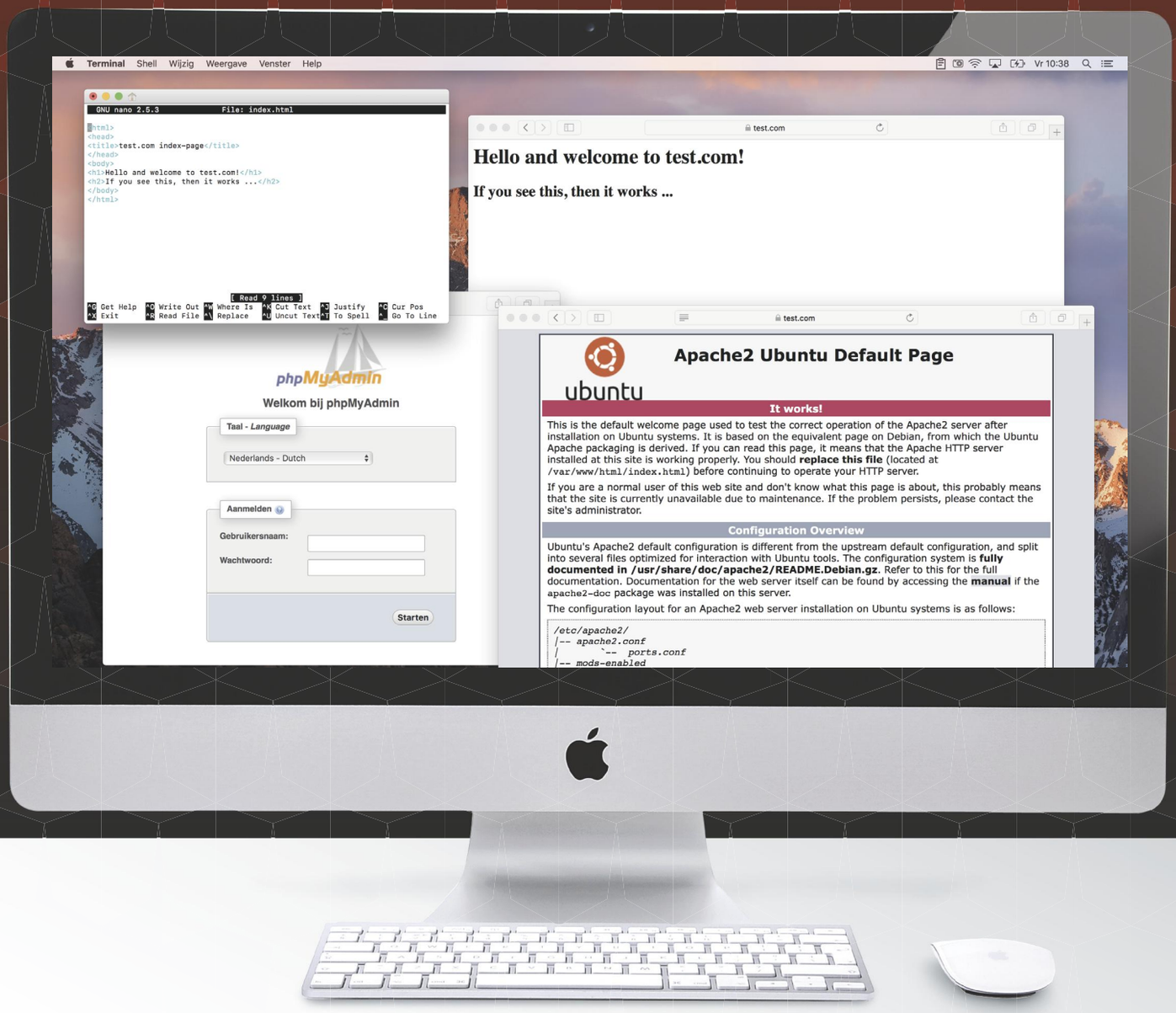
Installeer je eigen webserver met Apache

Om volledig controle te hebben over je webserver kun je het beste zelf de handen uit de mouwen steken en de benodigde componenten met behulp van de commandline installeren. Ook handig als testserver



Om een contentmanagementsysteem te installeren of een passieve HTML/CSS-site op internet te krijgen, is een kant-en-klare webspace die je bij een webhoster kunt huren een prima mogelijkheid. Dan hoef je je niet bezig te houden met wat er op de achtergrond allemaal draait om de boel aan de gang te krijgen en te houden.

Vaak kun je de functionaliteit die geboden wordt modulair aan- of uitzetten via een eigen gebruikersinterface. Je kunt alle functies die je niet gebruikt, uitzetten. Sterker nog: die móet je uitzetten omdat ze de performance en veiligheid van je webserver negatief beïnvloeden. Extra plug-ins maken je server vatbaarder voor aanvallen van buitenaf.

Maar vaak heb je helemaal geen idee wat er op een webserver allemaal draait. Het enige wat je kunt doen, is je bestanden via FTP of een andere manier naar de webserver kopiëren en dan moet alles als een soort automatische magie gaan werken. Als dat niet zo is, resteert je vaak niets anders dan contact opnemen met de supportafdeling van de hoster.

Wil je meer controle over je webserver en alleen over de functionaliteiten beschikken die je daadwerkelijk nodig hebt, of heb je functionaliteiten nodig die gewone webspaces niet bieden, dan moet je zelf aan de slag om je webserver naar je hand te zetten.

Dat klinkt moeilijker dan het is. Iedereen die wel eens met XAMPP of LAMP een webserver op zijn systeem geïnstalleerd heeft, heeft de stappen die nodig zijn om een webserver in de lucht te krijgen eigenlijk al automatisch laten uitvoeren. Voor een lokale testserver biedt dat voldoende mogelijkheden, maar voor een productieserver geldt dat meestal niet.

Gelukkig kun je bij veel hosters ook een 'kale' server huren, bijvoorbeeld in de vorm van een Virtual Private Server (VPS). Bij een VPS wordt een fysieke server – van bijvoorbeeld Dell of HP – opgedeeld in meerdere kleinere virtuele servers. Die virtuele servers worden als VPS verhuurd. Elke gebruiker krijgt dus een eigen virtuele server en kan daarop doen en laten wat hij wil. Doordat de hoster dit zo kan inrichten dat de hardware dag en nacht gelijkmatig wordt belast, bijvoorbeeld door gebruikers uit verschillende tijdzones op één fysieke server te zetten, heb je nauwelijks last van de andere VPS'en. Hierdoor wordt de hardware efficiënter benut en kan deze dus goedkoper worden aangeboden dan een dedicated server. Bij deze laatste optie huur je in feite een server in een groot serverrack in een rekencentrum. Jouw webserver draait dan daadwerkelijk op de kale hardware met een eigen processor, schijven en werkgeheugen. In de praktijk wegen de meerkosten vaak niet op tegen die van een virtuele server, tenzij je verwacht dat de sites die je gaat hosten een enorme internationale toevlucht aan bezoekers oplevert. Heb je een website die dag en nacht druk wordt bezocht, dan is een VPS niet per se de juiste keus.

Zoals gezegd is een VPS helemaal van jou alleen en hoef je deze niet met anderen te delen. Je bent zelf verantwoordelijk voor het installeren van de software op zo'n server, maar vaak wordt hij geleverd met een basisinstallatie op basis van Windows of Linux. Bij webservers wordt vanwege de ontbrekende licentiekosten en de simpelere gebruiksvoorwaarden meestal Linux gebruikt. Daarbij speelt ook mee dat de meeste webserversoftware goed onder Linux wordt ondersteund en eenvoudig te installeren is. Omdat Linux beheerd kan worden via de commandline, dus zonder grafische schil, is dat een aanvalspunt minder en gaat dat een stuk minder stroperig dan met een GUI.

In dit artikel gaan we daarom uit van een minimale Ubuntu 16.04-installatie, waar je een complete webserver van maakt. Dat betekent dat je Apache2, databaseserver MySQL en PHP-ondersteuning van de grond af aan installeert. Als je niet helemaal zeker bent van jezelf, probeer je dat eerst uit op een tijdelijke Ubuntu-server op een pc in je netwerk of een virtuele machine op je eigen computer. Om echt online te gaan, heb je dan een Linux-VPS of dedicated server nodig waar Ubuntu 16.04 al op geïnstalleerd is – met zo min mogelijk extra modules.

Er zijn namelijk aanbieders die bij wijze van service meteen alle toeters en bellen en een desktop-interface voor je mee installeren, zodat je al meteen de beschikking hebt over alle webservermogelijkheden – maar dat is in dit geval nou net niet wat je wilt. Wie Ubuntu op een eigen test-pc of virtuele machine installeert, moet gedurende de installatie een gebruiker aanmaken. Deze heeft standaard geen beheerdersrechten – in Linux-jargon root-rechten genoemd. Inloggen als root is niet heel veilig omdat je dan alles mag. Het risico dat er dan serieus wat misgaat omdat je per ongeluk een verkeerd commando intypt, is behoorlijk groot. Daarom staat Ubuntu dit standaard niet toe. Wil je opdrachten uitvoeren waarvoor je root-rechten nodig hebt, dan moet je die opdracht vooraf laten gaan door `sudo`, wat je kunt lezen als Super User DO. Vervolgens wordt gevraagd om je eigen wachtwoord in te voeren. Weet je van tevoren al dat je behoorlijk wat opdrachten met root-rechten moet uitvoeren, dan kun je beheerder 'root' worden door `sudo -i` en je eigen wachtwoord in te typen. Vergeet na het uitvoeren niet om als root uit te loggen en als gewone gebruiker verder te gaan. Wil je vanaf een Windows-machine inloggen dan moet op Linux ook een SSH-server draaien. Deze voeg je onder Ubuntu met het commando `apt` op dezelfde manier toe als waarop we verderop Apache installeren.

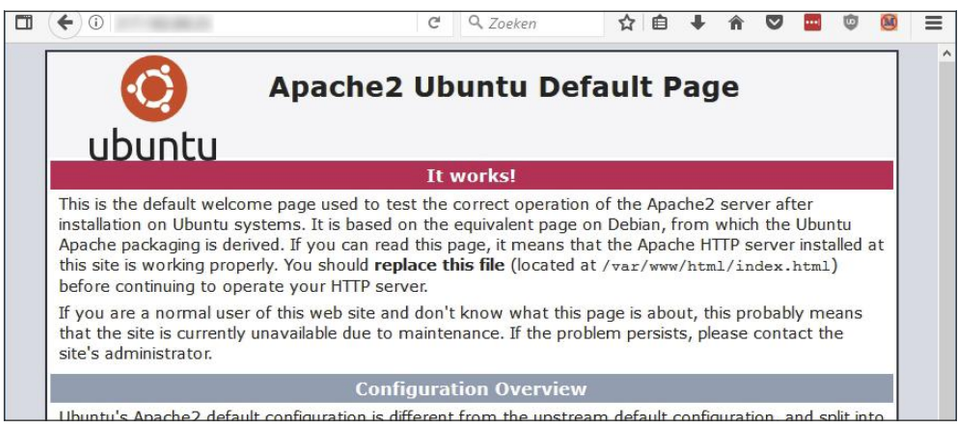
1. Contact met de server

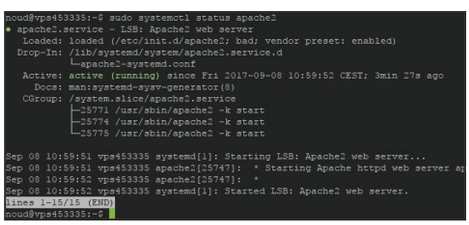
Voor het voorbeeld gaan we uit van een minimale Ubuntu-installatie op een goedkope VPS. Hierop draait geen webserver en er is ook geen gebruikersinterface geïnstalleerd. Om contact te leggen, gebruiken we SSH. Hiervoor heb je een SSH-client nodig voor je besturingssysteem. Voor Windows kan dat met PuTTY, in macOS is standaard een SSH-client ingebouwd, Linux-gebruikers kunnen de Openssh-client installeren.

Bij Putty vul je bij Host Name het IP-adres van je server in, dat je van je provider hebt gekregen. Zorg ervoor dat als protocol SSH is geselecteerd en klik op Open. Daarna moet je inloggen op de server met

Apache License 2.0

Apache HTTP Server, de officiële naam van de webserver, wordt uitgebracht onder de Apache 2.0 License (ALv2). Die houdt in dat de software gratis gebruikt, gedistribueerd en aangepast mag worden zonder dat daar royalty's mee gemoeid zijn.

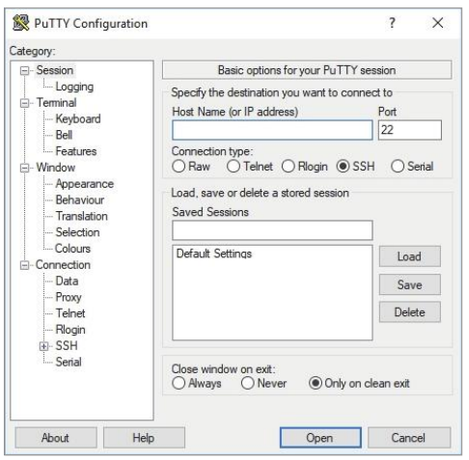




Boven
Met het commando *sudo systemctl status apache2* krijg je informatie over de status van de webserver.

Links
Als de Apache-webserver draait, is dit het standaardscherm dat je dan in een browser te zien krijgt.

de inloggegevens die je van de hoster hebt gekregen. Als je net als wij alleen een wachtwoord hebt, moet je als inlognaam *root* gebruiken. Op dat moment zie je niet veel meer dan een cursor op een zwart scherm, maar je bent dan wel ingelogd op de webserver en kunt commando's intypen om het systeem verder te installeren en configureren.



2. Nieuwe gebruiker

Zoals gezegd is standaard werken als root niet heel veilig omdat je dan alles mag. De kans dat er dan wat serieus misgaat omdat je per ongeluk een verkeerd commando intypt, is behoorlijk groot. Maak daarom eerst een 'gewone' gebruiker aan, die slechts na het intypen van een wachtwoord dingen mag installeren. Vul in het onderstaande commando bij <username> de naam van de gewone gebruiker in. Die wordt toegevoegd aan de groep van beheerders. Typ de volgende commando's in op de SSH-console om in te loggen als gebruiker <username>:

```
adduser <username>
usermod -aG sudo <username>
sudo su  <username>
```

3. Apache installeren

Daarna ga je de webserver installeren. In de praktijk hebben we het meestal over een Apache-server, maar formeel is dat Apache2. Dat zie je bij de installatiecommando's dan ook terug. Zoals boven aangehaald, moeten commando's worden voorafgegaan door `sudo`. Het commando wat daarna volgt, mag immers alleen uitgevoerd worden door een beheerder. Probeer je het commando zonder sudo ervoor te laten werken, dan krijg je een foutmelding dat je daar de rechten niet voor hebt. Update en upgrade de server en laat hem dan Apache installeren. Bij de commando's staat apt voor Advanced Packaging Tool, oftewel het pakketbeheerprogramma dat bij Ubuntu (en andere op Debian gebaseerde besturingssystemen) zorgt voor het beheer van softwarepakketten.

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

4. Firewall instellen

Omdat je Ubuntu-server standaard beveiligd wordt door een eigen firewall, moet je ervoor zorgen dat hij wel van buitenaf via internet bereikbaar is. Daarbij gaat het om de protocollen HTTP en HTTPS, die voor het dataverkeer van websites gebruikt worden. De firewall van Ubuntu heet ufw, wat een afkorting is van uncomplicated firewall (maar je mag het ook lezen als Ubuntu-firewall). Bij die firewall sta je de protocollen HTTP en HTTPS met de volgende twee commando's toe:

```
sudo ufw allow http
sudo ufw allow https
```

5. Basiswebsite

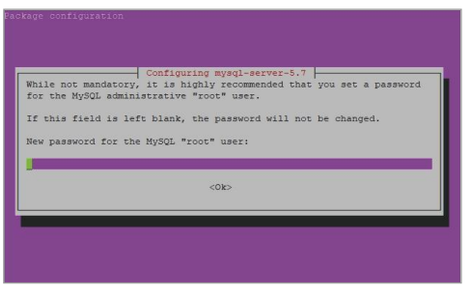
Met het volgende commando krijg je een statusoverzicht van de webserver. Als achter 'Active' in het groen 'active (running)' staat, is de webserver geactiveerd.

```
sudo systemctl status apache2
```

In principe ben je dan al klaar. Als je met een browser naar het IP-adres van je server gaat of naar de url van een domeinnaam die je aan het IP-adres hebt gekoppeld, moet de standaardpagina van Apache2 verschijnen.

6. Databaseserver installeren

De meeste sites bestaan niet uit passieve HTML-pagina's, maar halen hun steeds veranderende content uit een database. Om de content en andere gegevens in een database op te kunnen slaan, installeer je als databaseserver MySQL (of MariaDB).




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

Tijdens het installeren wordt gevraagd naar een wachtwoord voor de databasegebruiker. Nadat je dat ter bevestiging nog een keer ingetypt hebt, is de databaseserver klaar voor gebruik. Met

```
sudo systemctl status mysql
```

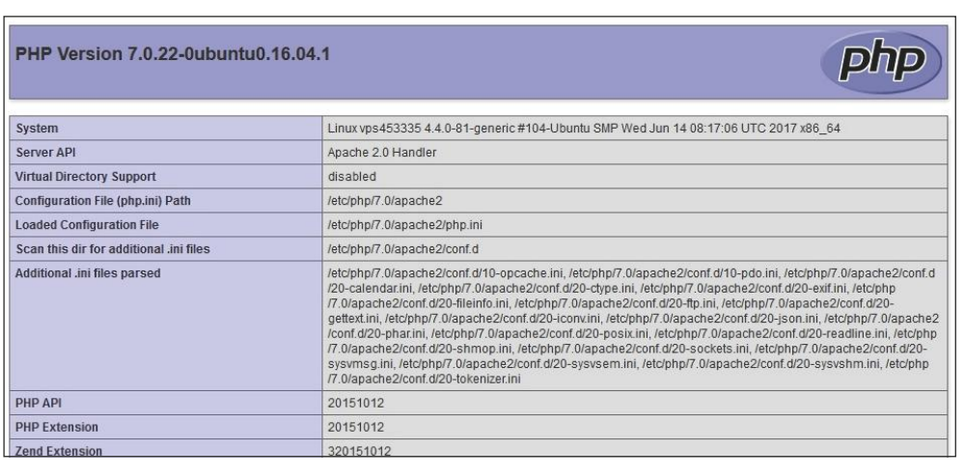
kun je controleren of dat inderdaad het geval is. Onthoud het databasewachtwoord goed, want dat heb je later regelmatig nodig, bijvoorbeeld om back-ups of nieuwe tabellen aan te maken.

Apache 2

Op dit moment is versie 2.4.27 de stabiele versie van webserver Apache, maar 2.8 is in de maak. Hij is onderdeel van WAMPP-, LAMP- en XAMPP-stacks en daardoor beschikbaar voor alle platforms. Het installeren op een Linux-server blijft echter de makkelijkste manier om hem te beheren.

Boven
Het PHP-testbestand bevat maar een paar regels om de status van de PHP-installatie te laten zien.

Rechts
Als PHP op de webserver geïnstalleerd is, levert een aanroep van het PHP-testbestand in de browser dit beeld op.



PHP Version 7.0.22-0ubuntu0.16.04.1

System	Linux vps453335 4.4.0-81-generic #104-Ubuntu SMP Wed Jun 14 08:17:06 UTC 2017 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012

SSL op meerdere manieren

In het artikel werk je met Let's Encrypt om op een makkelijke manier certificaten voor je server aan te maken. Je krijgt dan erkende certificaten die een browser als veilig accepteert. Je kunt ook zelf aan de slag met het aanvragen en verwerken van een certificaat. Er zijn meerdere commerciële partijen die je daarbij willen en kunnen helpen. Een certificaat moet sowieso van een vertrouwde partij afkomen. De prijzen daarvoor lopen uiteen van een paar tientjes tot meer dan honderd euro per jaar. Zo'n certificaat geldt dan voor één enkel domein, dus als je meerdere websites op je server host, wordt het al snel een prijzige aangelegenheid. Soms worden er wel lagere prijzen gehanteerd voor particuliere websites, maar dan nog kan het aardig aantikken. Het loont om even te kijken of de hoster bij wie je de webserver huurt zelf mogelijkheden heeft om certificaten te genereren.

De certificaten en hun onderdelen die je dan krijgt, moet je wel op de server installeren. Dat is dan gelukkig niet meer dan een kwestie van op de juiste plek zetten. Daarna moet je bij verschillende configuratiebestanden nog wel aangeven dat die certificaten gebruikt moeten worden. Dat begint met het toevoegen van poort 443 voor het veilige verkeer en verwijzingen naar de certificaatbestanden aan het bestand /etc/apache2/sites-available/test.com.conf.

```
<VirtualHost *:443>
    ServerAdmin <yourname>@test.com
    DocumentRoot /var/www/test.com
    SSLCertificateFile /etc/ssl/
certs/<certificatename>.pem
    SSLCertificateKeyFile /etc/ssl/
private/<certificatename>.key
    ServerName test.com
    ServerAlias www.test.com
    ErrorLog ${APACHE_LOG_DIR}/
error.log
    CustomLog ${APACHE_LOG_DIR}/
access.log combined
</VirtualHost>
```

Na uitvoeren van de commando's `sudo a2enmod ssl` en `sudo service apache2 restart` moet je HTTPS-website nu een groen slotje hebben. Dan blijft de HTTP-versie ook nog beschikbaar, dus die moet je handmatig door laten schakelen (redirect) in het VirtualHost-deel:

```
<VirtualHost *:80>
    ServerName test.com
    ServerAlias www.test.com
    DocumentRoot /var/www/test.com
    Redirect / https://test.com
</VirtualHost>
```

Een in eerste instantie elegante tussenoplossing is het werken met het ingebouwde certificaat van Apache zelf. Verander de regels

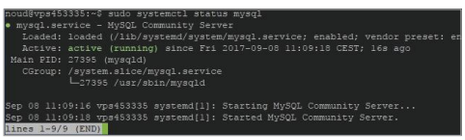
```
SSLCertificateFile /etc/ssl/
certs/<certificatename>.pem
SSLCertificateKeyFile /etc/ssl/
private/<certificatename>.key
```

in

```
SSLCertificateFile /etc/ssl/certs/
ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/
ssl-cert-snakeoil.key
```

om hem te gebruiken. Dat heeft een nadeel: er is wel HTTPS-verkeer mogelijk, maar het certificaat is niet ondertekend door een Certificate Authority, wat resulteert in een geel slotje. Die tussenoplossing is daarom goed genoeg om mee te testen, maar zal op de lange termijn geen vertrouwen uitstralen.

De gratis oplossing van Let's Encrypt is daardoor optimaal: niet alleen zijn de kosten minimaal (een donatie op **https://letsencrypt.org** is altijd welkom), maar de inspanningen die je moet verrichten om het aan het werk krijgen, zijn eveneens minimaal.



7. PHP installeren

Voor het installeren van de taal PHP moet je wat extra modules toevoegen. Vandaar dat dit commando wat langer is dan de commando's die je tot nu toe ingetypt hebt:

```
sudo apt-get install php libapache2-mod-php
php-mcrypt php-mysql php-cgi php-curl php-json
```

8. PHP testen

Om te testen of PHP goed geïnstalleerd is, moet je eerst een PHP-pagina hebben die je uit kunt voeren. Om zo'n pagina te maken, heb je een editor voor het bewerken van tekstbestanden nodig.. Voor beginners is nano het eenvoudigst, maar gevorderde gebruikers geven vaak de voorkeur aan ViM, Emacs o.i.d. Welke je ook wilt gaan gebruiken, je moet hem eerst installeren – het is immers een minimale Ubuntu-installatie. We gaan hier uit van de editor nano:

```
sudo apt-get install nano
```

Daarna kun je een testpagina maken. Bij Apache staan de bestanden op de webserver standaard in de directory /var/www/html. In die directory maak je een testpagina met de extensie php aan:

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

Vervolgens typ je daar de volgende drie regels in:

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

Bewaar het bestand (Ctrl+O) en sluit de editor af (Ctrl+X). Ga met de browser dan naar http://<mijnserver>/test.php. Als het goed is, krijg je nu de systeeminformatie van PHP te zien. Omdat je waarschijnlijk niet wilt dat iedereen alle informatie over jouw webserver kan bekijken, verwijder je die pagina na het testen met het commando *rm*:

```
sudo rm /var/www/html/test.php
```

9. Een echte website

Meestal wil je meerdere websites op een enkele webserver hosten. Die komen dan allemaal in een eigen directory te staan en werken met een eigen database, zodat ze elkaar niet in de weg zitten. In dit geval maak je met het commando *mkdir* een directory aan voor de website test.com, waarna je met het commando *cd* naar die directory gaat en daar met nano het bestand index.html aanmaakt dat standaard geopend wordt als iemand een website bezoekt.

```
sudo mkdir -p /var/www/test.com
cd /var/www/test.com
sudo nano index.html
```

In dat index-bestand zet je met nano de volgende content:

```
<html>
<head>
<title>test.com index-page</title>
</head>
<body>
<h1>Hello and welcome to test.com!</h1>
<h2>If you see this, then it works ...</h2>
</body>
</html>
```

10. Website toevoegen

Je hebt een aparte directory aangemaakt voor de nieuwe website, maar Apache weet nog niet wat hij daar mee moet. In de directory /etc/apache2/sites-available/ staan de configuratiebestanden voor de websites die door je Apache-server gehost worden. In die directory moet het configuratiebestand voor het nieuwe domein test.com aangemaakt worden.

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

In het configuratiebestand zet je de volgende regels. Bij ServerAdmin vul je je mailadres in.

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

11. Website aanmelden

Daarna moet die configuratie nog toegevoegd en geladen worden. Met *a2ensite* (Apache 2 enable site) voeg je de site aan de webserver toe en met het *reload*-commando laad Apache de configuratie opnieuw in. Hierna zou de site beschikbaar moeten zijn.

```
sudo a2ensite test.com.conf
sudo service apache2 reload
```

12. Website testen

Als je met de browser nu naar http://<mijnserver>/test.com gaat, moet je de indexpagina zien. Als je de domeinnaam test.com geregistreerd hebt en die naar het IP-adres van jouw webserver verwijst, moet je hetzelfde resultaat krijgen als je naar http://test.com gaat.

Apache of Nginx?

Apache is niet de enige webserver die je makkelijk zelf kunt installeren. Nginx (spreek uit engine-x) werkt op een vergelijkbare manier. Online benchmarks lijken erop te wijzen dat Nginx betere performance levert dan Apache (Zie **bit.ly/apachenginx**). Daar staat tegenover dat Apache langer bestaat, out-of-the-box meer kan en beter wordt ondersteund door third parties. Volgens cijfers van Netcraft's Web Server Survey worden de verschillen echter steeds kleiner, zie **bit.ly/netsept17**.

13. SSL installeren

Het internetverkeer tussen je webserver en de browser van je bezoekers is nog niet versleuteld. Iedereen die toegang heeft tot dat netwerkverkeer kan daarom alles meelezen en zien wat er verstuurd wordt. Bij pagina's met inloggegevens of betaalinformatie wil je dat natuurlijk niet. Vandaar dat we dit verkeer moeten versleutelen. Dat kan tegenwoordig op meerdere manieren: een moeilijke, een redelijk makkelijke en een erg makkelijke. In een kader elders in dit artikel staat meer informatie over de moeilijke en redelijk makkelijke manier, hier gaan we voor de makkelijkste – en gratis – methode met Let's Encrypt. Je kunt Let's Encrypt de certificaten laten regelen die nodig zijn voor het versleutelen en ook alle instellingen laten doorvoeren die nodig zijn om het internetverkeer te beveiligen:

```
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get update
sudo apt-get install python-certbot-apache
sudo certbot --apache -d test.com -d
www.test.com
```

Daarbij wordt het bestand test.com-le-ssl.conf in de directory /etc/apache2/sites-available/ gemaakt, waarin poort 443 voor het veilige SSL-verkeer wordt geconfigureerd. Het normale dataverkeer via HTTP gebruikt poort 80, maar het protocol HTTPS (waarbij de S inderdaad van Secure is) werkt met poort 443. Gelukkig zijn zowel HTTP als HTTPS bij stap 4 al vrijgegeven door de firewall, zodat het veilige dataverkeer niet wordt tegengehouden.

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
        ServerAdmin [redacted]
        DocumentRoot /var/www/test.com
        ServerName test.com
        ServerAlias www.test.com
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/accesslog combined
SSLCertificateFile /etc/letsencrypt/live/test.com/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/test.com/privkey.pem
Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
</IfModule>
```

14. Alleen veilig verkeer

Voor een optimale beveiliging wil je natuurlijk dat al het verkeer dat via HTTP bij je webserver binnenkomt omgeleid wordt naar HTTPS. Ook dat stelt Let's Encrypt voor je in bij het configuratiebestand test.com.conf van je website.

```
<VirtualHost *:80>
        ServerAdmin [redacted]
        DocumentRoot /var/www/test.com
        ServerName test.com
        ServerAlias www.test.com
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/accesslog combined
RewriteEngine on
RewriteCond %{SERVER_NAME} =www.test.com [OR]
RewriteCond %{SERVER_NAME} =test.com
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

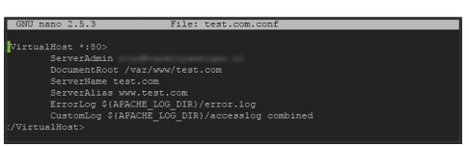
Als je vervolgens met je browser naar **https://www.ssllabs.com/ssltest/analyze.html?d=test.com&latest** gaat, kun je testen of je site inderdaad goed beveiligd is. Als je met je browser naar http://test.com gaat, zul je zien dat je automatisch wordt doorgestuurd naar https://test.com. En belangrijker: je krijgt naast de url een groen slotje te zien ter bevestiging van de beveiligde status van het webverkeer.

15. Automatisch verlengen

Een Let's Encrypt-certificaat heeft een maximale geldigheid van 90 dagen, waarna je het certificaat normaal gesproken handmatig moet verlengen. Dat is voor één website nog wel te doen, maar beheer je meerdere sites dan wil je niet het risico lopen dat je er toevallig een vergeet te verlengen, waardoor je site ineens niet meer veilig te bereiken is. Gelukkig is er een mogelijkheid het certificaat automatisch te laten verlengen.

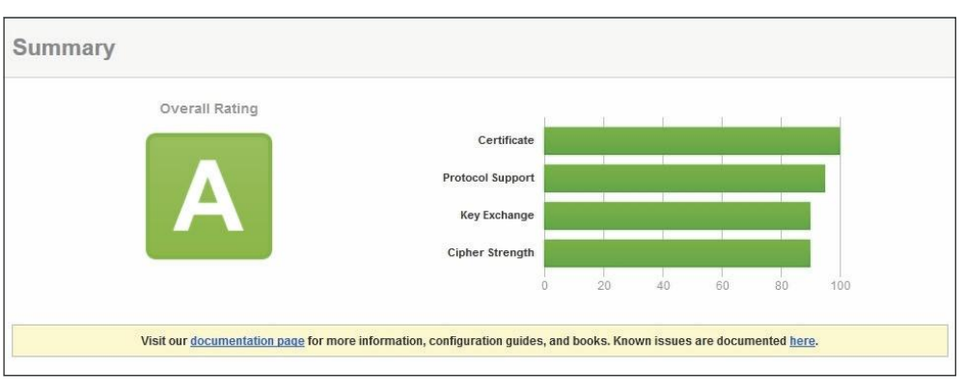
Hiervoor maken we een systeemtaak aan, in Linux cronjob of crontab geheten. Met Cron (afkomstig van het Engelse woord chronograph, een soort stopwatch) kun je taken op een vooraf ingesteld tijdstip uitvoeren. Onderaan de lijst bestaande taken voeg je een nieuwe taak toe voor het verlengen van de certificaten. In dit geval gebeurt dat elke ochtend om 07:00. Als een certificaat nog niet verlengd hoeft te worden, gebeurt er verder niets.

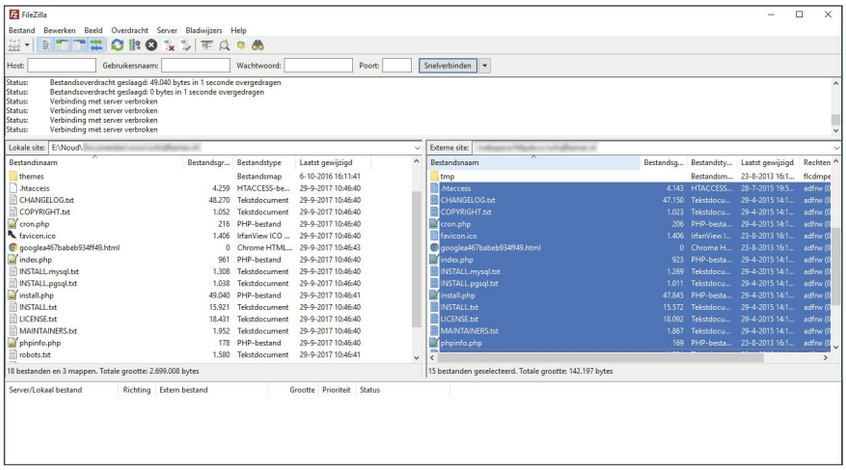
```
sudo crontab -e
0 7 * * * /usr/bin/certbot renew --quiet
```



Boven
Het configuratiebestand voor het domein test.com geeft aan waar de bestanden van de site op de webserver staan.

Links
Bij **www.ssllabs.com** test je of de site het dataverkeer goed versleuteld aanlevert en de gegevens beschermd zijn tegen meekijkers.





File Transfer Protocol

In dit artikel heb je gezien hoe je met SSH en een console op een webserver commando's uitvoert. Maar uiteindelijk moet een website gevuld worden met content in de vorm van teksten en afbeeldingen. Om HTML-, JPG- en andere bestanden naar je webserver te kopiëren, wordt het File Transfer Protocol (FTP) gebruikt. Op de webserver moet je een FTP-server installeren die bepaalde gebruikers toegang geeft tot bepaalde directory's op de webserver. Die gebruikers en de directory's moet je zelf configureren.

Als dat gedaan is, maak je met een FTP-programma op een computer contact met de FTP-server op je webserver met een gebruikersnaam en wachtwoord, waarna je met dat programma – bijvoorbeeld FileZilla – bestanden van en naar de webserver kunt kopiëren.

In het volgende nummer gaan we uitgebreid in op het installeren en configureren van zo'n FTP-server. Als je deze workshop redelijk hebt kunnen volgen, zal dat niet al te veel problemen opleveren.

16. Databasebeheer installeren

Nu al het verkeer van en naar je webserver versleuteld is, kun je verder aan de slag met de databaseserver. Die heb je al wel geïnstalleerd, maar er is nog geen beheerinterface voor. Daarvoor wordt meestal phpMyAdmin gebruikt. Installeer deze software met de volgende commando's:

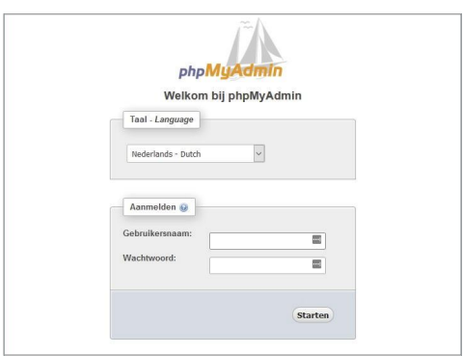
```
sudo apt-get update
sudo apt-get install phpmyadmin php-mbstring
php-gettext
```

17. Databasebeheer inloggen

Bij het eerste scherm selecteer je apache2 door op de spatiebalk te drukken en bij de vraag naar dbconfig-common zeg je 'yes'. Daarna moet je een wachtwoord invullen om bij phpMyAdmin in te kunnen loggen. De modules mcrypt en mbstring moet je nog even expliciet toestaan omdat phpMyAdmin die nodig heeft en ze niet automatisch geïnstalleerd worden. Daarna moet je Apache herstarten:

```
sudo phpenmod mcrypt
sudo phpenmod mbstring
sudo systemctl restart apache2
```

Vervolgens kun je in de browser naar **https://test.com/phpmyadmin** en wordt je gevraagd in te loggen. Gebruik daarbij het wachtwoord dat je net hebt opgegeven. Als gebruikersnaam gebruik je root.



18. Database afschermen

Door als root bij phpMyAdmin in te loggen, ga je als een soort super-user de database beheren. Dat is net als bij het beheer van de webserver in de praktijk absoluut niet wenselijk, zeker niet omdat die interface nog wel eens doelwit van aanvallers is. Als het kwaadwillenden lukt om de interface te hacken, hebben ze totale toegang tot al je databases. Dat voorkom je door phpMyAdmin beter af te schermen. Open het configuratiebestand van phpMyAdmin met

```
sudo nano /etc/apache2/conf-available/
phpmyadmin.conf
```

En voeg de regel met AllowOverride daaraan toe om de standaardinstellingen later te overrulen:

```
<Directory /usr/share/phpmyadmin>
    Options FollowSymLinks
    DirectoryIndex index.php
    AllowOverride All
    . . .
```

Herstart de Apache-server voor de nieuwe instellingen:

```
sudo systemctl restart apache2
```

19. Gebruikers toevoegen

Daarna ga je ervoor zorgen dat alleen expliciet opgegeven gebruikers bij phpMyAdmin mogen aanmelden. Daar maak je een .htaccess-bestand in de phpMyAdmin-directory voor aan

```
sudo nano /usr/share/phpmyadmin/.htaccess
```

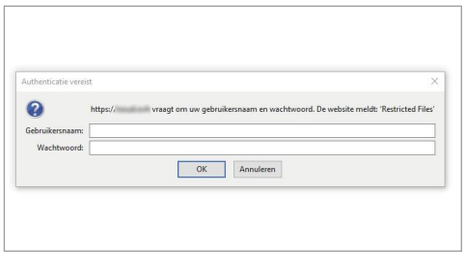
met daarin de volgende content:

```
AuthType Basic
AuthName "Restricted files"
AuthUserFile /etc/phpmyadmin/.htpasswd
Require valid-user
```

Sla het bestand op en maak het genoemde bestand .htpasswd aan met daarin de toegestane gebruikers:

```
sudo apt-get install apache2-utils
sudo htpasswd -c /etc/phpmyadmin/.htpasswd
<newusername>
```

voor de eerste gebruiker en voor iedere volgende gebruiker het tweede commando nog een keer, maar dan zonder -c. Als je dan naar https://test.com/phpmyadmin gaat, zul je eerst als een van die gebruikers moeten inloggen om bij de interface te mogen.



20. Webserver klaar

Dan is de basisconfiguratie van je webserver klaar en je website klaar voor gebruik. Als je met een contentmanagementsysteem (CMS) aan de slag wilt, kun je met phpMyAdmin een database voor het CMS-pakket (bijvoorbeeld WordPress, Jooma of Drupal) aanmaken en de gegevens daarvan bij het installeren van dat pakket opgeven. In principe heb je nu een veilige webserver met alle basisfuncties – maar ook de mogelijkheid die functionaliteit uit te breiden waar, hoe en wanneer je dat zelf wilt.

21. Bestanden naar de server

Er blijven natuurlijk altijd dingen over om te doen. Je basisconfiguratie is nu wel klaar, maar hoe krijg je daar nu bestanden op? Als je bestanden op je webserver wilt zetten, zul je daar bijvoorbeeld eerst een FTP-server op moeten installeren. In het kader hierboven staat daar wat meer over, maar in het volgende nummer gaan we daar verder op in.