che abbiamo prima installato.

Fare il backup significa, dunque, copiare tutti i file che abbiamo messo nella web root (la directory `/var/www/html`). Inoltre dobbiamo copiare anche i database associati a Joomla e a WordPress. Infatti i file di un CMS senza il database non funzionano.

Fermiamoci qualche istante e facciamo qualche considerazione fondamentale sui file e sui database.

File:

- in primo luogo dobbiamo copiare insieme al file anche i permessi associati ai file.

Una prima difficoltà è legata ai programmi che copiano i file; non tutti i programmi copiano anche i permessi. Bisogna rilevare che tutti i programmi specializzati nel backup ormai salvano anche i permessi.

Questo, però, non risolve il problema. Infatti se restiamo sempre in server di tipo Linux, non è detto che il nuovo server abbia la stessa mappatura di permessi, di gruppi e di proprietari. Pertanto il file può conservare correttamente i permessi di sicurezza, ma l'UID 33, ad esempio, può essere associata ad un utente diverso rispetto al nostro nuovo server. In effetto domino questo produrrà problemi sul corretto funzionamento del web server che avrà problemi (anche potenzialmente bloccanti) nell'accedere ai file, alle directory e a creare-modificare i file;

Web server e siti web. Manuale pratico in stile How To 1

- un secondo problema rilevante è legato alle dimensioni totali dei backup. Normalmente i file prodotti dai software di backup sono file monolitici. Il che, in linea di principio, significa che hanno dimensioni uguali alla somma totale di tutti i file che compongono il sito (o i siti). Rapidamente dai circa 250MB occupati con le attività sopra descritte, si passa a diversi GB (vi assicuro che velocemente si raggiungono le decine di GB). File di dimensioni così grandi sono facilmente copiabili con gli attuali dispositivi di memoria e le velocità di copiatura, ma la manipolazione (estrazione, compressione, ecc...) non è né scontata, né di facile gestione anche per PC di ultima generazione.

Fondamentalmente si modera il rischio e la complessità con software specializzati per il backup;

- come accennato sopra i backup occupano sempre molto spazio. Normalmente vengono compressi i file generati dai software di backup. Questo significa dimensioni finali ridotte, in modo da poterli collocare su un disco estraibile, piuttosto che su cassette a nastro. Ma ciò comporta anche tempi e capacità di calcolo superiori. Va tenuto presente che il processo richiede anche dello spazio swap durante la creazione-compressione dei file.

Se per un web server dedicato, come abbiamo fatto con le operazioni sopra descritte, l'operazione crea solo un rallentamento nel funzionamento con piccole noie per i navigatori, per server reali, dove hanno centinaia o migliaia di accessi contemporanei, l'overhead può creare sospensioni di servizio e problemi nella creazione del backup;

- il backup per essere tale non deve essere sul disco del server per ovvi motivi. Se il server si rompe si perde anche il backup.

Ciò non significa, però, che siamo sollevati dal dover calcolare anche uno spazio libero per i backup. Servirà soprattutto per il restore, ma anche durante la creazione dei backup stessi. In prima approssimazione, del tutto esemplificativa, dobbiamo tenere presente che lo spazio disco disponibile deve essere uguale o superiore allo spazio occupato dai dati;

- un aspetto insidioso da tener presente sono i file volatili creati dai siti (anche dai semplici e popolari CMS Joomla e WordPress). Per una ragionevole economia di spazio non vanno copiati. Inoltre l'eventuale loro copia può creare problemi anche importanti nel restore. La difficoltà è legata al fatto che non esiste un solo standard di riferimento per tutte le varie soluzioni di siti web dinamici esistenti. Pertanto la configurazione del software di backup comporta anche un buon livello di competenza del tecnico che cura il server e/o il sito;
- i backup che otteniamo devono avere alcune caratteristiche costanti (altrimenti **non sono backup**).

Web server e siti web. Manuale pratico in stile How To 1

La prima è che devono essere su supporti di memoria diversi dal server e devono essere portabili (un disco esterno, una cassetta a nastro, ecc...).

Una seconda caratteristica è che devono essere criptati. Infatti se vanno in mano ad un estraneo non deve essere in grado di estrarli. Se avviene un accesso al contenuto di un backup c'è un accesso a potenziali dati riservati e, comunque, verrebbe a conoscere dati tecnici del sito come, nel nostro lavoro fatto fino qui, le credenziali di accesso al database.

Terzo elemento, di difficile gestione, il backup deve essere estraibile anche nel caso di guasto del sistema nativo di restore. Infatti se ci trovassimo nella situazione di un guasto o un incidente particolarmente grave del server, deve essere percorribile una via di restore alternativa. Di converso questo significa anche un adeguato livello skill dell'operatore e la conoscenza (o accesso) dell'operatore alle chiavi di decodifica;

- l'ultimo scenario introduce un'ulteriore insidia che resta quasi sempre nascosta: la consistenza dei dati contenuti nel file di backup.

Invisibile a molti, ma i computer fanno errori nel copiare i file. Anche i supporti di memoria si possono corrompere per cause del tutto imponderabili. A volte gli errori o le compromissioni sono minime, ma usando file criptati e compressi possono impedire il restore di tutto il backup.

Questo significa che per un backup affidabile dobbiamo usare prodotti di qualità (il che si traduce in costi maggiori rispetto ai prodotti che troviamo al supermercato o su Amazon), l'adozione di buoni processi, e una gestione competente;

- dobbiamo anche fugare un concetto gravemente forviante: il recupero ed il ripristino dei file contenuti in un backup non comportano che il sito torna in vita.

C'è da considerare il ripristino del database, che vedremo qui sotto. I dati del database devono essere sincroni con il momento di backup. Inoltre dobbiamo ripristinare il backup all'interno di un web server compatibile e che riproduca lo stesso ambiente software presente al momento della creazione del backup.

Ad esempio il ripristino dal backup del database MariaDB in PostgreSQL non ripristina correttamente i dati malgrado entrambi parlino l'SQL.

Il ripristino dei file del CMS in un web server che usi come web root il path `/srv/html` non permette il funzionamento al CMS che era installato nel path `/var/www/html`; il web server dovrebbe avere le stesse configurazioni ed i medesimi path oppure va configurato in modo totalmente compatibile con quello di origine.

Database:

- iniziamo da una premessa generale. Anche se molti database parlano SQL, ciascuno gestisce i dati in modo diverso. Inoltre per tutti i database non possiamo copiare i file che costituiscono la base dati, ma possiamo estrarre i dati ed esportarli in un formato legacy come l'SQL, il CSV, ecc...

Pertanto teniamo ben presente che l'export e l'import richiede una procedura che si basa su file intermedi.

Di converso questo significa che non possiamo salvare alcuni oggetti come gli indici. Al momento del ripristino le istruzioni SQL disporranno che il database ricrei questi oggetti, mentre il ripristino dei dati contenuti nelle tabelle comporta che il database ricodifica tutte le entry;

- come si può capire dalla premessa generale fatta al punto sopra il database lo salviamo tramite un'operazione di estrazione che ci permette di avere i dati in un formato intermedio.

L'estrazione dipende dal database in uso: `mysqldump`, `mariadb-dump`, `pg_dump` per gli RDBMS che abbiamo citato e usato prima. Possiamo usare le istruzioni native SQL da DBeaver, PhpMySQL o altro tool SQL.

In ogni caso otteniamo file lunghi (e spesso molto voluminosi) di testo base contenenti istruzioni SQL. Normalmente si comprimono e inseriscono o allegano al backup dei file dei siti;

- un ulteriore aspetto da conoscere sono le relazioni tra le tabelle. Tradizionalmente le relazioni erano nelle query che il CMS inoltra al database. Alcuni database salvano internamente le relazioni come anche procedure o script. Insomma: anche se tutti capiscono e parlano SQL i database sono molto diversi e il salvataggio in un formato legacy comporta:
 - potenziale perdita di meta-informazioni;
 - i dati estratti non sono significativi-usabili se non vengono ri-inseriti nel CMS (cioè: possiamo non trovare l'articolo come un record del database);
 - i file vanno ripristinati all'interno dello stesso tipo di database;
- altra piccola, ma insidiosa, particolarità è la codifica dei caratteri. Questo significa che il ripristino dei dati prevede che il database di destinazione sia stato creato con la stessa tipologia di codifica (es.: latin1, UTF8, UTF16, ecc..);
- ultimo elemento da conoscere riguarda MySQL e MariaDB. A differenza della maggioranza degli RDMS i database MySQL e MariaDB hanno motori diversi nel loro cuore: MyISAM, InnoDB, Memory, XtraDB, ecc..

Questo significa che il restore, per produrre dati corretti, necessita di tabelle associate allo stesso motore.

Questo aspetto specifico di MySQL e MariaDB non trova una controparte speculare in PostgreSQL o altri RDMS.

Il backup è un copia semplice dei file, ma già da queste caratteristiche generali vediamo che non è una copia facile.

6.2. Backup: tipologie e frequenza

Il mondo dei backup introduce altri tre elementi rilevanti: tipologia, snapshot e frequenza.

6.2.1. Tipologia

Iniziamo dalla tipologia. Per capire immaginiamo di fare oggi il backup dei file del nostro CMS e di rifare la stessa operazione tra una settimana. La quasi totalità dei file saranno gli stessi. Ragionevolmente avremmo qualche file nuovo che potrebbero essere le foto degli articoli aggiunti tra i due momenti di backup. In questa situazione semplice è un eccesso di zelo fare entrambe le volte una copia completa. La seconda volta è più ragionevole fare la copia solo dei file nuovi. Questo significa fare un backup molto più rapido e risparmiare molto spazio.

Un secondo caso da prendere in considerazione sono i file modificati. Ipotizziamo che il nostro CMS abbia un file di testo con i log degli accessi. Immaginiamo di fare come sopra il backup oggi ed un secondo backup a distanza di una settimana. Come detto sopra possiamo salvare solo file nuovi. Ma in questo caso dobbiamo salvare anche il file di log perché sarà cresciuto e, pur avendo lo stesso nome, sarà diverso.

Si intuisce che possiamo realizzare tipi diversi di backup: backup completo, incrementale e differenziale:

- **backup completo:** significa la copia integrale di tutti i file senza alcuna condizione o filtro;
- **backup incrementale:** significa la copia di tutti i nuovi file non presenti nel precedente backup;
- **backup differenziale:** significa la copia di tutti i file che sono cambiati rispetto al precedente backup.

6.2.2. Snapshot

Dopo la tipologia di backup è opportuno introdurre un elemento insidioso. Come spiegato è necessario del tempo per creare il backup. Più grande è il sito (o i siti) maggiore è il tempo necessario. Pertanto possiamo incorrere in situazioni dove avvengono modifiche nei file o nel database durante il backup. Si tratta di una situazione difficile da gestire. Una soluzione è il blocco temporaneo del sito e dei servizi, ma questo introduce un disservizio che può essere molto antipatico. Una seconda soluzione può essere nella generazione di snapshot ed il backup viene fatto nei dati dentro lo snapshot. Questa tec-

Web server e siti web. Manuale pratico in stile How To 1

nica si basa sulla capacità di alcuni file system e sistemi operativi di congelare un istante dell'hard disk senza bloccare l'attività e permettendo l'accesso ai dati presenti nello snapshot e successivamente di cancellarlo. Il processo è poco impattante in termini di calcolo e performance, ma richiede un accesso sistemistico e non tutti i server supportano gli snapshot.

Possiamo considerare che gli snapshot sono in se stessi un backup, ma comportano dei problemi. In primis è una meta-archiviazione dello stato dei bit del disco che appesantisce il disco e che crea un progressivo decremento delle performance (infatti più tempo passa, più sono le differenze che vanno salvate). Poi si tratta del congelamento dell'intero disco, non possiamo isolarlo a una sola directory, né distribuirlo su più utenti e ciascuno gestisce una parte specifica del disco.

In conclusione gli snapshot sono una fantastica tecnologia, ma non sono un backup, ma possono aiutare nel creare backup.

6.2.3. Frequenza

Non esiste una tempistica ideale, corretta e universale, ma abbiamo delle buone pratiche da tenere presenti per avere una corretta politica dei backup:

- il backup va eseguito nel momento di minor carico e minor accesso. Idealmente nel cuore della notte, ma nel caso di siti web in produzione che hanno visite da tutto il mondo non esiste una fascia oraria ideale. Potenzialmente tutte le ore sono momento di accesso;
- la messa in produzione del sito è il primo momento di backup completo;
- usando i CMS, o soluzioni con architettura analoga, possiamo investire su un backup frequente del solo database, magari giornaliero, e periodico uno incrementale e un full con frequenza mensile o bimestrale;
- è buona pratica fare un backup completo prima di ogni intervento massivo o potenzialmente rischioso (come aggiornamenti, aggiunta di funzionalità, cambio del template, ecc...);
- in fine è bene scegliere le frequenze e tipologie di backup alla luce della frequenza dell'aggiornamento dei contenuti e/o delle attività degli utenti (nel caso di siti attivi come shopping online, forum, blog con possibilità di commenti, ecc...).

A valle di tutto ciò si può guardare a una pianificazione come segue:

1. backup completo alla messa in produzione, prima di ogni intervento impattante-pericoloso e post intervento finiti i test di verifica;
2. backup settimanale del database e differenziale;
3. backup quindicinale incrementale;
4. backup completo bimensile.

6.3. Backup in esecuzione

Esistono diverse soluzioni. In linea di principio non è una politica saggia una soluzione di solo backup cloud.

Nel caso di servizi acquistati su provider spesso offrono soluzioni di backup con un piccolo costo aggiuntivo; sono da prendere in considerazione con la precauzione che i file fanno anche copiati in locale, su un proprio supporto di memoria (hard disk esterno, pendrive, ecc...).

Nel caso in cui abbiamo la possibilità di gestione del server (come abbiamo fatto in questo manuale) possiamo usare uno strumento tradizione come tar, o il potente rsync o adottare software più completi come Amanda o Bacula.

6.3.1. Device

In primo luogo abbiamo bisogno di un supporto di memoria di massa diverso dal web-server. Inoltre l'ideale è che i singoli supporti (dischi, cassette, ecc...) si possano *asportare*.

Una politica che ha dimostrato la sua bontà è l'adozione di un set di supporti: metà sono nel sistema di backup, metà in cassaforte in un altro edificio. Periodicamente i supporti vengono girati. La bontà della soluzione si scontra con due problemi:

- le quantità di dati che richiedono supporti molto capienti. L'esplosione delle dimensioni dei dati rende eccessivamente grande (e costoso) l'adozione di un adeguato set di supporti;
- il secondo problema è il lavoro umano: ripetitivo, delicato e costoso.

La soluzione con due set di supporti resta molto valida, oggi, solo per server di piccole dimensioni e per server privati.

I noti NAS (sistemi di storage su disco) possono essere usati come device di backup. Esistono versioni per tutte le tasche e per molti casi d'uso. Va considerato, innanzitutto, che i modelli piccoli hanno schede di rete che possono essere lente per archiviare backup grandi. Inoltre molti modelli hanno un solo alimentatore rendendoli vulnerabili ai guasti che lo possono riguardare. Se guardiamo ai modelli entry level non sono in formato rack mount.

Sul mercato server esistono diverse appliance dedicate al backup: sono dispositivi rack mount che si alloggiano negli armadi dei server. Connessi direttamente alla fibra del CED o ai server sono particolarmente flessibili. Il controllo remoto, normalmente fatto dal software stesso del backup, permette di automatizzare tutta l'attività. Le tradizionali tape library fanno automaticamente anche la rotazione delle cassette.

Il punto debole di questa soluzione sta nell'ubicazione: le appliance ed i device sono nello stesso luogo dei server. In caso di un evento grave (un fulmine, un incendio, un terremoto, ecc...) insieme ai serve perdiamo anche i backup.

Web server e siti web. Manuale pratico in stile How To 1

Una mitigazione di queste debolezze la introduce la rete. Ad esempio collegando due edifici tramite una fibra, con schede veloci (come quelle da 800G), possiamo trasferire da un edificio all'altro i backup velocemente.

La grande quantità di dati, nel caso di server e server farm (=cloud) limita la bontà di queste soluzioni che comunque sono soluzioni molto costose.

L'adozione della rete apre le porte all'uso del cloud per il backup. Ci sono più provider che vendono soluzioni di backup online. A conti fatti sono convenienti e normalmente offrono un buon software. Dobbiamo però considerare la lentezza di internet: trasferire qualche giga oggi non è particolarmente lungo, ma spostare set interi di dati via internet può richiedere tempi molto lunghi. Inoltre dobbiamo considerare l'ipotesi che la connessione non sia disponibile, o sia troppo lenta, o discontinua... Tutte situazioni che inficiano la bontà di un backup basato solo sul cloud. Nascosto, ma molto rilevante, c'è anche il problema della sovranità dei dati quando il server fisico si trova in continenti diversi dal proprio con normative diverse...

insomma: in ultima analisi non è una buona soluzione se la immaginiamo come soluzione unica o principale.

Per tornare agli obiettivi di questo manuale e immaginando uno scenario didattico con un dataset contenuto (immaginiamo qualche centinaio di GB) possiamo guardare con interesse a una soluzione basata su una tipo library o una con hard disk estraibili su bus veloci (i vecchi SATA, o un SAS, un NVMe, ecc...).

La rotazione dei supporti la possiamo gestire manualmente o tramite automazione software. La soluzione sarà abbastanza economica e soprattutto funzionale. Se optiamo per una rotazione manuale dei supporti l'effort umano aggiuntivo sarà contenuto.

6.3.2. Tar

Nato per archiviare su nastro magnetico in ambiente Unix, è diventato successivamente uno elemento dello standard Posix, motivo per cui è uno standard di fatto per tutti gli ambienti Unix e *nix come Linux e Mac.

Come *software a se* non prevede nativamente una gestione di tutto il ciclo di vita dei backup. Restando su obiettivi piccoli possiamo creare qualche script che lancia tar, magari comprimendo i file di output e salvandoli su un disco rimovibile. Il restore, di un file o dell'intera collezione, sarà una procedura manuale.

Cercando sarà facile trovare soluzioni basate su script che creano, in modo semiautomatico, i backup.

A titolo d'esempio qui di seguito degli script di backup basati su `tar`. Gli script creeranno file compressi tramite `lbzip2` che sfrutta tutti i core ed i thread del processore accelerando notevolmente i tempi di compressione. I file generati vengono messi nella directory `/home/backuser/backup/daily`, accessibile via SFTP per il download.

Web server e siti web. Manuale pratico in stile How To 1

Gli script si ispirano alla guida di Tony Teaches Tech «How to Backup Daily, Weekly, and Monthly with tar, rsync, and cron» <https://tonyteaches.tech/rsync-backup-tutorial>.

Iniziamo installando l'utility `lbzip2`

```
sudo apt install -y lbzip2
```

aggiungiamo l'utente `backuser` con password `MiaBak73Pass##webSrv` per il download via SFTP

```
sudo useradd -m -s /usr/sbin/nologin backuser
passwd backuser
New password: MiaBak73Pass##webSrv
Retype new password: MiaBak73Pass##webSrv
passwd: password updated successfully
```

NB 1: abbiamo creato un utente che non può loggarsi (`-s /usr/sbin/nologin`), ma ha una password;

NB 2: per motivi di migliore sicurezza è bene che il client SFTP si colleghi tramite certificato invece che password.

Creiamo le directory dei backup

```
sudo mkdir -p /home/backuser/backup/daily
sudo mkdir -p /home/backuser/backup/weekly
sudo mkdir -p /home/backuser/backup/monthly
sudo chown backuser:backuser -R /home/backuser/backup
```

Aggiungiamo l'utility di compressione `lbzip2`

```
sudo apt install lbzip2
```

Passiamo ora a creare i tre script di backup. Poi automatizzeremo l'esecuzione tramite cron.

1. Creiamo i file degli script rendendoli eseguibili

```
sudo touch /usr/local/sbin/bak-daily.sh
sudo touch /usr/local/sbin/bak-weekly.sh
sudo touch /usr/local/sbin/bak-monthly.sh
sudo chmod 700 /usr/local/sbin/bak-daily.sh
sudo chmod 700 /usr/local/sbin/bak-weekly.sh
sudo chmod 700 /usr/local/sbin/bak-monthly.sh
```

2. editiamo `/usr/local/sbin/bak-daily.sh`

```
sudo nano /usr/local/sbin/bak-daily.sh
```

3. e popoliamolo come segue

```
#!/bin/bash

#
# backup GIORNALIERO
# /usr/local/sbin/bak-daily.sh

# creazione del backup dei file
```


Web server e siti web. Manuale pratico in stile How To 1

```
tar -I lbzip2 -cf /home/backuser/backup/daily/backup-$(date +%Y%m%d).tar.gz --
absolute-names /var/www/html

# creazione del backup dei database
mariadb-dump --all-databases | lbzip2 -c > /home/backuser/backup/daily/backup-$(
date +%Y%m%d).sql.bz2

# correzione PERMESSI
chown backuser:backuser -R /home/backuser/backup/daily

# rimozione dei backup più vecchi di 7 giorni
find /home/backuser/backup/daily/* -mtime +7 -delete
```

4. editiamo /usr/local/sbin/bak-weekly.sh

```
sudo nano /usr/local/sbin/bak-weekly.sh
```

5. e popoliamolo come segue

```
#!/bin/bash

#
# backup SETTIMANALE
# /usr/local/sbin/bak-weekly.sh

# creazione del backup dei file
tar -I lbzip2 -cf /home/backuser/backup/weekly/backup-$(date +%Y%m%d).tar.gz --
absolute-names /var/www/html

# creazione del backup dei database
mariadb-dump --all-databases | lbzip2 -c > /home/backuser/backup/daily/backup-$(
date +%Y%m%d).sql.bz2

# rimozione dei backup più vecchi di 31 giorni
find /home/backuser/backup/weekly/* -mtime +31 -delete
```

6. editiamo /usr/local/sbin/bak-monthly.sh

```
sudo nano /usr/local/sbin/bak-monthly.sh
```

7. e popoliamolo come segue

```
#!/bin/bash

#
# backup MENSILE
# /usr/local/sbin/bak-monthly.sh

# creazione del backup dei file
tar -I lbzip2 -cf /home/backuser/backup/monthly/backup-$(date +%Y%m%d).tar.gz --
absolute-names /var/www/html

# creazione del backup dei database
```

Web server e siti web. Manuale pratico in stile How To 1

```
mariadb-dump --all-databases | lbzip2 -c > /home/backuser/backup/daily/backup-$(date +%Y%m%d).sql.bz2
```

```
# rimozione dei backup più vecchi di 365 giorni
find /home/backuser/backup/monthly/* -mtime +365 -delete
```

8. invochiamo l'editor `crontab` per root in modo che gli script vengono eseguiti al di sopra di ogni restrizione

NB: questo approccio garantisce una esecuzione senza problemi, ma non è una buona scelta di sicurezza

```
sudo crontab -e
```

9. e inseriamo le seguenti stringhe

```
15 0 * * * /usr/local/sbin/bak-daily.sh
30 0 * * 1 /usr/local/sbin/bak-weekly.sh
45 0 1 * * /usr/local/sbin/bak-monthly.sh
```

Abbiamo completato un semplice sistema di backup che copia tutti i file, estrae tutti i database (eccetto `mysql` dove sono salvate le configurazioni, i permessi, gli utenti e le password), comprime tutti i backup al volo, gestisce la conservazione con una rotazione giornaliera, settimanale, mensile, annuale e permette lo scaricamento remoto tramite un accesso sicuro SFTP.

6.3.3. Rsync

`rsync` è uno storico programma di sincronizzazione di directory e file. Nato in ambiente *nix sono disponibili porting nativi per i principali sistemi operativi. I sorgenti sono disponibili per tutti garantendo portabilità e mantenimento.

Il programma è molto interessante perché è costruito intorno alla logica delta: si sincronizzano le differenze. Adottando questo principio si minimizzano i trasferimenti offrendo un'alta performance. L'uso su rete impiega la porta TCP 873.

accanto alla logica delta il programma permette diverse logiche per attuare la sincronizzazione. Come programma autonomo è facile integrarlo all'interno di script, anche in ambiente Windows magari integrandolo nei potenti script PowerShell.

Per molti utenti sarà una sicuramente un'importante difficoltà l'assenza di una finestra grafica per l'invocazione. Comunque esistono applicazioni di terze parti che offrono potenti e comode finestre grafiche. Da considerare anche la complessità delle opzioni: la disponibilità di tante possibilità avanzate e di dettaglio creano complessità e richiedono competenze superiori al livello medio. Anche la sua struttura client-serve è una difficoltà aggiuntiva per utenti alle prime armi.

Come per `tar` anche `rsync` non prevede nativamente una gestione di tutto il ciclo di vita dei backup. Possiamo gestire la mancanza tramite script come abbiamo fatto prima con `tar`.

Web server e siti web. Manuale pratico in stile How To 1

Per un'implementazione è sufficiente googlare e la rete offre diversi esempi pronti all'uso.

Una piccola miniera di informazioni e guide è il sito ufficiale: <https://rsync.samba.org> .

Allo scopo di facilitare la ricerca riportiamo alcune risorse-esempi di qualità presenti in rete:

- Backup automatico con rsync via ssh,
https://guide.debianizzati.org/index.php/Backup_automatico_con_rsync_via_ssh
- Learning Rsync,
https://docs.rockylinux.org/it/books/learning_rsync/01_rsync_overview/
- Using rsync to back up your Linux system,
<https://open source.com/article/17/1/rsync-backup-linux>
- Come usare rsync, esempi pratici,
<https://devdev.it/comando-rsync-esempi-pratici-144/>
- Rsync: cos'è e come usarlo per creare backup dati sincronizzati in cartelle locali e remote,
<https://www.cybersecurity360.it/soluzioni-aziendali/rsync-cose-e-come-usarlo-per-creare-backup-dati-sincronizzati-in-cartelle-locali-e-remote>

6.3.4. Amanda

Amanda (<https://www.zmanda.com>) è vero software di backup di rete. Disponibile per i principali sistemi operativi è concepito per backuppare PC, singoli server fino a grandi server farm anche geograficamente distribuite. Il suo punto di forza è la centralizzazione, l'automazione e l'indipendenza sia dal sistema operativo, sia dall'hardware e anche dal media di conservazione dei backup.

Tra i punti di forza c'è la doppia disponibilità: versione in totale open source e versione aziendale distribuita con il nome Zmanda. la versione open source è di serie nei repo di tutte le principali distribuzioni Linux. Inoltre Amanda prevede tutto quello che serve ad un moderno, efficiente e sicuro sistema di backup di rete.

L'implementazione è un'attività che richiede un po' di spiegazioni, un po' di studio e un po' di tempo. Insomma: non possiamo ridurla a due semplici pagine d'esempio. Qui di seguito la rappresentazione dell'architettura globale di Amanda o Zmanda per offrire un'idea della complessità e delle potenzialità.

Web server e siti web. Manuale pratico in stile How To 1

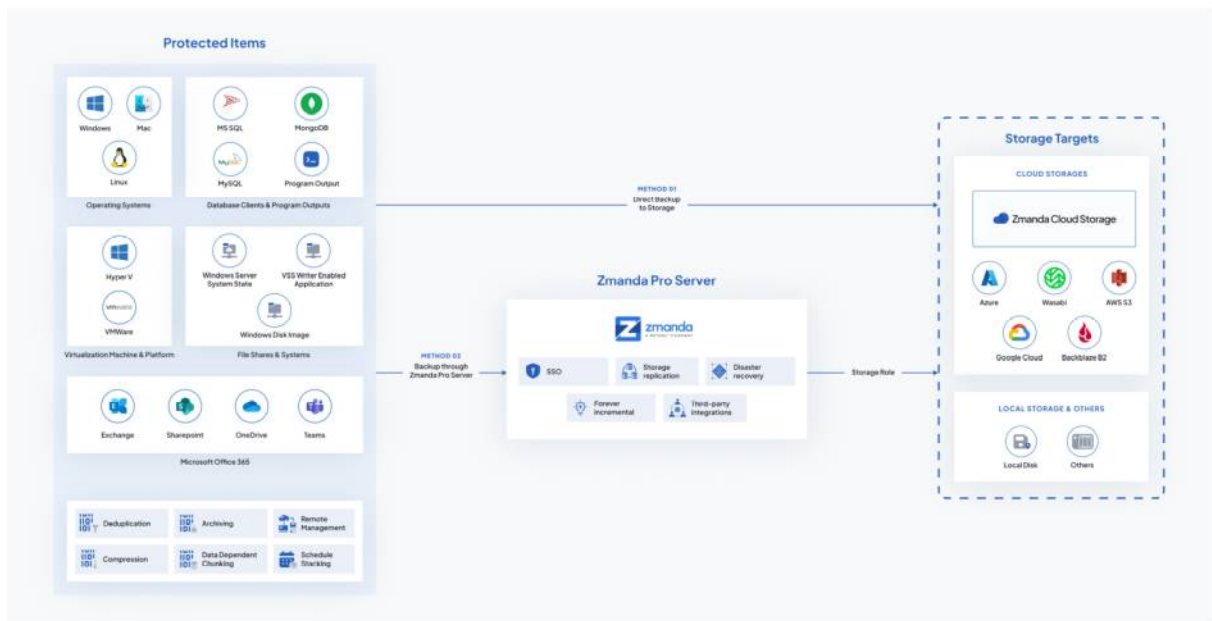


Figura 37: Amanda e Zmanda: architettura generale

Per l'implementazione rimandiamo alla documentazione ufficiale: esaustiva e ben fatta sia per la versione community, sia per quella enterprise. In caso di bisogno sono disponibili sia sottoscrizioni, sia aziende che offrono il servizio di installazione-configurazione e/o l'assistenza.

Qui di seguito alcuni link, a partire da quelli del prodotto, quelli della documentazione e qualche guida:

- Amanda,
<https://www.amanda.org/>
- Zmanda,
<https://www.zmanda.com/>
- Zmanda Knowledge Base,
<https://kb.zmanda.com/>
- Welcome to the Zmanda Pro Documentation,
<https://docs.zmanda.com/>
- Amanda wiki,
<https://wiki.zmanda.com/>
- GitHub Zmanda,
<https://github.com/zmanda/amanda>

6.3.5. Bacula

Bacula (<https://www.bacula.org>) è un software di backup di rete come Amanda. Si tratta di un software di tipo enterprise, che permette una gestione totalmente centralizzata, automatizzata e securizzata.

Permette di tenere sotto backup PC, singoli server e server farm singole o geograficamente distribuite.

Accanto alla versione open source c'è la versione enterprise con servizi di sottoscrizione, mantenimento e servizi di assistenza rintracciabili al sito ufficiale: <https://www.baculasystems.com>.

La grande potenzialità si accompagna a una complessità importante.

Bacula, forse, a differenza di Amanda e Zmanda, offre qualcosa in più anche nella versione solo open source, come, ad esempio, l'interfaccia grafica e l'integrazione nativa con alcuni tool di gestione sistemistica.

L'implementazione è un'attività che richiede un po' di spiegazioni, un po' di studio e un po' di tempo. Insomma: non possiamo ridurla a due semplici pagine d'esempio. Qui di seguito la rappresentazione dell'architettura globale di Bacula per offrire un'idea della complessità e delle potenzialità.

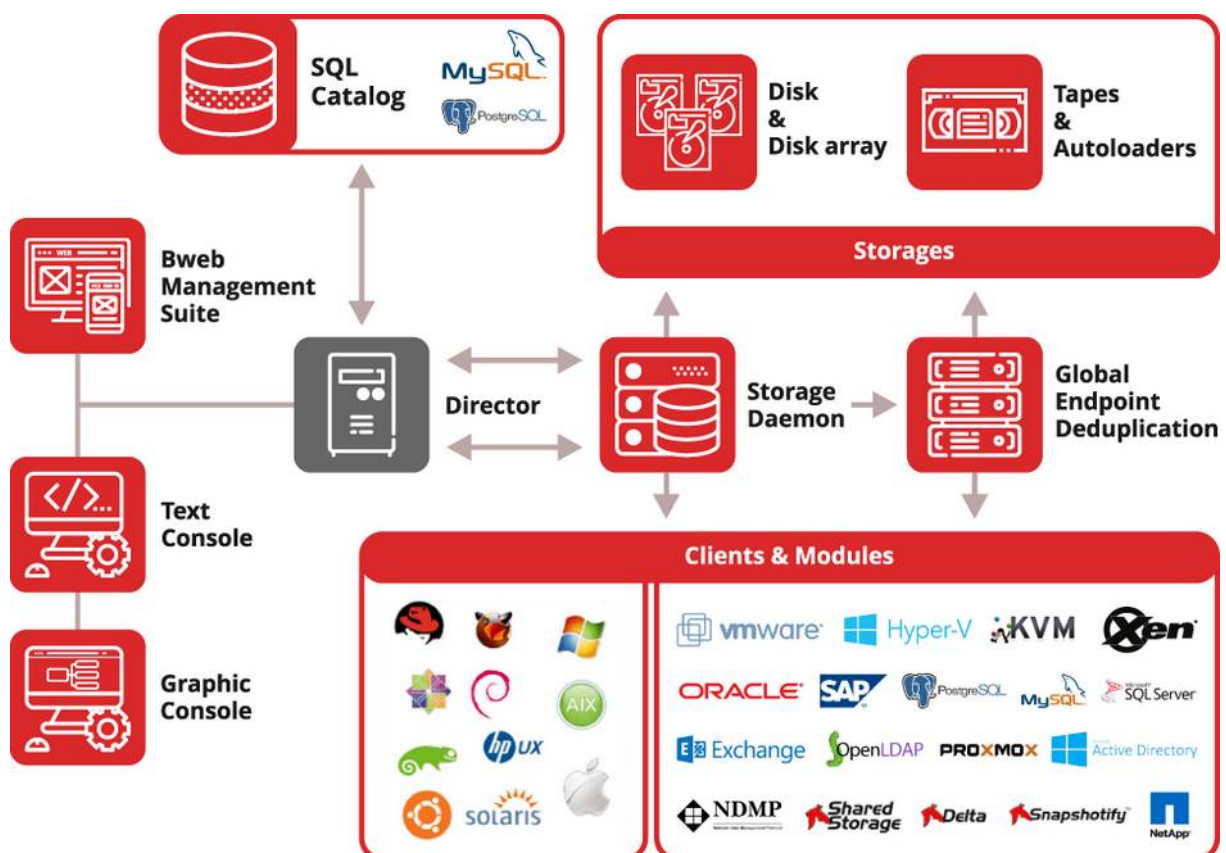


Figura 38: Bacula: architettura generale

Per l'implementazione rimandiamo alla documentazione ufficiale: esaustiva e ben fatta sia per la versione community, sia per quella enterprise. In caso di bisogno sono disponibili sia sottoscrizioni e aziende che offrono il servizio d'installazione-configurazione e/o l'assistenza.

Segnaliamo il tool Baculum (<https://www.bacula.it/community/baculum-9-graphical-bacula-configuration-administration-and-api/>). Si tratta di un completo, potente e comodo software di gestione e controllo di Bacula via web.

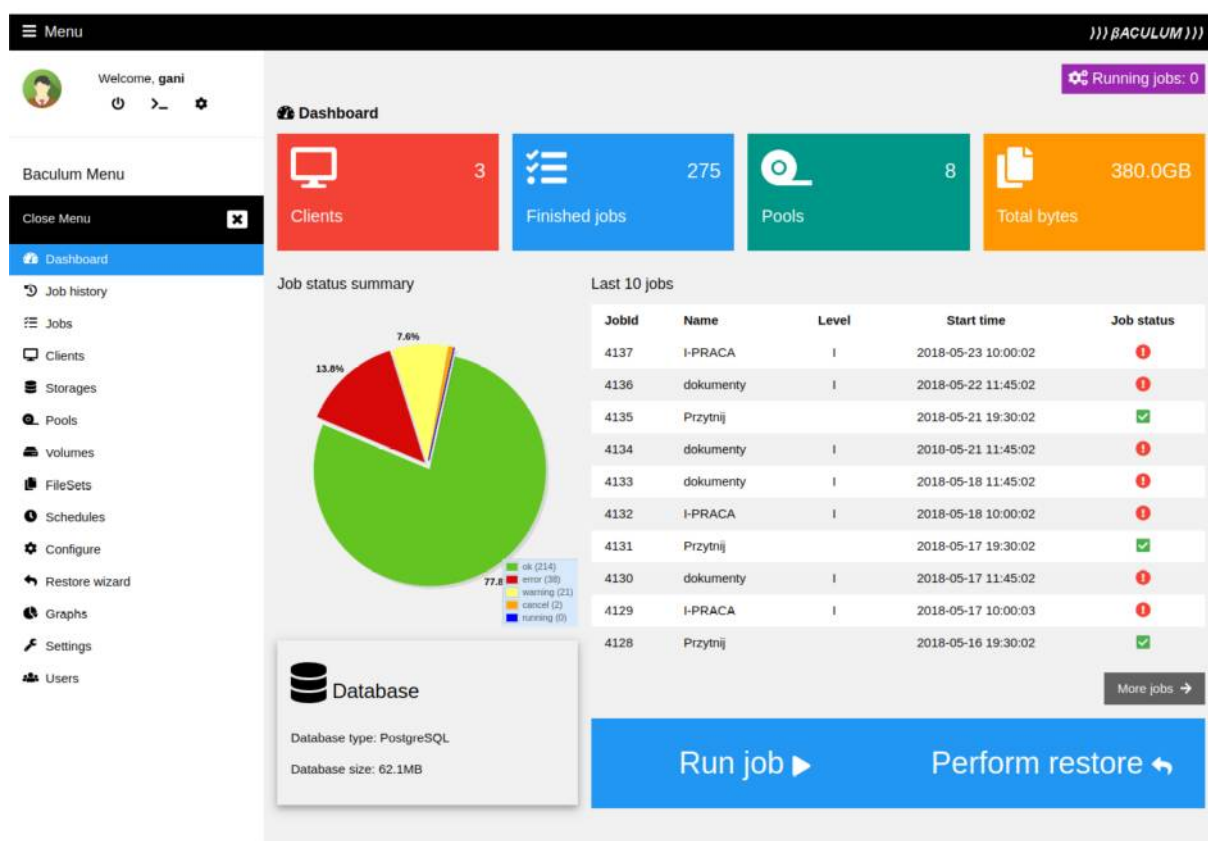


Figura 39: Schermata di Baculum

Qui di seguito alcuni link, a partire da quelli del prodotto, della documentazione e qualche guida:

- Bacula Systems - backup software for modern data center,
<https://www.baculasystems.com>
- The best open source backup software for Linux,
<https://www.bacula.org/>
- Bacula® Manuals,
<https://www.bacula.org/documentation/documentation/>

- How to install and configure Bacula,
<https://ubuntu.com/server/docs/how-to-install-and-configure-bacula>
- How I use the Bacula GUI for backup and recovery,
<https://open source.com/article/22/5/baculum-open source-backup>

6.4. Backup: la conservazione

La conservazione dei backup si organizza fondamentalmente su due elementi:

- **tempo**: rotazione e tempo di ritenzione dei device;
- **luogo**: sito dello storage e/o cloud.

Dobbiamo, però, aggiungere subito un terzo elemento di natura discrezionale: **importanza** e **rilevanza** dei dati. A seconda dell'importanza di questi elementi si organizza tutto il resto.

Teniamo presente che la conservazione dei backup è un costo.

Completata questa piccola premessa ricordiamoci che abbiamo già affrontato, in parte, l'argomento.

6.4.1. Tempo

Per tempo dobbiamo pensare alla durata della disponibilità di una certa versione dei nostri file.

Con i moderni sistemi di siti web insieme ai file dobbiamo guardare soprattutto ai database in backend al sistema di sito e alle configurazioni dell'ambiente.

Idealmente possiamo pensare a 12 mesi massimi di conservazione.

Accanto al tempo dobbiamo valutare la frequenza di rotazione. Come visto sopra un buon approccio è:

- copia giornaliera con ritenzione settimanale;
- copia settimanale con ritenzione mensile;
- copia mensile con ritenzione annuale.

6.4.2. Luogo

Anche questo argomento è già stato affrontato.

Il concetto centrale irrinunciabile è che il backup non risiede sul server e i supporti di memoria, che contengono i backup devono essere rimovibili.

Web server e siti web. Manuale pratico in stile How To 1

Riproponiamo una riflessione sulla soluzione cloud diventata popolare, economica e, soprattutto, ci sgrava da diversi aspetti logistici. In realtà introduce diversi problemi a iniziare dalla sovranità sui dati, alla dipendenza dalla connessione internet e alle tempistiche eccessivamente lunghe per il completo trasferimento attraverso internet.

Se la soluzione cloud è ottima nel caso di piccoli server (che generano file di qualche giga), legati a servizi non critici, in tutti gli altri casi il cloud è positivo solo come device aggiuntivo.

Discorso leggermente diverso nel caso di una sottoscrizione ad un servizio di web hosting, dove, a fronte di una integrazione di contratto, possiamo avere il backup insieme all'hosting. In questo specifico caso abbiamo molti vantaggi ad avere tutto integrato. Dobbiamo pensare e decidere circa la sovranità dei backup (quindi se averne una copia in locale e con quale frequenza farli) e l'eventuale gestione della migrazione di provider.

Per quest'ultima situazione (migrazione da un provider ad un altro) teniamo presente che possiamo trovarci con configurazioni dell'ambiente web-server diverse che rendono incompatibili i backup.

Dopo questa premessa (o promemoria) vediamo come gestire le copie fuori dal server.

Una soluzione intuitivamente semplice è dotare il server id un **NAS**. Accanto alla semplicità all'economicità dei NAS retail target dobbiamo considerare:

- le interfacce di rete domestiche sono troppo lente per volumi importanti da backuppare. Già 100GB creano qualche problema;
- i NAS comuni prevedono facilmente il cambio a caldo dei dischi, ma spesso non permettono di mettere un adeguato numero di dischi. Inoltre una volta estratti i dischi possono non permettere un accesso diretto ai file contenuti come, ad esempio, quando sono parte di un array RAID;
- per avere una buona performance del NAS dobbiamo collocarlo nel CED dove si trova il web server creando il cortocircuito che un grave incidente nel CED (es.: un incendio) compromette anche il backup;
- i NAS con caratteristiche che risolvono i limiti sopra hanno costi analoghi come i sistemi di storage.

Un'alternativa molto valida è l'adozione di uno **storage**:

- possiamo immaginare ad un server con configurato come storage o a uno storage nativo. Entrambe le soluzioni possono risolvere i problemi sopra. Probabilmente un server configurato a storage permette, a parità di risultati, di spendere di meno;
- uno storage ha costi decisamente superiori ad un NAS (eccetto i casi di NAS di alta fascia per CED);

Web server e siti web. Manuale pratico in stile How To 1

- dotati di interfacce di rete su fibra permette di avere transfer rate adeguati, rendendo funzionale il backup;
- la vulnerabilità legata alla collocazione all'interno del CED è compensata dalla possibilità di estrazione e cambio dei dischi a caldo;
- con una dotazione di questo tipo è facile integrare automaticamente un nodo mirror collocato in un diverso edificio (o diverso luogo geografico) mitigando ulteriormente il rischio legato ad incidenti gravi nel CED;
- resta che i costi totali diventano rilevanti.

Ancora valide sono le tradizionali soluzioni basate su appliance **tape library**:

- sono soluzioni poco note, ma largamente consolidate;
- sono ben integrate con tutto l'ecosistema di backup;
- abbiamo cassette a nastro di buona capienza;
- permettono la rotazione delle cassette (a titolo d'esempio: una muta nel caricatore della tape library e una in cassaforte);
- sono macchine completamente automatizzate;
- però sono dispositivi lenti sia in scrittura che in ripristino.

Per i media che mettiamo in cassaforte:

- ovviamente deve essere un luogo sicuro,
- è bene che sia in condizioni ambientali controllate (raffrescato, a umidità controllata, non polveroso, senza campi magnetici, senza raggi solari diretti sui media e possibilmente con un sistema di log degli accessi e uscite).

Infine ricordiamoci dei grandi limiti dei supporti di memoria:

- le scritture magnetiche deteriorano con il tempo. Nastri e dischi vanno copiati e dismessi prima del tempo medio di persistenza del dato magnetico;
- tutte le tecnologie che abbiamo hanno una durata prestabilita. I backup vanno conservati meno del tempo medio di vita del supporto;
- le tecnologie quando diventano obsolete impediscono l'utilizzo dei supporti. Pertanto bisogna cambiare tutto prima che una tecnologia diventi obsoleta.

6.5. Backup VS disaster recovery

Il backup ed il disaster recovery sono cose diverse, anche se complementari:

- il disaster recovery è la copia delle informazioni necessarie per ripristinare il PC o il server o il cellulare;
- il backup è una copia dei dati.

Web server e siti web. Manuale pratico in stile How To 1

Obiettivo del disaster recovery è di ripristinare la funzionalità della macchina (pc, server o cellulare) e dei servizi, ma non di offrire il ripristino dei dati.

Obiettivo del backup è di ripristinare i dati, ma non la funzionalità della macchina e dei servizi.

Pertanto le due pratiche funzionano a reciproco completamento.

Nel caso di un web server in produzione è importante (e necessario) avere entrambi, averli aggiornati e avere gli operatori capaci di effettuare sia il recovery, sia il restore.

7. Consultare-Analizzare i log

Torniamo alla parte operativa del nostro web server. Affrontiamo l'importante argomento dei log. Iniziamo con una piccola sezione di teoria, per poi capire dove sono e come vi si accede e, in fine, prendere familiarità con i tool di analisi e rappresentazione dei log.

Qui è opportuno sottolineare che l'analisi dei log è strategica soprattutto nei casi d'uso professionali:

- ci permettono di capire, dati alla mano, cosa viene cercato e acceduto nel nostro sito, aspetto determinante se abbiamo uno shop o vogliamo aumentare i nostri affezionati;
- ci permette di vedere e scovare malintenzionati e possibili vulnerabilità del nostro sito;
- in caso di danni e/o breach abbiamo la possibilità di dati e analisi anche ai fini delle compliance legali.

7.1. Log: la teoria

Apache HTTP Serve genera 4 file di log per ogni server virtuale definito nelle sue configurazioni: 2 per gli accessi in HTTP, 2 per gli accessi in HTTPS.

Un log è riservato per gli accessi. Il secondo per gli errori ed i warning.

La configurazione di default dei log è diventata, di fatto, uno standard. È possibile, in ogni caso, modificare la configurazione per rilevare più informazioni o ridurre quanto viene tracciato.

I log sono file di testo semplice che vengono scritti. Ciò significa che il server deve dedicare un certo numero di cicli processore a creare queste stringhe e spendere cicli di IO per la scrittura di questi sul disco. Finché si tratta di test o di un piccolo server con pochi accessi al minuto non vedremo nessun decremento nelle prestazioni. Ma in server con decine o migliaia di accessi è un problema di assoluto rilievo che rallenta significativamente il server.

Web server e siti web. Manuale pratico in stile How To 1

Data la loro natura i log sono file che crescono continuamente. È importante non farci ingannare dalle grandi capacità di memoria degli hard disk. Con un traffico medio-piccolo possiamo avere alcuni giga all'ora. Il che significa che avremo tera di log nell'arco di un paio di settimane e arrivare a riempire un disco in pochi mesi.

In ultima analisi i log sono una preziosa risorsa da usare con molta attenzione.

Ai fini di gestione, profilazione, promozione, ecc... vanno assolutamente usati ed elaborati con qualche tool specializzato.

Infine i sistemi *nix (Linux + Unix) di serie hanno una configurazione di rotazione dei log che controlla automaticamente la crescita dei log comprimendoli, controllando lo spazio totale occupato e cancellandoli periodicamente. Cosa fondamentale perché se il disco si riempie il web-server si blocca e gli utenti vedranno o un errore di time-out o un errore di risorsa-pagina non disponibile. In ogni caso una situazione imbarazzante per un sito serio.

7.2. Log: accesso, lettura e gestione

Quello che abbiamo costruito nella prima parte di questo manuale posiziona i file di log nella directory

```
/var/log/apache2/
```

Si tratta della posizione default per tutte le distribuzioni Debian e derivate come la Ubuntu. L'altro grande filone Linux (le Red Hat e derivate) mettono i log di default nella directory

```
/var/log/httpd/
```

Se abbiamo qualche dubbio è sufficiente cercare le parole chiave `ErrorLog` e `CustomLog` nella configurazione di Apache. Nel nostro server, al momento, è definito il solo dominio default il cui file di configurazione è `/etc/apache2/sites-enabled/000-default.conf`.

A questo punto è opportuno anticipare l'argomento di "Let's Encrypt" e l'utility `certbot`. Si tratta della soluzione che crea i certificati SSL. L'utility crea un file di configurazione Apache per ogni dominio che mettiamo nel nostro server.

Ipotesizzando di certificare con `certbot` il nostro dominio `www.example.com` troveremo il file di configurazione `/etc/apache2/sites-enabled/001-example.com-le-ssl.conf` in Apache.

Questa configurazione automatica prevede le seguenti due directory:

```
/var/www/digitaldsb.it/html  
/var/www/digitaldsb.it/log
```

Il significato è piuttosto immediato:

- `html` : contiene le pagine dei siti ed i file dei CMS per il dominio `example.com` ;
- `log` : contiene i file dei log per il dominio `example.com` .

Web server e siti web. Manuale pratico in stile How To 1

Se avessimo certificato il dominio `www.pippopluto.it` le directory previste dalla configurazione automatica di `certbot` sarebbero:

```
/var/www/pippopluto.it/html  
/var/www/pippopluto.it/log
```

È arrivato il momento di aprire un file di log.

Useremo lo strumento `less`, storico tool presente in tutti i sistemi `*nix`. Si tratta di un programmino che mostra su un terminale il contenuto di un file di testo, senza modificarlo e offrendo alcune funzioni aggiuntive a corredo, come la ricerca, il salto a un punto preciso, ecc... una volta invocato per uscirne basta premere il tasto Q (=quit).

Invochiamolo:

```
less /var/log/apache2/access.log
```

Vedremo qualcosa di simile a quanto segue:

```
10.10.10.1 - - [25/May/2024:06:15:18 +0000] "GET /joomla HTTP/1.1" 200 3120 "-"  
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/125.0.0.0 Safari/537.36"  
10.10.10.1 - - [25/May/2024:08:12:01 +0000] "POST  
/joomla/index.php/component/ajax/?format=json HTTP/1.1" 200 1032  
"http://www.example.com/joomla/" "Mozilla/5.0 (X11; Linux x86_64)  
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36"  
10.10.10.1 - - [25/May/2024:08:26:01 +0000] "POST  
/joomla/index.php/component/ajax/?format=json HTTP/1.1" 200 606  
"http://www.example.com/joomla/" "Mozilla/5.0 (X11; Linux x86_64)  
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36"
```

Facciamo la stessa operazione con il file `error.log`.

Apriamolo:

```
less /var/log/apache2/error.log
```

Vedremo qualcosa di simile a quanto segue:

```
[Sat May 25 06:08:58.551027 2024] [mpm_prefork:notice] [pid 830] AH00163:  
Apache/2.4.52 (Ubuntu) configured -- resuming normal operations  
[Sat May 25 06:08:58.551084 2024] [core:notice] [pid 830] AH00094: Command line:  
'/usr/sbin/apache2'  
[Sat May 25 06:27:08.162712 2024] [mpm_prefork:notice] [pid 830] AH00170: caught  
SIGWINCH, shutting down gracefully
```

Anche senza spiegazioni qualcosa risulterà *parlante* per tutti, molte altre informazioni, invece, saranno piuttosto criptiche. Evidente è il fatto che si tratta di stringhe ricche di informazioni: come possiamo rendere tutto questo patrimonio informativo immediato e facile?

7.3. Log: tool di analisi

I tool di analisi log mirano a offrire una soluzione al nostro ultimo quesito: come rendere i log immediati e facili.

Una tradizionale soluzione è **Awstats**. Si tratta di un software che viene fatto girare tipicamente una volta al giorno, legge tutti i log e li trasforma in comode pagine HTML con grafici, numerosità, indici, geolocalizzazioni, ecc...

L'applicazione è adatta per piccoli server, con poco traffico e pochi accessi. L'elaborazione che fa Awstats impiega tempo, richiede calcolo e una buona quantità di RAM. Inoltre è in grado di lavorare solo su raccolte di dati e non permette di lavorare in tempo reale. Insomma: Awstats è una tecnica datata e abbandonata.

Una potente ed elegante soluzione è **ELK** (<https://www.elastic.co>): usa i tre programmi Elasticsearch, Logstash e Kibana. Lavora in tempo reale. Permette di analizzare tutti log. Una volta avviato permette ricerche avanzate nei log, permette di salvare query, evidenziare anomalie e molto, molto altro. Pur essendo una soluzione completamente open source (non ci sono costi di licenza, né limitazioni nelle funzioni) è una soluzione che richiede esperienza e competenza per essere correttamente configurata e attivata. Inoltre necessita di periodiche manutenzioni per controllare la crescita della base dati interna.

Questa complessità e qualche altro motivo strettamente tecnico ha portato a sviluppare soluzioni più semplici, ma altrettanto potenti. **Grafana** (<https://grafana.com>) è un progetto nato nel 2014, integra tecnologie più recenti e che permettono migliori performance e integra nativamente il supporto ai requisiti più avanzati (analisi big data, fonti eterogenee, funzionamento cluster, ecc...). Accanto a tutte queste meraviglie (ed altre ancora) è necessaria competenza ed un po' di esperienza. È un prodotto ancora in crescita.

Un ottimo e semplice prodotto che unisce potenza e semplicità è **Matomo** (<https://matomo.org>). Scritto in PHP è concepito come un prodotto che legge gli accessi direttamente dal sito sotto osservazione, tramite un piccolo script in JavaScript da inserire nelle pagine. L'installazione è del tutto analogo all'installazione di Joomla o WordPress. Per inserire lo script nei CMS ci sono plugin per Joomla e per WordPress che rendono semplice l'operazione. Si tratta, sicuramente, del miglior compromesso possibile per avere un sistema open source con servizi di livello professionale e compliant con le normative europee ed italiane.

Infine è noto il software **Analytics** di Google. Molti lo usano per l'assenza di costi in diverse situazioni e per la facilità di implementazione. Funziona sulla stessa logica di Matomo, ma configurazioni, grafici, post elaborazione è tutto dentro Google. Sovranità dei dati (come principio e come reale possesso dei bit) non è possibile. Anche la compliance con le norme europee e italiane va analizzata con attenzione.

In conclusione se Matomo è il miglior compromesso e la scelta preferibile nello scenario affrontato in questo manuale non mancano alternative e prodotti con capacità superiori.

8. Provider

Passiamo al capitolo Provider. Iniziamo spiegando cosa sono e cosa offrono. Vediamo di seguito la procedura da seguire per acquistare un servizio di web hosting con dominio. Chiuderemo con le indicazioni per l'accesso via SFTP-SSH per il trasferimento dei file attraverso internet.

8.1. Provider: premessa

Prima di addentrarci nel capitolo è utile fare delle premesse. Infatti i provider hanno tutti come scopo la fornitura di servizi internet, ma non esiste un unico standard su come vengono venduti ai clienti e concessa la gestione ai clienti:

- i servizi offerti dai provider sono servizi professionali erogati dietro pagamenti;
- il cliente accede alla vendita dopo aver creato un account;
- tutto il ciclo è elettronico e smaterializzato, dalla creazione dell'account, la vendita del servizio, il pagamento e la successiva gestione;
- i prezzi possono essere molto diversi, ma l'aspetto cruciale per la scelta del provider è l'assistenza durante la vita del contratto.

NB: l'assistenza è normalmente offerta tramite una knowledge base ed un sistema di ticketing;

- i prodotti venduti dai provider sono noleggi. Al termine del contratto può esserci un tempo di sicurezza, poi viene tutto cancellato automaticamente;
- i prodotti tipici offerti dai provider sono: domini internet, email, web hosting, server online (virtuali, condivisi o dedicati), cloud, PEC/REM, certificati SSL/TLS, firme elettroniche e marche temporali. Non tutti offrono tutto, alcuni offrono anche altri prodotti-servizi aggiuntivi;
- l'acquisto di uno spazio web (=web hosting) è un pacchetto composto dall'acquisto di un dominio internet, un servizio email e un servizio di web hosting.

8.2. Provider: cosa sono

I Provider sono aziende che realizzano servizi e prodotti per internet e li rivendono.

Normalmente non creano il sito web, ma offrono tutto il servizio tecnico per far funzionare il sito. Abituamente è la web agency a creare, pubblicare e gestire il sito. È prassi comune che la web agency si interpone tra il cliente ed il provider. In questo modo il cliente fa un solo pagamento comprensivo di tutto e non ha visibilità di quanto avviene in backend.

Diversi provider cercano un filo diretto con i clienti retail e diversi offrono servizi di installazione e preconfigurazione automatizzata di siti web lasciando poi al cliente il solo onere di inserire i contenuti con qualche foto d'impatto.

Se da una parte il ruolo di mercato dei provider è chiaro, il mercato reale offre una parziale sovrapposizione dei ruoli. Pertanto il cliente al dettaglio vede offerte che si sovrappongono e vede un'ampia area di scelte che vanno da un "faccio tutto io" a un completo "fa tutto l'agenzia".

Da dire che la territorialità del provider è un concetto poco pertinente a livello pratico, ma è un distinguo importante per almeno due motivi:

- **digital act:** sostanzialmente il mondo è diviso in regioni con normative diverse. Pertanto un cliente deve scegliere un provider con data center nella regione di domicilio legale:
- **latenza:** per quanto i bit siano veloci fa differenza avere il data center a qualche centinaio di chilometri oppure oltre oceano. Il problema è rilevante al punto che esistono provider specializzati (detti CDN, Content Delivery Network) che creano una rete sopra internet per minimizzare i problemi quando è eccessiva la distanza tra il navigatore internet e il data center.

Non necessariamente i provider sono aziende di grandi dimensioni, anche se queste stanno diventando sempre più potenti togliendo di fatto la possibilità di sopravvivere ai più piccoli. Esistono anche aziende di dimensioni contenute o piccole con caratteristiche competitive.

Per completare la panoramica, l'Italia ha diversi operatori grandi e piccoli. Alcuni totalmente nostrani, altri che sono parte di aziende di grandi dimensioni. A titolo di orientamento, non certo esaustivo o indicativo, ecco un elenco di provider in ordine alfabetico:

- Aruba, <https://www.aruba.it>
- AWS – Amazon Web Services, <https://aws.amazon.com>
- EticoWeb, <http://www.eticoweb.it>
- FlameNetworks, <https://www.flamenetworks.com>
- Host.it, <https://host.it>

Web server e siti web. Manuale pratico in stile How To 1

- HostingSolutions, <https://www.hostingsolutions.it>
- KeliWeb, <https://www.keliweb.it>
- NetSons, <https://www.netsons.com>
- OVHcloud, <https://www.ovhcloud.com>
- Register.it, <https://www.register.it>
- SeeWeb, <https://www.seeweb.it>
- ServerPlan, <https://www.serverplan.com>
- ShellRent, <https://www.shellrent.com>
- SiteGround, <https://it.siteground.com>
- SupportHost, <https://supporthost.com>
- TopHost, <https://www.tophost.it>
- Utixo, <https://utixo.net>
- VirtualHosting, <https://www.VirtualHosting.com>
- WebHosting, <https://webhosting.it>

8.3. Provider: acquisto di un dominio e di un web-hosting

Per realizzare quanto illustrato nelle pagine precedenti abbiamo bisogno di acquistare un servizio di **server privato** (in gergo un **VPS**, Virtual Private Server).

Se si desidera saltare la parte didattica dei capitoli precedenti, possiamo comprare direttamente un servizio di web hosting per installare il nostro CMS.

Mentre con il primo servizio dobbiamo farci carico di tutte le operazioni descritte in precedenza, con il secondo ci troveremo tutto fatto e ci dobbiamo preoccupare unicamente di installare il CMS e caricare i contenuti.

Teniamo presente che la prima scelta ci permette di personalizzare profondamente il server adattandolo ad ogni situazione, anche ad applicativi web particolari a fronte di un costo aggiuntivo di conoscenze ed esperienza dell'amministratore. La seconda scelta ci offre una soluzione pronta all'uso, chiavi in mano, ma sarà una soluzione molto ingessata che non può adattarsi per applicativi particolari (come il log analysis).

Premesso questo ci servirà:

- creare un account sul provider;
- acquistare un dominio internet;
- acquistare un VPS o un web hosting;

Web server e siti web. Manuale pratico in stile How To 1

- acquistare un servizio email.

Come detto in precedenza non esiste uno standard unico per tutti i provider. Pertanto non è possibile riprodurre delle schermate che saranno uguali su tutti i provider, però le procedure sono le stesse per tutte le aziende.

Teniamo presente che l'acquisto di domini internet non è un vero acquisto, ma un noleggio. Normalmente è per un anno. Se intendiamo mantenere il dominio quando sta per scadere è importante rinnovare il possesso almeno un mese prima della scadenza. Se non rinnoviamo l'acquisto e se non lo facciamo entro i tempi giusti potremmo perderlo per sempre. In rete vale la regola che vince chi arriva primo.

In merito al web hosting controlliamo che sia un servizio LAMP, ovvero con Linux, Apache, MySQL o MariaDB e PHP. Non tutti i provider nel web hosting offrono questi prodotti.

Procediamo descrivendo le azioni da eseguire:

- accediamo alla home del provider scelto;
- creiamoci un account: servirà per accedere, per ordinare i prodotti, pagarli ed essere contattati dall'azienda (normalmente per ricevere le fatture e i dati di accesso).

Nota: durante questa operazione vengono chiesti anche i dati che serviranno per la fatturazione e per il pagamento. Normalmente a fine operazione riceveremo un'email, sull'indirizzo indicato, con la conferma della creazione dell'account e i dati per accedere al portale di gestione dei propri prodotti, del rinnovo, pagamenti e fatture;

- passiamo ad acquistare il dominio web.

NB 1: non possiamo acquistare il dominio example.com . Si tratta di un dominio riservato alla didattica e agli esempi, non esiste e non può essere venduto da nessuno.

NB 2: la scelta del dominio dovrebbe essere qualcosa di studiato perché sia significativo, facile da ricordare e corto. Bisogna arrendersi, comunque, a quello che è disponibile. Non sempre siamo i primi a pensare a un preciso dominio.

NB 3: l'acquisto del dominio porta con sé il servizio di gestione in proprio del DNS. Questo significa che terminato il pagamento ci verranno fornite le credenziali per accedere ad un pannello che ci permette di creare nomi FQDN di 3°, 4°, ecc... livello come www.miosito.it , srv.miosito.it , eshop.miosito.it , ecc....

NB 4: ai fini didattici immaginiamo che sia disponibile il dominio miosito.it .

Web server e siti web. Manuale pratico in stile How To 1

Accediamo alla home del provider che abbiamo scelto. Normalmente propongo una pagina simile a Google per inserire il nome del dominio che vorremmo comprare. Inseriamo il nome, confermiamo e una pagina ci dirà se è disponibile e ci proporrà di acquistare anche altri domini. Ad esempio con miosito.it ci proporrà di comprare anche miosito.eu e miosito.com .

Facciamo la nostra scelta e procediamo a confermare l'ordine.

- Passiamo ad acquistare il servizio di VPS o di web hosting.

Normalmente il provider mette queste offerte in uno dei menu principali: individuato il nostro servizio accediamo per fare l'ordine di uno dei due (non dobbiamo prenderli entrambi almeno che non sia una scelta voluta):

- **VPS:** una volta entrati normalmente i provider sottopongono subito dei pacchetti preconfezionati. Un dimensionamento libero, se c'è, è una pagina da cercare.

Un dimensionamento adatto per piccole attività potrebbe essere un 2 core, 4GB di RAM, 80GB di HDD. Molto spesso il traffico sulla scheda di rete è senza limiti e viene dato 1 IP pubblico senza costi aggiuntivi.

Trovata la dimensione che fa al nostro caso confermiamo l'ordine.

- **Web Hosting:** una volta entrati normalmente i provider sottopongono subito dei pacchetti preconfezionati. Individuiamo il pacchetto che offre un ambiente LAMP. Molti automaticamente mettono anche il servizio email, ma limitato o nello spazio o nel numero di caselle che si possono creare.

Trovato il pacchetto che fa al nostro caso confermiamo l'ordine.

NB: in caso di dubbio è bene contattare l'area vendite del provider. Normalmente hanno una form aperta o un'email di contatto visibile nell'area clienti;

- passiamo a ordinare il servizio email se non incluso nel pacchetto di web hosting. Le indicazioni sono come per i passi sopra.
- Confermiamo gli ordini e passiamo al checkout, paghiamo il costo totale e attendiamo l'attivazione di quanto acquistato.

Normalmente l'attivazione viene fatta a valle del ricevimento del pagamento. Per acquisti personali pagati con un servizio elettronico (carte di credito, carte di debito, bonifici istantanei, ecc...) l'attesa può essere di qualche minuto. Per acquisti aziendali l'acquisto può richiedere un giorno o poco più. Per le aziende normalmente i provider hanno dei pacchetti speciali con servizi a corpo e forme di pagamento massive.

Normalmente, al termine del processo, riceveremo una o più mail che ci comunicano la buona chiusura degli ordini, l'attesa di ricevimento del pagamento, un'email con le credenziali di accesso e con una manualistica essenziale (o l'URL delle guide con le istruzioni per operare).

8.4. Provider: usare SFTP e SSH

Una volta in possesso del servizio di un VPS dovremo fare ancora due semplici operazioni:

1. accedere al pannello di controllo fornito dal provider, scegliere la distribuzione Linux che desideriamo usare e attendere il deploy. Abitualmente è un'operazione che impiega qualche minuto.

Al termine di questo ci viene data una password di accesso tramite un terminale disponibile su una pagina web del sistema di controllo offerto dal provider: accediamo, aggiorniamo il nostro nuovo VPS e, se non installato di default, installiamo OpenSSH (ovvero il server SSH) e creiamo un utente con diritti sudo;

2. a questo punto possiamo usare il collegamento ssh dalla nostra work station, soluzione decisamente più agevole rispetto all'operare da una pagina web.

Se usiamo una workstation Linux, Mac o *nix il tool ssh è di serie. Se operiamo da un PC Windows possiamo usare PuTTY (<https://putty.org>) o MobaXterm (<https://mobaxterm.mobatek.net>).

Il protocollo SSH offre non solo la funzione di terminale, ma permette anche il forwarding X, il trasporto di file tramite SFTP e il montaggio remoto di hard disk e/o directory come se fossero un disco locale:

- **terminale**: si tratta delle funzioni usate nei primi capitoli di questo manuale. In altre parole un schermo alfanumerico collegato, tramite la rete, al server remoto;
- **Forwarding X**: si tratta di un'antica funzione del mondo Unix in combinazione con X (ovvero il sistema grafico nel mondo *nix). In pratica ci permette di installare nel server remoto un'applicazione grafica (ad esempio Firefox), lanciarlo sul server remoto ed avere le finestre sulla nostra workstation. La soluzione funziona, in diversi casi è molto utile, ma per un meccanismo di sicurezza di SSH è lenta indipendentemente dalla velocità della nostra connessione internet;
- **SFTP**: si tratta di un protocollo per spostare file attraverso una connessione SSH. È in tutto uguale all'antico FTP, ma a differenza di questo è un sistema sicuro.

Per usarlo abbiamo di serie il supporto nelle applicazioni native per i sistemi *nix. Se usiamo Windows dobbiamo aggiungere un programma come il noto FileZilla (<https://filezilla-project.org>) o WinSFTP (<https://winscp.net>);

- **Mount**: il protocollo SSH può trasportare un disco come se fosse una sorta di prolunga USB per connettere un pendrive al nostro PC.

Il servizio è decisamente comodo, sicuro e veloce.

A fattore comune il protocollo SSH è una forma di connessione criptata end-to-end, quindi molto sicura e affidabile.

Web server e siti web. Manuale pratico in stile How To 1

Per sfruttare le funzioni sopra indicate rimandiamo alle risorse e alle guide disponibili su internet. Sarà sufficiente un paio di ricerche per trovare il manuale che fa al caso nostro.

Probabilmente sarà chiaro per ciascuno che le funzionalità native offerte da SSH permettono di gestire un VPS quasi come se fossimo sulla console fisica del server.

Nel caso avessimo acquistato un servizio di web hosting tradizionale i provider non offrono il protocollo SSH, ma offrono un pannello di controllo dell'ambiente LAMP e una pagina per gestire file e directory del nostro spazio. Non mancano provider che offrono anche un accesso SSH securizzato come una sandbox.

I pannelli di gestione dell'ambiente LAMP si sono standardizzati di fatto su cPanel (<https://www.cpanel.net>), Plesk (<https://www.plesk.com>) e DirectAdmin (<https://www.directadmin.com>).

Per la gestione dei file e delle directory non c'è un'interfaccia standard di fatto (anche se sono tutti intuitivi, facili e immediati). Ma una gestione totale tramite la pagina web è poco agevole.

In alternativa i provider offrono una connessione di tipo FTP o SFTP:

- **FTP**: si tratta di un antico sistema per trasferire file e directory da e verso un server remoto. Questo protocollo di fatto è dismesso e fortemente scoraggiato dove per qualche motivo è ancora richiesto (o in uso). Infatti è intrinsecamente insicuro (tutto viaggia in trasparente) e ha alcuni limiti-problemi tecnici;
- **SFTP**: è la stessa cosa dell'FTP, ma sicura e senza la serie di limiti e problemi che affliggevano l'FTP.

Per usare l'SFTP è semplice:

- se abbiamo una workstation *nix abbiamo tutto di serie;
- se abbiamo un PC con Windows dobbiamo aggiungere un programma come, ad esempio, FileZilla o WinSFTP.

Per creare la connessione basta inserire l'URL del nostro webhost, la username che abbiamo creato e la password che abbiamo disposto.

Diversi provider hanno ulteriormente alzato l'asticella di sicurezza e per creare la connessione SFTP non si usano più username e password, ma ci danno una chiave di sicurezza che sostituisce l'account. La chiave di sicurezza è una lunga, lunga sequenza di lettere e numeri apparentemente casuali.

Anche per questo aspetto rimando alle guide online per l'uso pratico:

- i provider offrono direttamente pagine loro di istruzioni con schermate ed esempi pratici;

- cercando in rete troviamo tutto, anche con le indicazioni specifiche per il provider che abbiamo scelto.

In fine teniamo presente che abbiamo acquistato un servizio. L'acquisto prevede anche il supporto di base da parte del servizio clienti del provider. Quindi è buona prassi chiedere supporto se riteniamo di averne bisogno.

9. Certificati SSL

9.1. SSL: premessa

L'avvento di internet avvenne sull'onda di una grande fiducia nell'onestà e bontà di tutti e di tutto. Successivamente ci si è accorti (dolorosamente) che non tutti sono buoni. Inoltre si è presa consapevolezza che le grandi potenzialità offerte dalla rete possono non essere positive.

Da qui è nato il concetto di connessioni sicure e criptate. Più recentemente questo aspetto di sicurezza e di privacy ha fatto un ulteriore passo in avanti introducendo il concetto e la pratica dello "zero trust".

In merito alle connessioni internet fino al 2014 tutti i siti offrivano la sola connessione HTTP: leggera, fa bene il suo lavoro, ma è totalmente trasparente e di conseguenza pericolosa. Solo i siti di banche, negozi online, ecc... offrivano connessioni HTTPS.

L'HTTPS è la stessa cosa dell'HTTP, ma tutto quello che transita sulla rete viene criptato. Il primo approccio, fatto tramite una tecnica nota come SSL, è stato aggiornato nel 2020, e si usa attualmente una tecnica nota come TLS.

Dal 2020 le connessioni HTTP non sono più accettate per accordo tecnico tra le aziende e per regolamenti di sicurezza ICT. Funzionano e si possono usare solo a scopo didattico e all'interno di reti locali private a scopo di laboratorio.

Da sottolineare, per completezza e per correttezza, che in questa evoluzione non ci sono stati obblighi legali, ma tutto è avvenuto come uno sviluppo per progressive scelte aziendali (anche di concerto tra più aziende) che hanno spinto rendendo di fatto standard l'HTTPS e il TLS. Un'evoluzione decisamente positiva.

Indipendentemente dall'evoluzione il principio e la meccanica non sono cambiati. Questo sistema di sicurezza e criptazione funziona sul seguente meccanismo di principio:

- il server ha al suo interno la metà di una chiave digitale;
- il cliente (ovvero il programma di web browsing) ha al suo interno l'altra metà della chiave digitale;
- al momento della connessione entra in gioco un terzo attore che sta in internet. Questo attore (detto in gergo CA, Certification Authority) viene interpellato e verifica le chiavi e le credenziali di tutti gli attori;

Web server e siti web. Manuale pratico in stile How To 1

- dopo una invisibile e veloce prima fase di verifica (detta handshake), viene creato un tunnel criptato di comunicazione e si naviga in internet in modo sicuro e confidenziale.

Se qualcosa non collima il browser ci mostra una pagina di allarme e rimette all'utente la decisione di procedere o meno.

Se non collima nulla la navigazione a quell'URL non sarà possibile.

In questo meccanismo per stabili connessioni sicure sono fondamentali tre elementi:

- **FQDN (Fully Qualified Domain Name)**: il nome internet del server (ovvero l'URL) deve esistere ed essere pubblica (ovvero registrata sui DNS mondiali);
- **certificato Server**: il server deve avere al suo interno un file di certificato valido e in corso di validità;
- **certificato client**: il browser deve avere al suo interno un file di certificato valido e in corso di validità.

Da questa premessa possiamo capire l'importanza dell'HTTPS e del suo impatto bloccante e di cosa abbiamo bisogno per poter avere un certificato TLS per il nostro server.

9.2. SSL e TLS; cosa sono

Nella premessa sopra abbiamo detto che TLS (Transport Layer Security) è, di fatto, l'evoluzione di SSL.

Si tratta di una tecnologia per creare connessioni criptate, sicure e garantite:

- **criptate**: il traffico viene codificato. Se qualcuno intercetta la comunicazione non può decodificarla;
- **integrità dei dati**: il protocollo prevede un sistema di controllo dei dati individuando e correggendo eventuali errori di trasmissione e impedendo l'interpolazione;
- **garantite**: si tratta di un'estensione del concetto di sicurezza. Tramite la CA l'utente riceve conferma che il server è veramente chi dice di essere.

Inoltre il meccanismo di generazione e gestione dei certificati prevede anche uno storico dei certificati scaduti e/o revocati. Pertanto è possibile verificare a chi apparteneva un certificato scaduto e verificare se un certificato in corso di validità è stato revocato.

Dietro le quinte c'è stato anche un importante avanzamento di unificazione e uniformazione. Lo stesso protocollo che ci permette di navigare sicuri è implementato nelle email, nelle PEC, nelle REM e nelle nostre connessioni SSH.

In pratica, oltre agli aspetti tecnici che ci permettono di usare una sola tecnologia, potremmo dotare il nostro server privato di un unico certificato wildcard che garantirà tutti i servizi.

9.3. SSL: acquisto da un provider

Nel caso di acquisto di un web hosting il pacchetto prevede anche il certificato TLS per la connessione HTTPS.

Se abbiamo comprato anche il servizio email il provider potrebbe emettere un unico certificato che garantisce e securizza sia il web server che il sistema di email.

È importante che durante la creazione dell'account sul provider, al momento della compilazione delle form con i dati dell'identità e i dati fiscali, abbiamo inserito dati esatti e corretti perché alcuni andranno nei certificati indicando chi è il proprietario e chi è il responsabile. Se i dati non sono corretti potremmo avere un errore di attendibilità del certificato anche se è tecnicamente perfetto.

Nel caso di acquisto di un VPS il provider non genera nessun certificato. Dobbiamo provvedere noi ad acquistare uno.

Abbiamo due scelte possibili:

- acquistarne un certificato da un provider;
- prenderne uno gratuito da Let's Encrypt.

Nel primo caso, accanto al pagamento, abbiamo l'onere di dover generare un primo certificato che verrà poi firmato e garantito dalla CA. inoltre questa forma di acquisto aggiunge una durata maggiore del certificato e la possibilità di avere certificati più estensivi (quindi soluzioni wildcard, chiavi per i supporti di memoria, chiavi per sistemi di identificazione come le tessere magnetiche e/o con chip, ecc..).

Nel caso di acquisto teniamo presente che esistono provider specializzati in questi servizi o provider generici che offrono certificati, web hosting, cloud, ecc...

La procedura di acquisto è analoga agli acquisti che abbiamo fatto prima:

- accediamo al sito del nostro provider e autenticiamoci;
- individuiamo la pagina per l'acquisto di un certificato;
- comparirà una pagina che chiede una serie di dati o di fornire un certificato non firmato.

NB: ricordiamoci di mettere dati veritieri e corretti perché verranno, in parte, inclusi nel certificato e sarà evidente a tutti se sono fasulli o inesatti;

- dopo qualche istante il provider ci offrirà una pagina con la conferma della buona riuscita dell'operazione e la possibilità di scaricare il certificato. Dovremmo ricevere anche un'email che riporta la riuscita (o meno) della creazione del certificato, alcune istruzioni e informazioni veloci e dei link per accedere a manualistica e istruzioni offerte dal provider stesso; in allegato dovremmo trovare il file del certificato che sarà da scaricare, installare nel server e conservare una copia in modo attento e sicuro.

A questo punto dobbiamo mettere il file del certificato in un punto preciso del server e apportare una configurazione al nostro web server Apache.

Passiamo a dare continuità al lavoro che abbiamo fatto all'inizio di questa guida installando e avviando il web server Apache. Quanto abbiamo creato è un server (fisico o virtuale non c'è differenza) ed il nostro provider non fornisce nel pacchetto un certificato. Possiamo scegliere Let's Encrypt che offre certificati ad uso gratuito.

9.4. SSL: Let's Encrypt

Let's Encrypt è una società nata nel 2015 dall'alleanza di alcune aziende del web che hanno voluto sostenere l'adozione universale del protocollo HTTPS per garantire a tutti un servizio internet migliore, più sicuro e più riservato.

A differenza di altre società precedenti, come la StartSSL, la Let's Encrypt ha stretto accordi con chi sviluppa software internet per inserire, all'atto della creazione dei programmi, il certificato pubblico. Pertanto chi usa i servizi offerti dalla Let's Encrypt ha la stessa esperienza d'uso che si ha acquistando un certificato a pagamento.

Come accennato sopra i certificati della Let's Encrypt hanno delle limitazioni che non li rendono adatti in alcuni casi d'uso, oltre ad avere una durata predeterminata di 3 mesi (**NB**: la breve durata è un vantaggio di sicurezza, anche se una complicazione in più per i gestori).

Accanto a questi interessanti benefici sono nati due script di pubblico dominio che rendono facile e agevole l'adozione di un certificato. Addirittura configurano automaticamente il web server Apache o Nginx.

I due script citati sono ACME (<https://github.com/acmesh-official/acme.sh>) e il più raccomandato CertBot (<https://certbot.eff.org>) .

Torniamo al nostro server che abbiamo abbandonato al capitolo 7, e procediamo aggiungendo un certificato per il nostro dominio example.com .

Prima di procedere dobbiamo avere chiare due idee:

1. in questo manuale usiamo dati di fantasia e un dominio universalmente bloccato. Pertanto le istruzioni seguenti riportano il processo corretto, ma con dati che non daranno mai un risultato di successo;
2. prima di iniziare la procedura dobbiamo aver già registrato e attivato il dominio.

Web server e siti web. Manuale pratico in stile How To 1

Visto che nel nostro server abbiamo creato un'installazione sfruttando la configurazione default dobbiamo creare le directory e la configurazione specifica per il dominio example.com.

9.4.1. Let's Encrypt: configuriamo example.com

Allo scopo didattico di questa parte non procederemo a migrare PhpMyAdmin, Joomla e WordPress.

1. Torniamo a collegarci al nostro server remoto

```
ssh weadmin@www.example.com
```

2. creiamo le directory per il nostro dominio example.com

```
sudo mkdir -p /var/www/example.com/{html,log}
```

3. creiamolo il file index.html per avere una pagina di cortesia

```
sudo nano /var/www/example.com/html/index.html
```

4. e popoliamolo come segue

```
<!DOCTYPE html>
<html>
<body>
<h1>SITO IN COSTRUZIONE</h1>
<p>Ci scusiamo per disagio. Saremo a breve nuovamente online</p>
</body>
</html>
```

5. correggiamo i permessi

```
sudo chown www-data:www-data -R /var/www/example.com/
```

6. creiamo il file di configurazione example.com.conf per Apache

```
sudo nano /etc/apache2/sites-available/example.com.conf
```

7. e popoliamo come segue

```
<VirtualHost *:80>
    ServerAdmin webmaste@example.com
    ServerName www.example.com
    DocumentRoot /var/www/example.com/html

    ErrorLog /var/www/example.com/log/error.log
    CustomLog /var/www/example.com/log/access.log combined

    <Directory "/var/www/example.com/html">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

8. attiviamo la configurazione

```
sudo a2ensite example.com
```

tutto è pronto per procedere alla creazione del certificato con `certbot`.

9.4.2. Let's Encrypt: creiamo il certificato con certbot

La prima volta che si usa `certbot` verrà chiesto l'indirizzo email per identificare l'utente responsabile. Facciamo attenzione a fornire un identificativo reale che verrà inserito nei certificati e servirà per identificare il responsabile. Se certifichiamo più server possiamo sempre usare la stessa email se siamo i responsabili (reali) dei vari domini e server.

Ai fini didattici ipotizziamo che l'email è `webmaster@example.com`.

Iniziamo la creazione del certificato. Useremo CertBot che installerà automaticamente anche un demone che gestisce automaticamente il rinnovo del certificato sollevandoci anche da questa noia:

- dalla nostra console accediamo al nostro web server via SSH;
- installiamo lo script (è di default nei repo delle principali distribuzioni Linux)

```
sudo apt install certbot python3-certbot-apache
```

- creiamo il certificato

```
sudo certbot --apache --agree-tos --redirect --hsts --staple-ocsp --email  
webmaster@example.com -d www.example.com
```

- dopo alcuni istanti otterremo un output simile al seguente

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log  
Requesting a certificate for www.example.com  
  
Successfully received certificate.  
Certificate is saved at: /etc/letsencrypt/live/www.example.com/fullchain.pem  
Key is saved at: /etc/letsencrypt/live/www.example.com/privkey.pem  
This certificate expires on 2024-09-29.  
These files will be updated when the certificate renews.  
Certbot has set up a scheduled task to automatically renew this certificate in  
the background.  
  
Deploying certificate  
Some rewrite rules copied from /etc/httpd/sites-enabled/example.com.080.conf  
were disabled in the VirtualHost for your HTTPS site located at  
/etc/httpd/sites-available/example.com.080-le-ssl.conf because they have the  
potential to create redirection loops.  
Successfully deployed certificate for www.example.com to /etc/httpd/sites-  
available/example.com.080-le-ssl.conf  
Congratulations! You have successfully enabled HTTPS on https://www.example.com  
  
-----  
If you like Certbot, please consider supporting our work by:  
* Donating to ISRG Let's Encrypt: https://letsencrypt.org/donate  
* Donating to EFF: https://eff.org/donate-le  
-----
```

NB: se sbagliamo qualcosa, ma completiamo l'operazione, possiamo rifare la procedura per un massimo di 3 tentativi sbagliati. Dopo si verrà bloccati per 3 mesi.

Web server e siti web. Manuale pratico in stile How To 1

A questo punto abbiamo creato il certificato. L'installazione e la creazione del certificato aggiungono un processo automatico di rinnovo. Pertanto non dobbiamo preoccuparci più di nulla. CertBot ha anche installato una regola che automaticamente reindirizza il traffico HTTP sul HTTPS.

A questo punto ci serve solo far ricaricare le configurazioni ad Apache

```
sudo systemctl reload apache
```

Nota finale: `certbot` è uno script che viene migliorato rapidamente. È possibile che le sintassi e gli output siano diversi nel momento in cui userete il tool.

10. Compendio

10.1. GDPR, crittazione e sicurezza

Nelle precedenti pagine abbiamo affrontato gli aspetti tecnici ed esecutivi per realizzare un web server e pubblicarlo. In realtà questo non è sufficiente per avere un sito completamente compliant soprattutto se introduciamo servizi come iscrizioni, automazioni, news letter, eshop, ecc...

Spendiamo qualche parola circa i tre elementi irrinunciabili: GDPR, crittazione e sicurezza.

10.1.1. GDPR

Si tratta di un regolamento europeo stringente ed ineludibile. Dobbiamo o studiarlo o chiedere una consulenza ad un esperto per gli eventuali adeguamenti.

Ipotizzando che il nostro web server esponga contenuti, senza servizi e usando solo cookies tecnici che restano nel sito web ci si adegua con poca fatica:

- studiamo la situazione e stendiamo il nostro piano GDPR;
- pubblichiamo nel sito un articolo corretto con le giuste informazioni circa finalità, trattamento e che non ci sono cookies di terze parti;
- aggiungiamo un sistema al nostro sito che notifichi al navigatore che accede la prima volta tutto questo e chieda la conferma di lettura e consapevolezza.

È oltre gli obiettivi di questo manuale entrare nel merito in modo esaustivo. Pertanto dobbiamo per forza rimandare i lettori ad una consulenza o all'assoldamento di un esperto.

10.1.2. Crittazione

Il GDPR e la sicurezza introducono una serie di attenzioni obbligatorie. Uno degli aspetti è la crittazione anche dei file per evitare che un accesso o un'esfiltrazione di file e di dati faccia uscire informazioni sensibili.

Web server e siti web. Manuale pratico in stile How To 1

Dobbiamo distinguere tra criptazione stretta e pseudonimizzazione:

- **criptazione:** si tratta della codifica di tutto quello che viene scritto sul nostro hard disk. Normalmente è un'operazione che viene fatta alla creazione del server adottando un file system criptato.

La scelta in questo senso è molto buona, ma ha un impatto da considerare e valutare:

- ad ogni avvio dobbiamo sbloccare l'accesso al disco. Se gestiamo un server remoto è un'operazione decisamente antipatica perché dovremmo accedere alla console remota del VPS;
 - la criptazione del file system aggiunge calcolo e rallenta il nostro server ed i servizi. Anche se in ambiente *nix con l'hardware di oggi abbiamo overload trascurabili, è bene valutarne l'adozione alla luce di tutto il pacchetto.
- **Pseudonimizzazione:** si tratta di una separazione delle informazioni che non compariranno mai aggregate e che possiamo riunire solo tramite una sigla che in se non dice nulla.

Ad esempio l'anagrafica ed i dati medici di Mario Rossi li possiamo dividere su più tabelle del nostro database e identifichiamo tutte queste informazioni con la sigla di fantasia ABCD1234FFT1. Se prendiamo i dati sparsi nelle tabelle non riusciremo mai a riunificarli. Pertanto individuato il nome Mario Rossi non riusciamo a sapere se è diabetico o meno. Solo se prendiamo la ABCD1234FFT1 possiamo sapere quali sono i dati corretti correlati a Mario Rossi.

Questa tecnica comporta un po' di programmazione alla portata di ogni sviluppatore. È una buona soluzione per database, ma non aiuta, ad esempio, con i dati su file o su hard disk.

È utile considerare che in un web server coesistono diverse tecnologie, diversi servizi e lo possiamo usare a scopi diversi. Il nostro web server, ad esempio, può ospitare solo locandina e programma delle attività ricreative estive o potrebbe ospitare il sistema di gestione completa delle attività estive compresi pagamenti, iscrizioni, database dei partecipanti, ecc...

Pertanto la pseudonimizzazione viene a rispondere a bisogni di anonimizzare i dati per il portale dei servizi delle attività estive, senza dover criptare il server o dover criptare il database.

10.1.3. Sicurezza

La sicurezza è un concetto più estensivo previsto anche dalle normative GDPR.

Alcuni aspetti sono sostanzialmente obbligatori:

- attivare il firewall interno al web server;

Web server e siti web. Manuale pratico in stile How To 1

- avere una procedura di disaster recovery;
- avere una politica di backup;
- adottare solo password sicure e dove è possibile, l'accesso tramite certificato e non tramite password;
- usare esclusivamente connessioni HTTPS.

Se decidiamo di andare in produzione con servizi reali tutto questo va anche correttamente documentato e gestito.

Altro aspetto da tenere in doverosa considerazione è la sicurezza intrinseca dei portali web:

- affidiamoci a soluzioni consolidate, in uso da tempo e che hanno alle spalle un'azienda manteiner o una community consolidata come per Joomla e Word-Press;
- teniamoli costantemente aggiornati e monitoriamoli con una adeguata frequenza;
- rinunciamo a componenti aggiunti esotici di programmatori improbabili;
- sviluppo e prove facciamo sempre su un sito gemello sulla propria work station, mai online sul sito in produzione;
- non concediamo accessi privilegiati o possibilità di gestione a persone inesperte o non formate.

Ebbene: internet è come l'oceano, ma a differenza dell'oceano c'è uno squalo affamato ogni 3 metri quadrati.

10.2. Apache: Host e VHost

Installando il certificato con `certbot` abbiamo visto che Apache ha una configurazione "Virtual Host" associata al nome FQDN del nostro sito: cosa significa?

Abitualmente i web server ospitano tanti siti web e sevizi web. Per indicare ad Apache Server HTTP quale sito deve offrire al navigatore tra tutti quelli che contiene può o usare l'indirizzo IP o usare il nome FQDN.

Se usiamo l'indirizzo IP significa che il nostro server deve possedere tanti IP quanti sono i siti ospitati. Questa pratica da tempo non si usa più perché gli indirizzi IP sono limitati e sono diventati una risorsa preziosa.

La seconda soluzione è che il nostro server ha un solo indirizzo IP, ma ci sono diversi nomi FQDN che puntano a un solo indirizzo IP. Apache capisce quale sito web deve servire leggendo l'FQDN con cui viene interrogato.

I web server permettono una configurazione mista: un pool di IP con gruppi di nomi FQDN che puntano a uno degli IP assegnati al web server.

Web server e siti web. Manuale pratico in stile How To 1

Apache usa il tag `<VirtualHost>` insieme alle direttive `ServerName` e `ServerAlias` per individuare quale sito offrire tra tutti quelli presenti al suo interno.

Non è oggetto di questo manuale approfondire l'argomento, ma potendo gestire un web server abbiamo molte possibilità che ci permettono da una parte di sfruttare bene le potenzialità e dall'altra parte di costruire soluzioni utili e interessanti.

10.3. Accelerare PHP

PHP è un linguaggio popolare e largamente conosciuto. Soffre di diversi limiti. In particolare, proprio per la sua architettura di fondo, è lento.

Per mitigare questo importante limiti si sono sviluppate soluzioni diverse nel tempo.

Sono nati framework di programmazione che ottimizzano il codice riducendo al minimo i cicli di calcolo necessari.

Si è intervenuti sull'integrazione tra il web server e PHP andando a rimpiazzare il primo modulo PHP per Apache con soluzioni alternative.

Parallelamente sono sorte applicazioni di cache che riducono il riprocessamento degli script introducendo benefici a volte molto, molto importanti.

Accanto alle soluzioni cache sono venute alla luce implementazioni di database no SQL che fanno da cache. Anche queste contribuiscono ad ottenere accelerazioni molto importanti.

Spesso è una sapiente combinazione di questi elementi che dà i risultati migliori.

Appreso il concetto l'argomento è oltre gli obiettivi di questo manuale. Però offriamo di seguito un elenco parziale delle tecnologie di accelerazione di PHP prendendo in considerazione le forme di collegamento tra il web server (Apache o Nginx) e le cache più usate.

- **FastCGI**: è stata la prima mitigazione introdotta. Si basa sull'esternalizzazione dei processamenti degli script PHP e il loro riutilizzo. Quando viene chiesta la stessa elaborazione non viene rifatto il calcolo, ma viene offerto direttamente il risultato;
- **FPM**: acronimo di FastCGI Process Manager. Si tratta di un modulo che sostituisce il vecchio FastCGI. Funziona sullo stesso principio di FastCGI, ma è molto più performante e permette molta più flessibilità;
- **OPCache**: è una cache all'interno di PHP. Di norma è attiva e ha una configurazione base già impostata. Serve per accelerare tutte le operazioni di processamento degli script e funziona a livello globale;
- **APCu**: è un modulo aggiuntivo di PHP e crea una cache gestita direttamente da PHP a fattor comune, esterna al core, usata direttamente dalle applicazioni;

- **Memcached:** si tratta di un software server aggiuntivo da installare nel server. Una volta lanciato crea nella RAM una cache molto veloce e performante che viene usata dalle applicazioni come Joomla, WordPress e altre;
- **Redis:** implementa la stessa filosofia di Memcached, ma realizza l'obiettivo attraverso un forma di database no SQL molto rapido. Non si presta molto bene per una cache generalista, ma gestisce molto bene, ad esempio, la tenuta delle sessioni.

10.4. Ruby, Node.js, Go

PHP è stato tra i primi linguaggi di scripting nati per il Web. La sua facilità, le potenzialità e la sua natura open source ne hanno decretato un successo rapido e universale.

L'uso così vasto, però, ha messo ben in luce i limiti ulteriormente evidenziati dalle sfide più recenti che arrivano dalla crescita della rete, dalla progressiva sistematica digitalizzazione di tutto, dei big data e dell'AI.

Da queste provocazioni hanno preso vita altri progetti che si propongono di nascere senza alcuni difetti e risolvere nativamente le nuove sfide della rete.

Vediamo un piccolo elenco selezionato delle tecnologie emergenti a cui guarda come alternativa a PHP o come tecnologia necessaria per far girare alcune applicazioni.

10.4.1. Ruby on Rail

«Un linguaggio open source dinamico che dà particolare rilevanza alla semplicità e alla produttività, dotato di una sintassi elegante, naturale da leggere e facile da scrivere» recita la descrizione ufficiale di questo linguaggio disponibile all'URL <https://www.ruby-lang.org>.

Il suo punto di forza è la sintassi semplice e intuitiva. Adotta formalismi più recenti e più performanti come YAML e, durante la progettazione, ha preso il meglio dalle altre tecnologie analoghe.

Ruby on Rail è un framework per applicazioni web basato sul linguaggio Ruby. Pensato per costruire applicazioni web di livello aziendale realizza software veramente performanti, sicuri e in grado di gestire grandi volumi di lavoro.

10.4.2. Node.js

Si tratta di un ambiente runtime open source per eseguire codice JavaScript lato server. Il sito del progetto è <https://nodejs.org>.

Permette di sviluppare applicazioni minimizzando le conoscenze necessarie. Semplifica parte della complessità di Java, migliora le performance delle tradizionali applicazioni Java in Application Server (vedete qui di seguito), gestisce in modo eccellente scalabilità e i cicli di deploy delle applicazioni e, soprattutto, è molto veloce.

10.4.3. Go

Sempre una soluzione open source, nota anche con il nome GoLang, disponibile al sito <https://go.dev>.

Creato dalla Google ha l'obiettivo di risultare efficiente, leggibile e semplice da apprendere. Decisamente veloce, soprattutto in compilazione, lo troviamo integrato in molte applicazioni di tipo aziendale.

10.5. Java a AS

Java è una tecnologia nata nel 1991. Fu un linguaggio che introdusse innovazioni dirompenti. In particolare la possibilità di funzionare su qualsiasi computer, di operare con *veti* diverse e di essere open. Fu presto adottato dal mondo aziendale portando a sviluppare standard industriali noti come J2EE e JSR.

Non mancano le controindicazioni: dichiarato morto più volte continua a vivere, ma si trascina limiti di rilievo, come l'esigenza di risorse hardware, una gestione vecchia dell'esaurimento della RAM, log di errore chilometrici per qualsiasi tipo di problema, una sintassi abbastanza complessa e poco intuitiva.

Nel tempo sono stati sviluppati diversi framework che permettono di ridurre notevolmente il numero di righe di codice da scrivere e/o introducono funzioni di alta affidabilità con solo un paio di istruzioni.

Java lo troviamo a totale libero uso nella versione OpenJava (noto anche come OpenJDK e OpenJRE) disponibile all'URL <https://openjdk.org>. Offre fondamentalmente due versioni di se stesso:

- **JDK**: per gli sviluppatori e completo dell'ambiente per far girare le applicazioni;
- **JRE**: per l'utente comune con il solo ambiente per far girare le applicazioni.

Java permette di fare applicazioni in scripting, ma anche applicazioni binarie che girano nella tradizionale forma desktop o come app per tablet e cellulari.

In ambiente web può funzionare come script puro all'interno di una pagina HTML, o come script server-side come PHP, Ruby o Node.js, ma anche in forma di applicazione vera e propria (in gergo Portlet). In questa forma dobbiamo mettere l'applicazione dentro un Application Server (da qui in poi AS) e sarà l'AS a parlare con il web server.

Detto questo, allo scopo di un web server, vale la pena conoscere alcune caratteristiche. In particolare sapere che esistono diverse razze di Java. In secondo luogo l'architettura attraverso gli AS e, infine, la tecnologia dei reverse proxy.

10.5.1. Java: quale razza?

Java fu inventato dalla Sun. Distribuito come open source, dopo una lunga guerra di brevetti con la Microsoft, ha visto degli spin off con IBM all'inizio. Nasceva e viveva bene anche Open Java.

Web server e siti web. Manuale pratico in stile How To 1

Nel 2009 la Sun fu comprata dalla Oracle. Dopo una prima fase di grigio e di cambio licenza di Java, si è tornati ad avere una versione di libero uso targata, però, Oracle.

Pertanto al momento abbiamo, tra i più noti, Open Java, Oracle Java e IBM Java.

Questa varietà ha portato gli sviluppatori ad usare, a volte porzioni di codice esclusivo di un brand. Pertanto possiamo incappare in una applicazione che se fatta girare con Open Java si blocca perché cerca una classe Oracle.

Pertanto i Java sono tutti uguali, ma anche tutti diversi...

10.5.2. Java e gli AS

Gli AS sono dei particolari programmi che servono per far girare i programmi web scritti in Java.

Come per il linguaggio esistono diversi AS perché varie house software hanno realizzato il loro AS. Come per il linguaggio non sono tutti veramente equivalenti e non tutte le applicazioni girano in tutti gli AS.

Tra gli AS più noti c'è:

- **TomCat**, <https://tomcat.apache.org> , è sicuramente il più noto. Mantenuto dalla fondazione Apache è open source, abbastanza completo, non ha particolari requisiti per girare ed è anche dotato di un pannello web di gestione;
- **GlassFish**, <https://glassfish.org> , sviluppato in origine da Sun, ora è mantenuto dalla fondazione Apache. È rimasto open source, completo in tutti gli aspetti necessari ad un AS enterprise lo si gestisce completamente tramite un'interfaccia web;
- **Jetty**, <https://eclipse.dev/jetty> , lo possiamo definire un TomCat senza la pagina web. È concepito per ridistribuire applicazioni in pacchetti demo autoconsistenti. Leggero, veloce, open source manca, oltre al pannello web, anche di tutto lo strato enterprise;
- **WildFly**, <https://www.wildfly.org> , è il fratello open source di JBoss, un AS completo di tutto di livello industriale;
- **JBoss**, <https://www.jboss.org> , è un AS prodotto dalla Red Hat. È un piccolo mostro nel senso che è completo di tutto ed è di livello enterprise in tutti i sensi.

10.5.3. Java e i reverse proxy

Immaginiamo per un momento che il nostro CMS sia stato realizzato in java. Per farlo girare lo mettiamo dentro un AS, ma se proviamo a collegarci alla nostra URL www.example.com non vedremo nulla.

Questo accade perché gli AS pubblicano le applicazioni sulle porte 8080 e 8443. Tornando al nostro CMS java dobbiamo correggere l'URL in www.example.com:8080 .

Web server e siti web. Manuale pratico in stile How To 1

Perché questa stranezza e come gestirla?

Gli AS sono pensati per girare in un ambiente isolato e staccato da internet. Motivo per cui non usano le porte web standard. Per rimettere le pagine sulle porte standard dobbiamo mettere uno speciale programma tra l'AS e internet. La funzione di questo programma è di fare da "reverse proxy".

Gli AS usano questa architettura fondamentalmente per tre motivi:

- **sicurezza:** l'AS, i suoi applicativi e tutti i dati sono in uno spazio digitale isolato fisicamente e a livello software rispetto a internet;
- **bilanciamento:** pensando ad applicazioni che offrono servizi di grande dimensioni (es.: il pagamento dei bollettini postali che contava una media di 400 transazioni al minuto) il software di reverse fa anche da bilanciatore, passando le richieste al server libero;
- **alta disponibilità:** il reverse può garantire la continuità di servizio costruendo un nodo ad alta disponibilità. Pensiamo al caso di una manutenzione bloccante su un server il bilanciatore ridirige tutto sul server online in modo del tutto invisibile all'utente.

10.6. IIS e .NET

La soluzione LAMP e LEMP sono molto popolari, sono dietro a moltissimi siti web, ma si riferiscono al mondo *nix anche se non mancano i porting, per sviluppo o prototipazione o per produzione, anche nel mondo Windows.

Microsoft offre come default una soluzione diversa basata su IIS e .NET. Le applicazioni che girano in questo ambiente si caratterizzano per l'estensione .ASP delle pagine web:

- **IIS:** è l'acronimo di Internet Information Services. L'applicativo è disponibile in tutte le versioni Windows anche se l'uso, come server pubblico, è vincolato da licenze software da acquistare;
- **.NET:** in prima battuta lo possiamo definire come il PHP di Microsoft. Di fatto funziona con la stessa logica. Da qualche anno il core è diventato uno spin-off ed è disponibile per tutti come open source.

È da considerare che la combinata IIS + .NET funziona bene ed è ben integrata con tutto l'ecosistema Microsoft. Anche lo sviluppo di applicazioni è semplice e con poche righe si integra sia Word, piuttosto che Excel o SQL Server.

È una tecnologia di cui averne sicuramente consapevolezza insieme al dato che per metterla in produzione dobbiamo comprare alcune licenze.

10.7. Altre tecnologie

Allo scopo introduttivo e didattico di questo manuale è opportuno far presente che “abbiamo guardato dal buco della chiave”.

Le tecnologie per realizzare un web server non sono solo queste.

È importante averne consapevolezza e conoscere i seguenti punti:

- alcune tecnologie sono verticali, legate a una sola azienda e realizzate per uno scopo preciso;
- altre tecnologie sono nate con l'aspirazione a diventare standard per contrastare altre aziende o conquistare quote di mercato;
- l'iniziale libertà che aveva e permetteva internet è diventata progressivamente uno spazio con standard precisi. Pertanto guardando a tecnologie diverse il criterio fondamentale è che siano conformi agli standard pubblici consolidati.

Per capirci un'eventuale app pubblicata oggi che fa brillantemente un determinato servizio internet è da prendere in seria considerazione se è conforme agli standard internet.

11. Conclusione

Costruire un web server è un'avventura bella e facile se restiamo su obiettivi base. L'operazione, oltre alla pregevole rilevanza didattica, permette realizzazioni anche senza costi o con costi contenuti offrendo una tecnologia democratica e sostenibile se adotta licenze open source.

Fatta questa considerazione riteniamo che sia evidente a tutti che si tratta di un mondo assai complesso e variegato.

Con le pratiche e le spiegazioni offerte in queste pagine possiamo raggiungere buoni livelli di base. Non è stato assolutamente affrontato l'argomento della personalizzazione e gestione del nostro CMS, ma questo è un obiettivo diverso da quello di questo manuale.

Mentre nella configurazione e gestione del server e dell'ambiente web abbiamo toccato solo il primo livello.

Questo significa che ci sono molti aspetti tecnici ancora da affrontare. Ma questi saranno oggetto delle prossime guide.

Intanto divertitevi e cimentatevi con i fondamentali qui esposti: questo è solo il piatto di degustazione!

12. Indice

1. Premessa.....	2
1.1. Panoramica.....	3
1.2. Concetti base.....	3
1.3. Architettura di un sistema Web.....	5
1.4. Prepariamo il server.....	7
2. Installare il webserver.....	8
2.1. Installare Apache HTTP Server.....	9
2.2. Testiamo Apache HTTP Server.....	9
2.3. Prima conclusione per Apache.....	11
3. Installare PHP.....	11
3.1. Installiamo PHP.....	12
3.2. Verifichiamo PHP.....	12
3.3. Prima conclusione per PHP.....	13
4. Installare MariaDB.....	14
4.1. Installiamo MariaDB.....	14
4.2. Verifichiamo MariaDB.....	16
4.3. Gestiamo MariaDB con PhpMyAdmin.....	17
4.4. Gestiamo MariaDB da DBeaver.....	20
4.5. PostgreSQL.....	24
5. Installiamo un CMS.....	25
5.1. Preparazione per Joomla.....	25
5.2. Installare Joomla.....	29
5.3. Verifichiamo Joomla.....	33
5.4. Preparazione per WordPress.....	33
5.5. Installare WordPress.....	33
5.6. Verifichiamo WordPress.....	37
5.7. Conclusione sui CMS.....	37
6. Backup.....	38
6.1. Backup: cos'è.....	38
6.2. Backup: tipologie e frequenza.....	42
6.2.1. Tipologia.....	42
6.2.2. Snapshot.....	42
6.2.3. Frequenza.....	43

Web server e siti web. Manuale pratico in stile How To 1

6.3. Backup in esecuzione.....	44
6.3.1. Device.....	44
6.3.2. Tar.....	45
6.3.3. Rsync.....	48
6.3.4. Amanda.....	49
6.3.5. Bacula.....	51
6.4. Backup: la conservazione.....	53
6.4.1. Tempo.....	53
6.4.2. Luogo.....	53
6.5. Backup VS disaster recovery.....	55
7. Consultare-Analizzare i log.....	56
7.1. Log: la teoria.....	56
7.2. Log: accesso, lettura e gestione.....	57
7.3. Log: tool di analisi.....	59
8. Provider.....	60
8.1. Provider: premessa.....	60
8.2. Provider: cosa sono.....	61
8.3. Provider: acquisto di un dominio e di un web-hosting.....	62
8.4. Provider: usare SFTP e SSH.....	65
9. Certificati SSL.....	67
9.1. SSL: premessa.....	67
9.2. SSL e TLS; cosa sono.....	68
9.3. SSL: acquisto da un provider.....	69
9.4. SSL: Let's Encrypt.....	70
9.4.1. Let's Encrypt: configuriamo example.com.....	71
9.4.2. Let's Encrypt: creiamo il certificato con certbot.....	72
10. Compendio.....	73
10.1. GDPR, crittazione e sicurezza.....	73
10.1.1. GDPR.....	73
10.1.2. Crittazione.....	73
10.1.3. Sicurezza.....	74
10.2. Apache: Host e VHost.....	75
10.3. Accelerare PHP.....	76

Web server e siti web. Manuale pratico in stile How To 1

10.4. Ruby, Node.js, Go.....	77
10.4.1. Ruby on Rail.....	77
10.4.2. Node.js.....	77
10.4.3. Go.....	78
10.5. Java a AS.....	78
10.5.1. Java: quale razza?.....	78
10.5.2. Java e gli AS.....	79
10.5.3. Java e i reverse proxy.....	79
10.6. IIS e .NET.....	80
10.7. Altre tecnologie.....	81
11. Conclusione.....	81
12. Indice.....	82



Repository del manuale e della serie
<https://www.digitaldsb.it/portal/informatica>



Piattaforma di elearning del corso
«Web server e siti web»
<https://www.digitaldsb.it/moodle>

