





# **Web server con Apache e Mysql**

## Indice

Prefazione.....	pag. 5
Prepariamo il server.....	pag. 6
Installiamo Apache e compagnia bella.....	pag. 10
Settaggio base di Apache.....	pag. 17
Creare Virtual Host.....	pag. 19
Connessione HTTPS.....	pag. 22
Awstats software per statistiche.....	pag. 25
Proteggere le pagine Web.....	pag. 28
Procedura.....	pag. 29
Redirect http/https.....	pag. 33
Riabilitare il firewall.....	pag. 34



## *Prefazione*

In questa guida ho voluto riportare le configurazioni di base importanti per il corretto funzionamento di un web server basato su Apache, che eseguo ogni volta che mi viene richiesto la configurazione di un nuovo web server presso le aziende dei miei clienti, in questo modo so che la macchina che sto mettendo in piedi è universale e pronta per qualsiasi esigenza.

La guida si basa principalmente sulla configurazione di virtual host, ovvero la possibilità di gestire sulla stessa macchina più siti internet, in poche parole è quello che fanno i vari service provider quando mettono a disposizione uno spazio per far alloggiare un sito del cliente, creando più virtual host per gestire più siti in una sola macchina server.

Nella guida ad un certo punto troverete come creare un certificato per la connessione sicura del server in SSL, il certificato qui auto prodotto non è da ritenersi valido nel caso in cui il server salga a livello produttivo, questi vanno acquistati presso gli enti competenti.

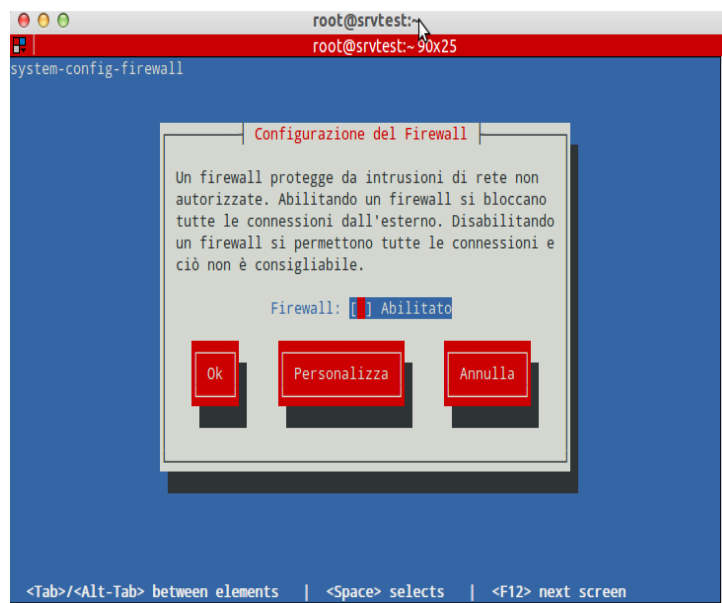
## Prepariamo il server

Dopo aver installato il sistema prepariamoci alla configurazione di base del nostro webserver, che avrà le seguenti configurazioni. Il server utilizzerà un unico IP, che per il nostro test sarà 172.16.200.170, dove verranno realizzati, due virtual host denominati sito1.com e sito2.org , iniziamo con il disabilitare momentaneamente il firewall interno e SELinux di CentOS, questo solo per il periodo di installazione e testing del sistema poi riabiliteremo il tutto.

Iniziamo con l'installazione dell'utility per disabilitare il firewall

```
# yum install system-config-firewall-tui
# system-config-firewall-tui
```

Questo è quanto dovremmo vedere, per ora disabilitiamo il firewall togliendo il simbolo di spunta.



Confermiamo per due volte la volontà di disabilitare il firewall e controlliamo che il firewall sia disabilitato.

```
# iptables -L

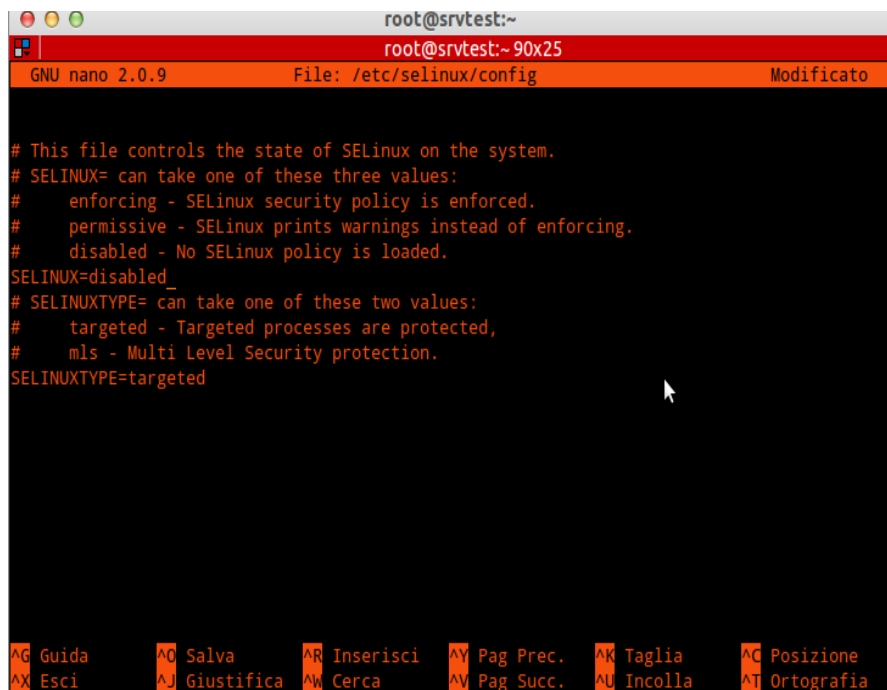
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination
```

```
Chain OUTPUT (policy ACCEPT)
target    prot opt source      destination
```

Disabilitiamo ora SELinux andando a sostituire enforcing con disabled accando a SELINUX

```
# nano /etc/selinux/config
```



```
root@srvtest:~
root@srvtest:~ 90x25
GNU nano 2.0.9 File: /etc/selinux/config Modificato

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=disabled_
# SELINUXTYPE= can take one of these two values:
#   targeted - Targeted processes are protected,
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted

^G Guida      ^O Salva      ^R Inserisci  ^Y Pag Prec.  ^K Taglia     ^C Posizione
^X Esci       ^J Giustifica ^W Cerca      ^V Pag Succ.  ^U Incolla   ^T Ortografia
```

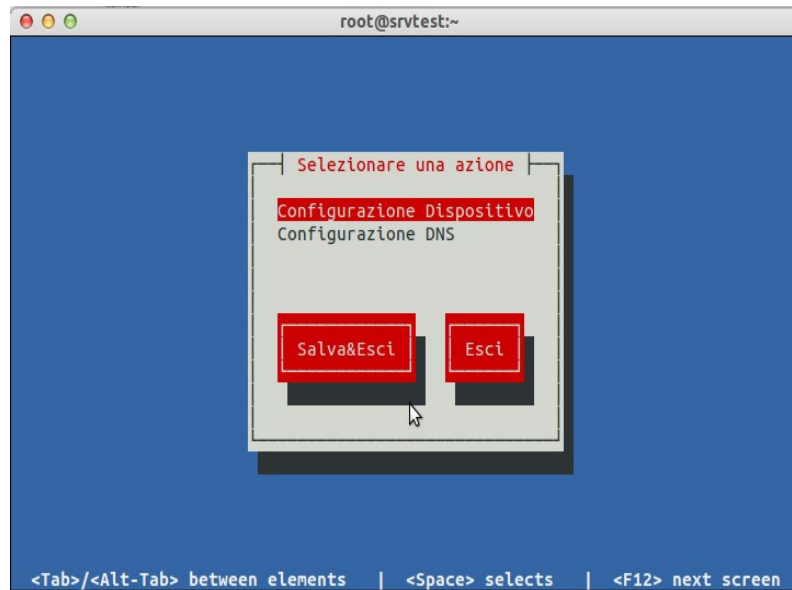
Riavviamo il server per abilitare le impostazioni.

Anche da console è possibile impostare la scheda di rete semplicemente installando il tools necessario:

```
# yum install system-config-network-tui
```

Una volta installato il tool ci basterà richiamarlo con il comando system-config-network-tui e avremo a video una comoda interfaccia per configurare la scheda di rete:

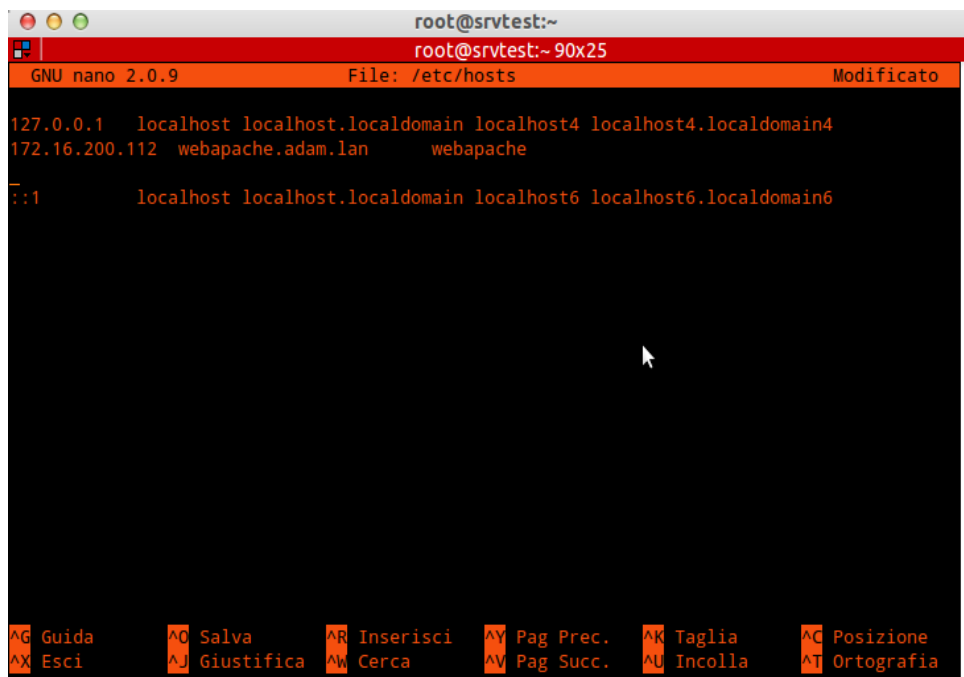




Impostiamo ora il nome del nostro server:

```
# nano /etc/hosts
```

Una volta entrati editiamo il file in questo modo:



Abilitiamo ora il repository addizionale EPEL per l'installazione di software particolare importiamo prima la chiave criptata

```
# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY*
```

Abilitiamo i repository RPMforge e EPEL non prima di aver installato il nostro fido wget

```
# yum install wget
```

```
# rpm --import http://dag.wieers.com/rpm/packages/RPM-
GPG-KEY.dag.txt

# cd /tmp
# wget http://pkgs.repoforge.org/rpmforge-release/rpmforge-
release-0.5.2-2.el6.rf.x86_64.rpm

# rpm -ivh rpmforge-release-0.5.3-1.el6.rf.x86_64.rpm
```

Installiamo ora il repository EPEL

```
# rpm --import https://fedoraproject.org/static/0608B895.txt

# cd /tmp
# wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-
release-6-8.noarch.rpm

# rpm -ivh epel-release-6-8.noarch.rpm

# yum install yum-priorities
```

Editiamo il file epel.repo per mettere questo repository come prioritario

```
# nano /etc/yum.repos.d/epel.repo
```

Aggiungiamo la linea priority=10 nella sezione [epel]

```
[epel]
name=Extra Packages for Enterprise Linux 6 - $basearch
#baseurl=http://download.fedoraproject.org/pub/epel/6/$basea
rch
mirrorlist=https://mirrors.fedoraproject.org/metalink?
repo=epel-6&arch=$basearch
failovermethod=priority
enabled=1
priority=10
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
```

Aggiorniamo i repository

```
# yum update
```

Installiamo ora dei pacchetti che ci possono divenire utili

```
# yum groupinstall 'Development Tools'
```

### *Installiamo Apache e compagnia bella*

Bene Installiamo ora Apache, MySQL e phpMyAdmin

```
# yum install ntp ntpdate ntp-doc httpd mod_ssl mysql-server
php php-mysql php-mbstring phpmyadmin
```

Prima di tutto per un server che si rispetti, dobbiamo configurare il server ntp in modo che l'ora e la data del nostro server sia sempre precisa.

Chiediamo al server di sincronizzare date e ora utilizzando gli orologi atomici messi a disposizione dell'Istituto Nazionale di Ricerca Metrologica che ci mette a disposizione ntp1.inrim.it e ntp2.inrim.it, inseriamo questo comando:

```
# ntpdate ntp1.inrim.it ntp2.inrim.it
```

Per assicurarci che il server ntp venga eseguito ad ogni avvio della macchina usiamo

```
# chkconfig ntpd on
# /etc/init.d/ntpd start
```

Iniziamo ora con l'impostazione del server MySQL, impostiamo la password di root, digitiamo il comando:

```
# mysql_secure_installation
```

Seguiamo ora i passi di configurazione, la prima richiesta ci indica se vogliamo creare una password per l'utente root, premiamo invio per accettare

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

```
In order to log into MySQL to secure it, we'll need the current
password for the root user. If you've just installed MySQL,
and
```

```
you haven't set the root password yet, the password will be
blank,
so you should just press enter here.

Enter current password for root (enter for none):
```

Ci viene ora richiesto se vogliamo settare la password per root, ovviamente rispondiamo si e inseriamo una password forte per l'utente root di mysql

```
Set root password? [Y/n] <-- ENTER
New password: <-- digitare la vostra password di root
Re-enter new password: <-- digitate nuovamente la password
Password updated successfully!
Reloading privilege tables..
... Success!
```

Per default Mysql ha abilitato all'accesso l'utente anonymous, qui di seguito dopo aver inserito la password di root ci viene chiesto se vogliamo disabilitare l'utente anonymous, ovviamente rispondiamo con un Y

```
By default, a MySQL installation has an anonymous user,
allowing anyone
to log into MySQL 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] <-- ENTER
... Success!
```

Normalmente root deve essere impostato solo come utente locale e non deve essere abilitato come utente d'accesso remoto

```
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] <-- ENTER
... Success!
```

Alla domanda rispondiamo con Y.

Come tutti i database anche Mysql possiede un database chiamato test, da qui può accedere chiunque pertanto è buona norma rimuoverlo.

```
By default, MySQL 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] <-- ENTER
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
```

Riavviamo ora i privilegi delle tabelle

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

Reload privilege tables now? [Y/n] <-- ENTER
... Success!

Cleaning up...
```

Con mysql abbiamo terminato ora la configurazione di base è sicura e non ci resta che usarlo.

Configuriamo ora l'interfaccia grafica di Mysql ovvero PhpMyAdmin in modo che ci permetta un accesso remoto per poter lavorare sul server, prima di tutto accediamo a questo path:

```
# nano /etc/httpd/conf.d/phpmyadmin.conf
```

Dovremo visualizzare una situazione simile a questa:

```
#
# Web application to manage MySQL
#

<Directory "/usr/share/phpmyadmin">
  Order Deny,Allow
  Deny from all
  Allow from 127.0.0.1
</Directory>

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

```
Alias /phpMyAdmin /usr/share/phpmyadmin
Alias /mysqladmin /usr/share/phpmyadmin
```

Apportiamo le modifiche che vediamo qui sotto:

```
#
# Web application to manage MySQL
#
#<Directory "/usr/share/phpmyadmin">
# Order Deny,Allow
# Deny from all
# Allow from 127.0.0.1
#</Directory>
Alias /phpmyadmin /usr/share/phpmyadmin
Alias /phpMyAdmin /usr/share/phpmyadmin
Alias /mysqladmin /usr/share/phpmyadmin
```

Salviamo e usciamo.

Cambiamo ora la tipologia di autenticazione di PhpMyAdmin da cookie a http:

```
# nano /usr/share/phpmyadmin/config.inc.php
```

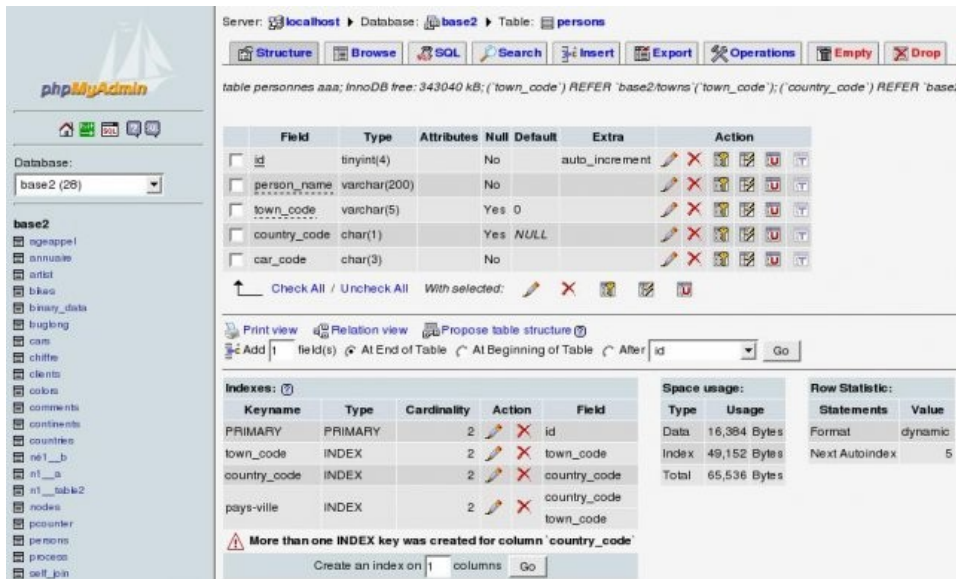
Cambiamo in questo modo:

```
[...]
/* Authentication type */
$cfg['Servers'][$i]['auth_type'] = 'http';
[...]
```

Salviamo, ora facciamo in modo che apache si avvii automaticamente ad ogni avvio del sistema e avviamo apache

```
chkconfig httpd on
/etc/init.d/httpd start
```

A. questo punto non ci resta che provare la connessione verso PhpMyAdmin inserendo nel browser la stringa: Ipserver o nomeserver/phpmyadmin, dovremmo visualizzare la richiesta di utente e password da parte di quest'ultimo.



ecco cosa dovreste vedere dopo il login l'interfaccia per MySQL

Installiamo ora alcune utility per apache come php, perl,

```
# yum install php php-devel php-gd php-imap php-ldap php-
mysql php-odbc php-pear php-xml php-xmlrpc php-pecl-apc
php-mbstring php-mcrypt php-mssql php-snmp php-soap php-
tidy curl curl-devel perl-libwww-perl ImageMagick libxml2
libxml2-devel mod_fcgid php-cli httpd-devel
```

Una volta installato tutto editiamo il file php.ini e sostituiamo il report degli error

```
# nano /etc/php.ini
```

Cerchiamo al suo interno la riga contenente cgi.fix\_pathinfo=1 e togliamo il simbolo #

```
[...]
;error_reporting = E_ALL & ~E_DEPRECATED
error_reporting = E_ALL & ~E_NOTICE
[...]
;
;      cgi.fix_pathinfo          provides          *real*
PATH_INFO/PATH_TRANSLATED support for CGI. PHP's
; previous behaviour was to set PATH_TRANSLATED to
SCRIPT_FILENAME, and to not grok
; what PATH_INFO is.  For more information on
PATH_INFO, see the cgi specs. Setting
; this to 1 will cause PHP CGI to fix its paths to conform to
the spec. A setting
; of zero causes PHP to behave as before.  Default is 1.  You
should fix your scripts
; to use SCRIPT_FILENAME rather than
```

```
PATH_TRANSLATED.  
; http://www.php.net/manual/en/ini.core.php#ini.cgi.fix-  
pathinfo  
cgi.fix_pathinfo=1  
[...]
```

Creiamo ora un file info in php per testare che quest'ultima funziona, entriamo nella cartella html, e creiamo il file test.php

```
# cd /var/www/html  
# nano test.php
```

Inseriamo ora questo piccolo codice in php:

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

Apriamo ora il browser e digitiamo sulla barra degli indirizzi `http://IP_del_Server/test.php`, dovremmo visualizzare una pagina in php di informazioni sul nostro server, un qualcosa simile a questo:



PHP Version 5.2.13	
System	Linux whub21.webhostinghub.com 2.6.18-274.7.1.el5 #1 SMP Thu Oct 20 16:21:01 EDT 2011 x86_64
Build Date	Apr 29 2010 16:43:18
Configure Command	./configure '--enable-bcmath' '--enable-calendar' '--enable-dbase' '--enable-discard-path' '--enable-exif' '--enable-ftp' '--enable-gd-native-ttf' '--enable-libxml' '--enable-magic-quotes' '--enable-mbstring' '--enable-pdo=shared' '--enable-soap' '--enable-sockets' '--enable-sqlite-utf8' '--enable-zend-multibyte' '--enable-zip' '--prefix=/usr' '--with-bz2' '--with-curl=/opt/curlssl' '--with-curlwrappers' '--with-freetype-dir=/usr' '--with-gd' '--with-gettext' '--with-imagick' '--with-imagick=/opt/php_with_imap_client' '--with-imagick-ssl=/usr' '--with-jpeg-dir=/usr' '--with-kerberos' '--with-libdir=lib64' '--with-libexpat-dir=/usr' '--with-libxml-dir=/opt/xml2' '--with-libxml-dir=/opt/xml2' '--with-mcrypt=/opt/libmcrypt' '--with-mhash=/opt/mhash' '--with-mime-magic' '--with-mm=/opt/mm' '--with-mysql' '--with-mysqli' '--with-openssl=/usr' '--with-openssl-dir=/usr' '--with-pdo-mysql=shared' '--with-pdo-sqlite=shared' '--with-pic' '--with-png-dir=/usr' '--with-pspell' '--with-sqlite=shared' '--with-tidy=/opt/tidy' '--with-ttf' '--with-xmirc' '--with-xpm-dir=/usr' '--with-xsl=/opt/xslt' '--with-zlib' '--with-zlib-dir=/usr'
Server API	CGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/lib
Loaded Configuration File	/home/whhsup5/public_html/php.ini
Scan this dir for additional .ini files	(none)
additional .ini files parsed	(none)
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no

Bene ora che abbiamo verificato che php funziona, installiamo ora anche il modulo per ruby (non si sa mai, potrebbe essere utile a qualche sviluppatore)

```
# yum install httpd-devel ruby ruby-devel
```

Dopo aver effettuato il download spostiamoci nella cartella tmp per installare il modulo ruby per apache.

```
# cd /tmp
# wget
http://fossies.org/unix/www/apache_httpd_modules/mod_ruby-1.3.0.tar.gz
# tar zxvf mod_ruby-1.3.0.tar.gz
# cd mod_ruby-1.3.0/
# ./configure.rb --with-apr-includes=/usr/include/apr-1
# make
# make install
```

Terminata l'installazione del modulo ruby, dobbiamo fare in modo che venga caricato dallo stesso apache, pertanto dovremmo creare il file ruby.conf

```
# nano /etc/httpd/conf.d/ruby.conf
```

E inseriamo al suo interno questa stringa

```
LoadModule ruby_module modules/mod_ruby.so  
RubyAddPath /1.8
```

Salviamo e usciamo, ora riavviamo apache per far inglobare il modulo.

Passiamo ora all'installazione del modulo Python

```
# yum install mod_python
```

Dopo l'installazione restartiamo il server apache

```
# /etc/init.d/httpd restart
```

Per quanto riguarda la configurazione del server avremmo finito, ma voglio installare anche un antivirus, non si sa mai?

Installiamo clamv come antivirus, abbiamo già visto l'installazione di quest'ultimo ma la riporteremo nuovamente

```
# yum install clamav clamd perl-DBD-mysql unzip unrar  
bzip2
```

Terminata l'installazione, permettiamo al motore di clamav di avviarsi automaticamente ad ogni avvio della macchina e poi eseguiamo il primo aggiornamento del database dei virus

```
# chkconfig clamd on  
# /usr/bin/freshclam  
# /etc/init.d/clamd start
```

L'antivirus è installato e funzionante.

## Settaggio base di apache

Iniziamo con il dire che dopo tutto questo il server web è già funzionante, infatti se digitate sul vostro browser l'indirizzo ip del vostro server dovrete visualizzare la pagina di benvenuto di apache su in CentOS

## Apache 2 Test Page

powered by **CentOS**

---

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page it means that the Apache HTTP server installed at this site is working properly.

---

**If you are a member of the general public:**

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems or is undergoing routine maintenance.


If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting `www.example.com`, you should send e-mail to "webmaster@example.com".

**If you are the website administrator:**

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

You are free to use the images below on Apache and CentOS Linux powered HTTP servers. Thanks for using Apache and CentOS!



**About CentOS:**

**The Community ENTERprise Operating System** (CentOS) is an Enterprise-class Linux Distribution derived from sources freely provided to the public by a prominent North American Enterprise Linux vendor. CentOS conforms fully with the upstream vendors redistribution policy and aims to be 100% binary compatible. (CentOS mainly changes packages to remove upstream vendor branding and artwork.) The CentOS Project is the organization that builds CentOS.

For information on CentOS please visit the [CentOS website](#).

**Note:**

CentOS is an Operating System and it is used to power this website; however, the webserver is owned by the domain owner and not the CentOS Project. **If you have issues with the content of this site, contact the owner of the domain, not the CentOS project.**

Unless this server is on the CentOS.org domain, the CentOS Project doesn't have anything to do with the content on this webserver or any e-mails that directed you to this site.

For example, if this website is `www.example.com`, you would find the owner of the `example.com` domain at the following WHOIS server:  
<http://www.internic.net/whois.html>

Ora vediamo come impostare una configurazione di base, iniziamo con l'editare il file `httpd.conf`:

```
# nano /etc/httpd/conf/httpd.conf
```

Nella configurazione di base del file `httpd.conf` andiamo a modificare le seguenti voci, impostiamo prima la voce Listen, ovvero la porta in ascolto di Apache

```
# Change this to Listen on specific IP addresses as shown
below to
# prevent Apache from glomming onto all bound IP addresses
(0.0.0.0)
#
#Listen 12.34.56.78:80
Listen 80
```

Nessuno comunque ci impedisce di utilizzare un'altra porta non standard per apache, impostiamo ora il nome del server cambiando la direttiva

```
# ServerName www.example.com:80
ServerName www.miosito.com
```

La direttiva `DocumentRoot` è quella che indica al server dove deve puntare per visualizzare il sito web, di default la `DocumentRoot` di apache la troviamo nel path `/var/www/html`, anche qui nessuno ci impedisce di cambiare questo path con altro, per esempio `/home/NomeSito/html`, questa se

modificata deve essere cambiata anche nella direttiva `<Directory "/var/www/html">`

Un'altra modifica l'apportiamo alla direttiva `<Directory/>`, questa nel caso in cui il nostro server sia fornito di connessione criptata https per il nostro sito (che vedremo tra poco) e quella di inserire come commento la voce `Options FollowSymKinks`, in questo modo:

```
<Directory />
#Options FollowSymLinks
Options Indexes FollowSymLinks Includes ExecCGI
AllowOverride None
Allow from all
</Directory>
```

Scendendo troviamo la direttiva `DirectoryIndex`, in questa direttiva si andrà ad aggiungere le estensioni delle pagine che faranno da index del sito come vediamo nell'esempio qui sotto:

```
# The index.html.var file (a type-map) is used to deliver
content-
# negotiated documents. The MultiViews Option can be used
for the
# same purpose, but it is much slower.
#
DirectoryIndex index.html index.php index.htm index.xml
```

Se volessimo eliminare la pagina di benvenuto di apache, ci basterà editare il file `welcome.conf`:

```
# nano /etc/httpd/conf.d/welcome.conf
```

E inserire il simbolo `#` come nell'esempio qui sotto

```
# <LocationMatch "/+$">
#   Options -Indexes
#   ErrorDocument 403 /error/noindex.html
#</LocationMatch>
```

Salvare e uscire ora la pagina di benvenuto non è più visibile

## Creare Virtual Host

Un Virtual Host è un metodo usato sui server web per ospitare più siti web con vari nomi di dominio su un unico server e avolte utilizzando solo un'indirizzo IP. La configurazione del virtual host è supportata dall'apertura e chiusura dei tag `<VirtualHost *:80>` e `</VirtualHost>`, ovvero tutti i parametri necessari per quanto riguarda path del `DocumentRoot`, `ServerName`, `ServerAlias` ecc.. deve essere inserita tra i due tag mostrati in precedenza; ammettiamo di voler creare due siti web

all'interno del nostro server, che chiameremo sito.com e sito2.org, nel nostro esempio indicheremo anche una DocumentRoot diversa rispetto a quella presente di default nel file di configurazione visto poc'anzi, iniziamo con il creare due folder dove saranno ospitati i siti web di test,

```
# mkdir -p /var/www/sito1.com/public_html
# mkdir -p /var/www/sito2.org/public_html
```

Create le cartelle dobbiamo indicare al server chi ne è il proprietario ovvero l'owner delle cartelle:

```
# chown -R apache:apache /var/www/sito1.com/public_html
# chown -R apache:apache /var/www/sito2.org/public_html
```

Una cosa molto importante dopo aver impostato l'owner delle cartelle è quella dei permessi, nel nostro caso i permessi da impostare alla root di apache sono:

```
# chmod 755 /var/www
```

Ora possiamo configurare i virtual host, editiamo il file httpd.conf e spostiamoci in fondo ad esso e scriviamo quanto segue nell'esempio, questo per il sito1.com. (La voce che viene scritta come NameVirtualHost \*:80, deve essere inserita una sola volta e poi sotto tutti gli host virtuali che volete inserire, inserendo il simbolo \* indichiamo che il virtual host risponde a qualsiasi IP, se invece vogliamo specificare a quale ip esso deve rispondere dobbiamo impostare come NameVirtualHost IndirizzoIP:80 e nel tag inseriamo <VirtualHost IndirizzoIP:80>.

```
NameVirtualHost 172.16.200.170:80

<VirtualHost sito1.com:80>
    ServerAdmin webmaster@example.com
    DocumentRoot /var/www/sito1.com/public_html
    ServerName www.sito1.com
    ServerAlias sito1.com
    <Directory "/var/www/sito1.com/public_html">
        Options All Indexes FollowSymLinks Includes ExecCGI
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
    ErrorLog /var/www/sito1.com/error.log
    CustomLog /var/www/sito1.com/access_log combined
</VirtualHost>
```

Questo è il secondo host virtuale sito2.org

```
<VirtualHost sito2.org:80>
  ServerAdmin webmaster@example.com
  DocumentRoot /var/www/sito2.org/public_html
  ServerName www.sito2.org
  ServerAlias sito2.org
  <Directory "/var/www/sito2.org/public_html">
    Options All Indexes FollowSymLinks Includes ExecCGI
    AllowOverride None
    Order allow,deny
    Allow from all
  </Directory>
  ErrorLog /var/www/sito2.org/error.log
  CustomLog /var/www/sito2.org/access_log combined
</VirtualHost>
```

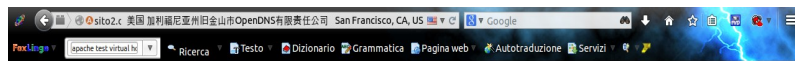
Una volta impostati i virtual host testiamo la configurazione con il comando `httpd -t` se tutto è ok, allora riaviamo apache con il comando.

```
# service httpd restart
```

Ora non resta che testare la connessione, se avete impostato l' IP del server nei vostri DNS locali e poi in un secondo tempo remoti questo è quanto dovrà apparire:



SITO1.COM



SITO2.ORG



## Connessione https

Vediamo ora come impostare una connessione criptata con SSL, iniziamo con l'installazione di mod\_ssl

```
# yum install mod_ssl
```

Creiamo ora una nuova directory, dove inseriremo i certificati di sicurezza, in questo caso auto

prodotti e pertanto non validi hai fini di business.

```
# mkdir /etc/httpd/ssl
```

Accediamo alla cartella appena creata

```
# cd /etc/httpd/ssl
```

Creiamo ora il certificato, con validita di 365 giorni ovvero un anno con una chiave di criptata di 2048 bit.

```
# openssl req -x509 -nodes -days 365 -newkey rsa:2048
-keyout /etc/httpd/ssl/apache.key -out /etc/httpd/ssl/apache.crt
```

Dando l'invio a questo comando inseriamo i dati richiesti.

```
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Italy
Locality Name (eg, city) []:Varese
Organization Name (eg, company) [Internet Widgits Pty Ltd]:test
inc
Organizational Unit Name (eg, section) []:Dept di manager
Common Name (e.g. server FQDN or YOUR name) []:test.com
Email Address []:webmaster@test.com
```

Prima di iniziare la configurazione e meglio eseguire una copia del file ssl.conf:

```
# cp /etc/httpd/conf.d/ssl.conf /etc/httpd/conf.d/ssl.conf.bak
```

Bene ora che abbiamo effettuato la copia procediamo con la configurazione.

```
# nano /etc/httpd/conf.d/ssl.conf
```

Lasciamo tutto invariato fino alla voce SSL Virtual Host Context e inseriamo questi tag, come abbiamo fatto per i virtualhost in httpd.conf anche qui dobbiamo dichiarare il NameVirtualHost o con l'IP o con un valido nome DNS es: "www.miosito.com"

```
NameVirtualHost 172.16.200.170:443
```



Ora possiamo dichiarare il primo virtual host, dichiarando prima il nome del sito o IP all'interno del tag <VirtualHost>, poi il percorso del sito nella DocumentRoot

```
<VirtualHost sito1.com:443>

# General setup for the virtual host, inherited from global
configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443
DocumentRoot "/var/www/sito1.com/public_html"
```

Successivamente lasciamo tutto come di default tranne per il percorso dei certificati, che nel nostro caso inizialmente troveremo il percorso configurato in questo modo

```
# certificate can be generated using the genkey(1) command.
SSLCertificateFile /etc/pki/tls/certs/localhost.crt

# both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
```

Dobbiamo sotituirlo con questo

```
SSLCertificateFile /etc/https/ssl/apache.crt

SSLCertificateKeyFile /etc/httpd/ssl/apache.key
```

Verifichiamo che la sintassi sia corretta con `httpd -t` se non vengono restituiti errori riavviamo apache:

```
# service httpd restart
```

Collaudiamo il nostro sito collegandoci con `https://sito1.com`,

Facciamo la stessa cosa per il secondo virtual host, copiando la configurazione del primo e inserendo solo i dati che riguardano il secondo sito:

```
<VirtualHost sito2.org:443>

# General setup for the virtual host, inherited from global
configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443
DocumentRoot "/var/www/sito2.org/public_html"

# Use separate log files for the SSL virtual host; note that
```

```

LogLevel
# is not inherited from httpd.conf.
ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
LogLevel warn

# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
SSLEngine on
SSLOptions +StrictRequire

# SSL Protocol support:
# List the enable protocol levels with which clients will be
# able to
# connect.  Disable SSLv2 access by default:
SSLProtocol all -SSLv2
SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:
+HIGH:+MEDIUM:+LOW
SSLCertificateFile /etc/httpd/ssl/apache.crt
SSLCertificateKeyFile /etc/httpd/ssl/apache.key
#SSLOptions      +FakeBasicAuth      +ExportCertData
+StrictRequire
<Files ~ "\.(cgi|shtml|phtml|php3?)$" >
    SSLOptions +StdEnvVars
</Files>
<Directory "/var/www/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>
# "force-response-1.0" for this.
SetEnvIf User-Agent ".*MSIE.*" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0
CustomLog logs/ssl_request_log \
    "%t %h %{SSL_PROTOCOL}x %{
{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>

```

Riavviamo apache e testiamo anche il secondo virtualhost in https

Bene ora che abbiamo terminato con la configurazione dei virtual host, vediamo come installare e configurare Awstat, uno dei migliori strumenti opensource per generare statistiche e grafici di siti web, ftp mail e tanto altro.

## Installare e configurare Awstats su CentOS

Come accennato poc'anzi, Awstats è uno dei migliori software opensource per le statistiche e grafici di vari servizi, queste statistiche vengo preparate automaticamente dai log generati dal

servizio stesso. Il software è configurabile per molti server Web come Apache, IIS, Nginx e altri.

Il primo passo da fare è quello di abilitare i repository EPEL per poter installare awstats e facciamo direttamente senza doverli scaricare prima:

```
# rpm -Uvh http://mirrors.kernel.org/fedora-epel/6/i386/epel-release-6-8.noarch.rpm
```

Per centos 6.\* a 32 bit

```
# rpm -Uvh http://mirrors.kernel.org/fedora-epel/6/x86_64/epel-release-6-8.noarch.rpm
```

Per Centos 6.\* a 64 bit

Verifichiamo poi che i repository siano abilitati con il comando:

```
# yum repolist
```

Ora possiamo installare awstats

```
# yum install awstats
```

Awstats, creerà in automatico il file di configurazione per apache (**attenzione sostituite alla voce allow from 127.0.0.1 con allow from all**)

```
# cat /etc/httpd/conf.d/awstats.conf
Alias /awstatsclasses "/usr/share/awstats/wwwroot/classes/"
Alias /awstatscss "/usr/share/awstats/wwwroot/css/"
Alias /awstatsicons "/usr/share/awstats/wwwroot/icon/"
ScriptAlias /awstats/ "/usr/share/awstats/wwwroot/cgi-bin/"
<Directory "/usr/share/awstats/wwwroot">
  DirectoryIndex awstats.pl
  Options ExecCGI
  order deny,allow
  allow from all
</Directory>
<IfModule mod_env.c>
  SetEnv PERL5LIB
  /usr/share/awstats/lib:/usr/share/awstats/plugins
</IfModule>
```

Creiamo ora il nostro file o i file di configurazione per il nostro/i sito/i basandoci sul file example

```
# cp /etc/awstats/awstats.host.example.com.conf
/etc/awstats/awstats.sito1.com.conf
```

Sostituendo sito1.com con il nome del vostro sito web, e modifichiamo il file appena creato con il nostro editor preferito:

```
# nano /etc/awstats/awstats.sito1.com.conf
```

E sostituiamo le direttive in questo modo:

```
LogFile="/var/www/sito1.com/access_log"
SiteDomain="sito1.com"
HostAliases="sito1.com 172.16.200.170"
```

Dove:

- LogFile: è il path dove risiede il log di apache da analizzare
- SiteDomain: il nome del vostro sito
- HostAliases: il vostro dominio e alias tipo "www.sito1.com" o IP.

A questo punto salviamo il file di configurazione e riavviamo apache:

```
# /etc/init.d/httpd/restart
```

Ricordiamoci di controllare che i log di apache devono essere in "combined" come nell'esempio qui sotto e che abbiamo già impostato precedentemente.

```
# CustomLog /var/www/sito1.com/access_log combined
```

Scheduliamo la generazione di statistiche in crontab ogni 5 minuti, in modo da avere statistiche sempre aggiornate. Per fare questo entriamo in crontab con il comando:

```
# crontab -e
```

E successivamente inseriamo questa riga:

```
*/5 * * * * /usr/bin/perl /usr/share/awstats/wwwroot/cgi-
bin/awstats.pl -config=sito1.com -update
```

Per generare la prima volta le statistiche, basta lanciare manualmente il comando messo in crontab:

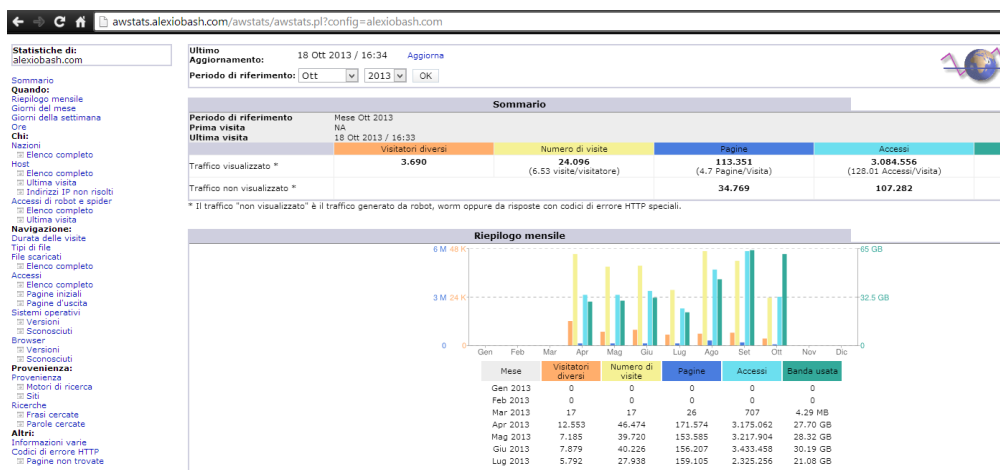
```
# perl /usr/share/awstats/wwwroot/cgi-bin/awstats.pl
-config=sito1.com -update
```

Dovremmo visualizzare un output simile a questo:

```

Create/Update database for config
"/etc/awstats/awstats.sito1.com.conf" by AWStats version 7.0
(build 1.971)
From data in log file "/var/www/sito1.com/access_log"...
Phase 1 : First bypass old records, searching new record...
Direct access after last parsed record (after line 22574)
Jumped lines in file: 22574
Found 22574 already parsed records.
Parsed lines in file: 0
Found 0 dropped records,
Found 0 comments,
Found 0 blank records,
Found 0 corrupted records,
Found 0 old records,
Found 0 new qualified records.
    
```

Apriamo il browser all'indirizzo <http://sito1.com/awstats/awstats.pl?config=sito1.com>, dovremmo visualizzare le statistiche del nostro sito.



## Proteggere pagine web con .htaccess

Proteggere siti web o pagine specifiche nel web server della rete è una prassi comune per tenere lontano occhi indiscreti.

La sicurezza è un concetto fondamentale per una rete, specialmente se gli accessi provengono anche dall'esterno. Soprattutto determinate aree (Intranet, IT, etc.) richiedono un certo grado di sicurezza per non precludere l'integrità dei dati o dei sistemi.

## Procedura

Supponiamo di voler proteggere una specifica pagina web costringendo l'utente a inserire delle credenziali "che ha ricevuto in precedenza". Per richiedere l'autenticazione all'accesso, un metodo utilizzato da Apache è l'utilità del file .htaccess, il file in questione va inserito nella directory da

proteggere, per creare il file utilizziamo il nostro editore nano, inseriremo il file nella directory principale del sito1.com in questo modo l'utente dovrà loggarsi per accedere.

```
# nano /var/www/sito1.com/public_html/.htaccess
```

E al suo interno digitiamo quanto segue:

```
Auth "Secure Area"
AuthType Basic
AuthUserFile /etc/httpd/conf/passwd/.htpasswd
Require valid-user
```

Da notare il path della stringa AuthUserFile, questo indica dove controllare le credenziali dei o degli utenti, notiamo anche AuthType, oltre a Basic, possiamo impostare la password con protezioni più elevate come MD5, o SHA1. Ora facciamo in modo che l'owner del file sia apache e non root

```
# chown apache:apache
/var/www/sito1.com/public_html/.htaccess
```

Ora dobbiamo creare il file delle password per gli utenti che saranno abilitati tranne il comando htpasswd, spostiamoci nella cartella /etc/httpd/conf e creiamo il file delle password per l'utente test, con password test

```
# htpasswd -c /etc/httpd/conf/passwd/.htpasswd test
New password:
Re-type new password:
Adding password for user test
```

La lettera -c va inserita solo quando si crea il primo utente del file .htpasswd, per crearne altri impostare htpasswd senza -c se vogliamo vedere il contenuto del file delle password digitiamo:

```
# cat .htpasswd
```

Per proteggere il file in scrittura assegniamo al file i seguenti diritti:

```
# chmod 644 .htpasswd
```

Diciamo ora che vogliamo proteggere la root ovvero fare in modo che gli utenti prima di accedere alla home o ad altra folder inseriscano le credenziali, accediamo al file httpd.conf che troviamo nel seguente path, /etc/httpd/conf/httpd.conf e andiamo a modificare le seguenti voci. Prima di tutto cerchiamo la <Directory "var/www/html"> e apportiamo le seguenti modifiche:

```
# This should be changed to whatever you set DocumentRoot
```

```

to.
#
<Directory "/var/www/html">

# The Options directive is both complicated and important.
Please see
# http://httpd.apache.org/docs/2.2/mod/core.html#options
# for more information.

Options All Indexes FollowSymLinks Includes MultiViews

# AllowOverride controls what directives may be placed in
.htaccess files.
# It can be "All", "None", or any combination of the
keywords:
# Options FileInfo AuthConfig Limit

AllowOverride AuthConfig

# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all

</Directory>

```

Salviamo le modifiche e riavviamo apache.

```
# service httpd restart
```

Bene ora come accediamo al nostro sito ci verrà richiesto di autenticarci.

Se invece della root volessimo effettuare un accesso tramite htaccess su un virtual host, allora procediamo in questo modo. Configuriamo ora nuovamente il file httpd.conf, in modo che nel virtual host ci sia la dichiarazione nella Directory, cosa importante la dichiarazione delle Directory va sempre fatta all'interno di un virtual host.

```

<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    DocumentRoot /var/www/sito1.com/public_html
    ServerName www.sito1.com
    ServerAlias sito1.com
    <Directory "/var/www/gallaratepec.local/public_html">
        Options All Indexes FollowSymLinks Includes ExecCGI
        Order allow,deny
        Allow from all
        AuthName "Inserire i dati per il Login"
    </Directory>
</VirtualHost>

```

```

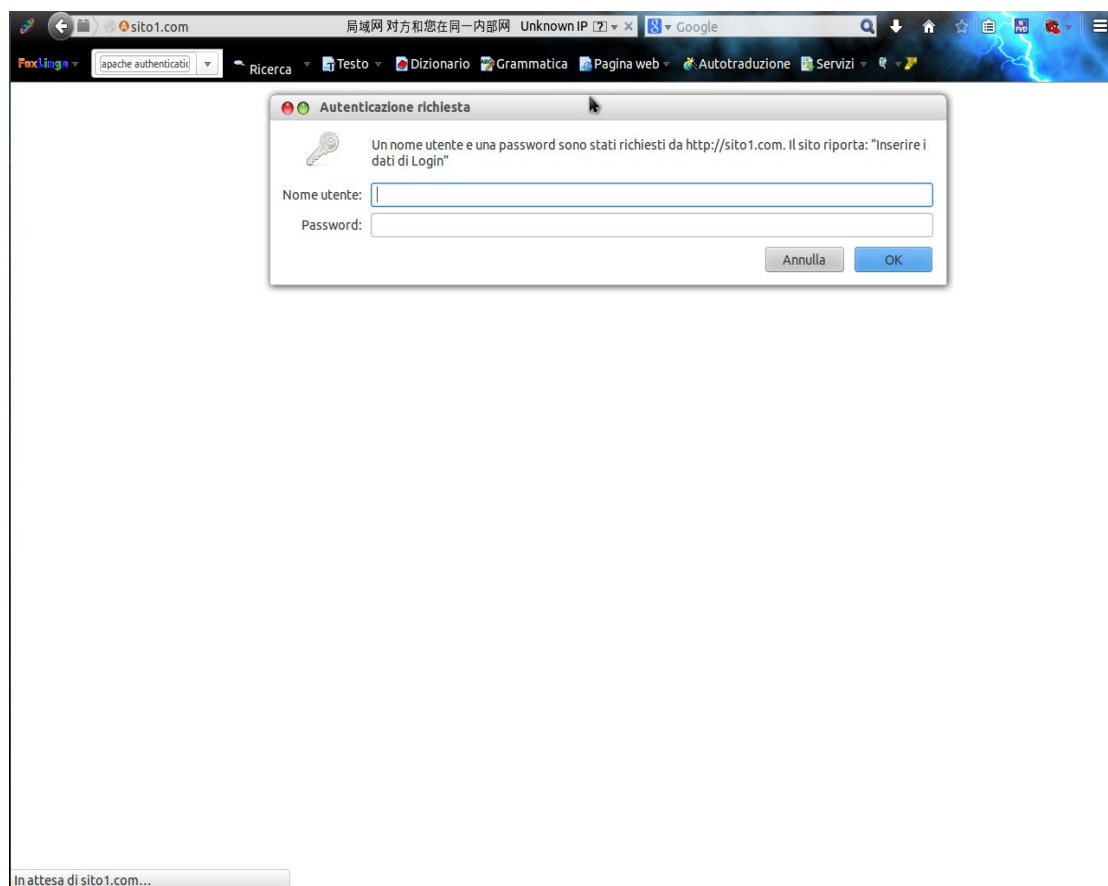
AuthType basic
AuthUserFile /etc/httpd/conf/passwd/.htpasswd
Require valid-user
</Directory>
ErrorLog /var/www/gallaratepec.local/error.log
CustomLog /var/www/gallaratepec.local/access_log
combined
</VirtualHost>

```

Dove:

- AuthName: Indica il testo del messaggio che ci apparirà
- AuthType: Indica il tipo di autenticazione
- AuthUserFile: Indica dove si trova il file relativo per il login degli utenti
- Require valid-user: indica gli utenti che hanno accesso (se non vengono specificati, tutti gli utenti presenti nel file .htpasswd verranno presi in considerazione per l'autenticazione).

Testiamo e dovremo trovarci davanti la richiesta di autenticazione al sito.





## Redirect da http a https su virtual host e autenticazione

Vediamo come eseguire il redirect automatico di un sito da http ad https, in modo che l'utente non deva impostare manualmente la sigla https sulla bara dell'indirizzo. In più faremo in modo che l'utente si loghi prima di arrivare alla pagina.

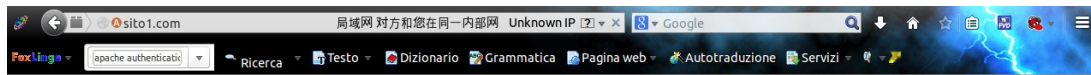
Torniamo all'interno del file httpd.conf e posizioniamoci sopra la riga AuthName, inserita precedentemente per l'autenticazione dell'utente e inseriamo questi tag evidenziati in giallo:

```
<VirtualHost *:80>
  ServerAdmin webmaster@example.com
  DocumentRoot /var/www/sito1.com/public_html
  ServerName www.sito1.com
  ServerAlias sito1.com
  <Directory "/var/www/gallaratepec.local/public_html">
    Options All Indexes FollowSymLinks Includes ExecCGI
    Order allow,deny
    Allow from all
    RewriteEngine On
    RewriteCond %{HTTPS} off
    RewriteRule (.*) https://%{HTTP_HOST}%
    {REQUEST_URI}
    AuthName "Inserire i dati per il Login"
    AuthType basic
    AuthUserFile /etc/httpd/conf/passwd
    Require valid-user
  </Directory>
  ErrorLog /var/www/gallaratepec.local/error.log
  CustomLog /var/www/gallaratepec.local/access_log
  combined
</VirtualHost>
```

Salviamo e chiudiamo il file e poi riavviamo apache


```
# service httpd restart
```

Ora possiamo testare la connessione, la prima cosa che ci viene richiesta è quella di autenticarci, una volta fatto questo veniamo indirizzati al sito in modalità protetta, come mostrato nelle figure qui sotto:



## SITO2.ORG

**Autenticazione richiesta**

 Un nome utente e una password sono stati richiesti da http://sito1.com. Il sito riporta: "Inserire i dati di Login"

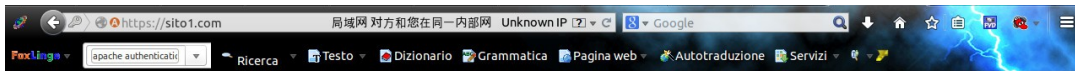
Nome utente:

Password:

In attesa di sito1.com...

---

richiesta di autenticazione




---

redirect automatico della pagina

Nel caso in cui non necessitiamo di autenticazione al sito, ma volessimo applicare solo il redirect in https, ci basterà aggiungere questi tag:

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

sotto la voce Allow from all, in questo modo:

```
<VirtualHost *:80>
  ServerAdmin webmaster@example.com
  DocumentRoot /var/www/sito1.com/public_html
  ServerName www.sito1.com
  ServerAlias sito1.com
  <Directory "/var/www/gallaratepec.local/public_html">
    Options All Indexes FollowSymLinks Includes ExecCGI
    Order allow,deny
    Allow from all
    RewriteEngine On
    RewriteCond %{HTTPS} off
    RewriteRule (.*) https://%{HTTP_HOST}%
    {REQUEST_URI}
  </Directory>
  ErrorLog /var/www/gallaratepec.local/error.log
  CustomLog /var/www/gallaratepec.local/access_log
```

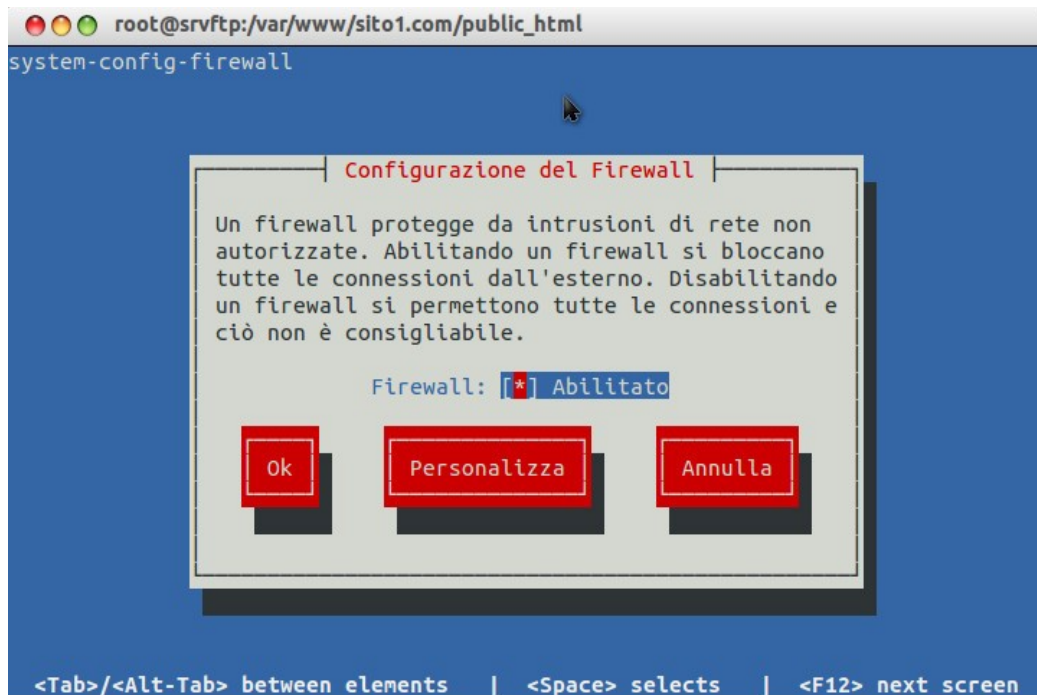
```
combined  
</VirtualHost>
```

## Riabilitare il firewall

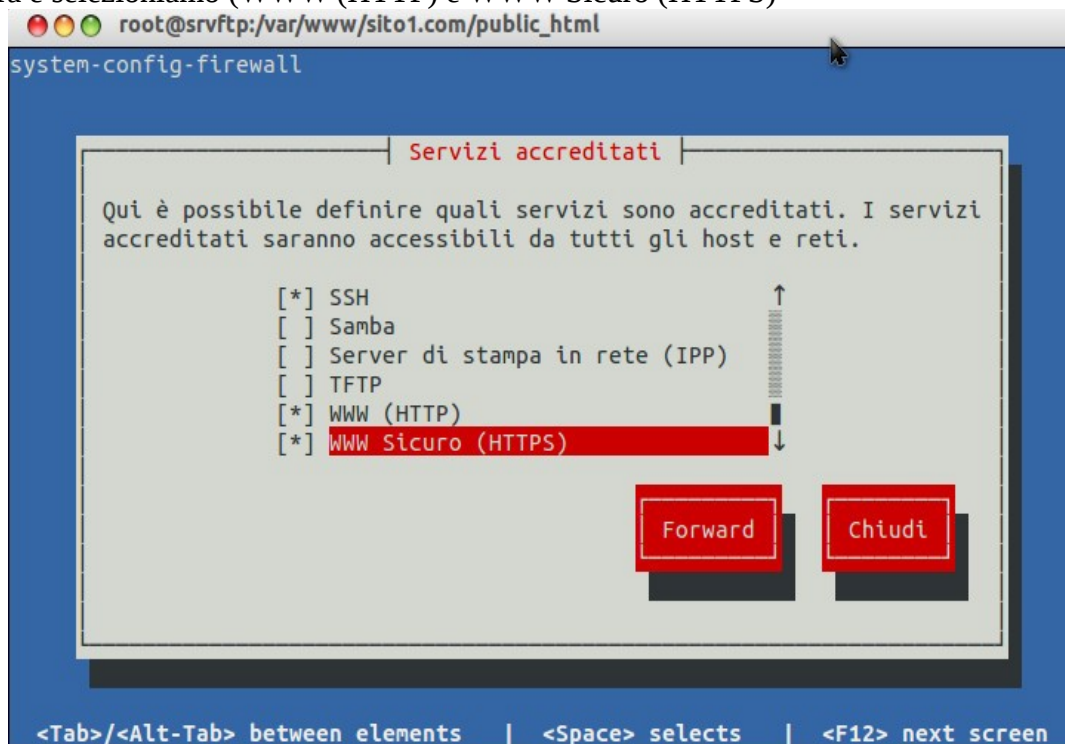
Una volta che abbiamo testato che tutto funzioni possiamo riabilitare SELinux e il firewall, aprendo le porte necessarie al nostro webservice, ovvero la porta TCP 80 e TCP 443, digitiamo sulla console il comando:

```
# system-config-firewall-tui
```

Abilitiamo il firewall



Selezioniamo il tasto personalizza, vi ricordo di muovervi all'interno con il tasto tab della tastiera, tasti freccia e per inserire o togliere il simbolo \* usate la barra spaziatrice. Scendiamo in basso alla maschera e selezioniamo (WWW (HTTP) e WWW Sicuro (HTTPS)



Selezioniamo Chiudi e poi successivamente Ok. Alla richiesta “se premiamo sul pulsante si verrà sovrascritta” la configurazione selezioniamo SI, il firewall è configurato, ora non resta che abilitare SELinux e abbiamo finito, per farlo digitiamo da terminale

```
# nano /etc/selinux/config
```

E inseriamo enforcing al posto di disabled come mostrato qui sotto:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of
enforcing.
#   disabled - No SELinux policy is loaded.

SELINUX=enforcing

# SELINUXTYPE= can take one of these two values:
#   targeted - Targeted processes are protected,
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Salviamo chiudiamo e riavviamo il server. Bene anche questo capitolo è terminato. Nel prossimo capitolo dedicato sempre al web vedremo come configurare un server ftp, da prima una configurazione di base per poi passare ad una connessione protetta.