STEP BY STEP

SETTING UP WORDPRESS ON A VPS FOR BEGINNERS

UDEMY COURSE COMPANION

This eBook is a companion guide to the Udemy Course: **Ubuntu 16.04**

Step by Step - Setting Up WordPress on a VPS for Beginners

If you have any questions, I can be reached on the Udemy Course support page.

CONTENTS

SOFTWARE

If you have never used terminal or the command line before, there is no need to stress as I will be guiding you through the entire process.

LINUX and MACOSX

Linux and **MACOSX** users only need to use **terminal**.

Windows All Versions

If you are using Windows, you need to download and use putty, puttygen, pageant and pscp.

Putty is the windows equivalent of terminal.

Puttygen is used to create and save ssh keys.

Pageant is used for ssh key management.

Pscp is used to copy files securely from your pc to your server.

All the above tools are free to download and use. I have created a video, just for Windows users, showing you all the steps from downloading putty, creating keys with puttygen to using pscp and key management with pageant.

Copying and Pasting using putty

Drag over any characters you want to copy and those characters will be copied to the clipboard. To paste, right click anywhere in the putty window.

LINUX, MACOSX and Windows

The last item required is **FileZilla**. It's a cross platform, meaning its available for LINUX, MACOSX and Windows. It's used for uploading and downloading files from your server. We will be using FileZilla to sFTP to the VPS. We will not be using ftp.

INITIAL SECURITY STEPS

First Login as the ROOT user

We need to change the root password that your VPS host emailed to you.

```
passwd
```

We are going to add a new user that will be given elevated privileges for when we need them. After the adduser command leave a space and type your desired username.

```
adduser
```

Type a password for the new user and the press enter at each question, then finally adding a y(yes) at the end.

The new user needs elevated privileges.

Some distributions don't have nano installed, we may need to install the editor nano. It will make no difference if nano is already installed. We do not need to type sudo, as we are working as the root user.

```
apt-get install nano
visudo
```

We need to prevent anyone from logging in as the root user, edit the file sshd_config, using nano and change the line PermitRootLogin to no

```
nano /etc/ssh/sshd_config
```

Let's restart ssh and logout to apply the changes we have made.

```
systemctl restart ssh && logout
```

First Login as a NON ROOT User

Login again as the user you just created and update the VPS. You will not be able to login as the root user.

```
sudo apt-get update && sudo apt-get upgrade
```

If certain packages are being kept back, you need to perform a dist-upgrade. You may need to reboot after the dist-upgrade command

```
sudo apt-get dist-upgrade
sudo reboot
```

After the reboot, login again.

Ensure you are in your home directory and create a directory called .ssh

```
cd
mkdir .ssh
```

Logout of your VPS

```
logout
```

SECURING SSH

SSH Key Authentication

MACOSX - View the video, creating ssh keys on MACOSX

```
ssh-keygen -t rsa -b 4096
```

WINDOWS - View the video, creating ssh keys using puttygen

```
Use puttygen, there is a tutorial regarding puttygen.
```

Once keys have been generated, copy the public key to you VPS. I'm referring to the public key as rsa_id.pub. Replace that name with the name you saved your public key as.

MACOSX

```
scp rsa_id.pub user@ip.address:/home/username/.ssh
```

WINDOWS - use the command prompt - Windows Key +R type CMD

```
pscp rsa_id.pub user@ip.address:/home/user/.ssh
```

Login to your vps

Change to the .ssh directory and create an authorized_keys file in the .ssh directory

```
cd .ssh
ls -l
touch authorized_keys
```

Concatenate the pub_id file with authorized keys files

```
cat id_rsa >> authorized_keys
```

Delete the id_rsa file

```
rm id_rsa.pub
```

Set permissions on authorized_keys file, very strict READ-ONLY for owner

```
sudo chmod 400 authorized_keys
```

Set permissions on .ssh directory, READ, WRITE and EXECUTE for owner only.

```
sudo chmod 700 /home/user/.ssh
```

Set an immutable bit on .ssh directory

```
sudo chattr +i ~/.ssh
```

If the chattr command is not found, install it then execute the above command

```
apt-get install e2fsprogs
```

Edit the file sshd_config using nano and

```
nano /etc/ssh/sshd_config
```

uncomment AuthorizedKeysFile

uncomment PasswordAuthentication and **change to no**

Let's restart ssh and logout to apply the changes we have made.

```
sudo systemctl restart ssh && logout
```

After you have logged out, you can login again using ssh key authentication.

There are videos showing you how to login using key authentication.

At this stage you have secured access to your VPS.

No one can access your VPS using a password, including yourself. Only the holder of the private key can login. Do not lose the private key, copy it to a USB stick.

THE FIREWALL

Please refer to the text file for the firewall rules. Do not copy and paste the rules from this document as the formatting as incorrect.

What we need to do here is open certain ports and close those ports we don't need.

For example, our webserver is going to operate on ports 80 and 443 and ssh on port 22. So at this stage we can close all other ports. If other services are added, we will need to revise these rules.

Check your default firewall rules by entering the following command:

```
sudo iptables -L
```

As we implemented any firewall rules yet, you will see an empty firewall ruleset.

```
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
```

Let's create a file to hold our firewall rules.

```
sudo nano /etc/iptables.firewall.rules
```

This is a basic ruleset, that will get you started. It will allow traffic to the following services and ports: http (80), https (443), ssh (22), and ping. All other services and ports will be blocked.

Add the following rules in the iptables.firewall.rules file you just created

Please refer to the text file, in the resources section for the firewall rules, do not copy and paste the rules from this document.

```
*filter #

Allow all loopback (lo0) traffic and drop all traffic to 127/8 that doesn't use lo0
-A INPUT -i lo -j ACCEPT
-A INPUT -d 127.0.0.0/8 -j REJECT

# Accept all established inbound connections
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allow all outbound traffic - you can modify this to only allow certain traffic
-A OUTPUT -j ACCEPT

# Allow HTTP and HTTPS connections from anywhere (the normal ports for websites and SSL)
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT
# Allow SSH connections
#
# The -dport number should be the same port number you set in sshd_config
#
-A INPUT -p tcp -m state --state NEW --dport 22 -j ACCEPT

# Allow ping
-A INPUT -p icmp --icmp-type echo-request -j ACCEPT

# Log iptables denied calls
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables denied: " --log-level 7

# Drop all other inbound - default deny unless explicitly allowed policy
-A INPUT -j DROP -A FORWARD -j DROP COMMIT
```

The rules will allow traffic to the following services and ports: HTTP (80), HTTPS (443), SSH (22), and ping. All other ports will be blocked.

Save the changes to the firewall rules file by pressing Control-X, and then Y.

Activate the firewall rules by entering the following command

```
sudo iptables-restore < /etc/iptables.firewall.rules
```

Recheck your firewall rules by entering the following command:

```
sudo iptables -L
```

The new firewall ruleset

```
Chain INPUT (policy ACCEPT)
target prot opt source       destination
ACCEPT all -- anywhere anywhere
REJECT all -- anywhere 127.0.0.0/8    reject-with icmp-port-unreachable
ACCEPT all -- anywhere anywhere       state RELATED,ESTABLISHED
ACCEPT tcp -- anywhere anywhere       tcp dpt:http
ACCEPT tcp -- anywhere anywhere       tcp dpt:https
ACCEPT tcp -- anywhere anywhere       state NEW tcp dpt:ssh
ACCEPT icmp -- anywhere anywhere
LOGall -- anywhere anywhere  limit: avg 5/min burst 5 LOG level debug prefix "iptables denied: "
DROP all -- anywhere anywhere

Chain FORWARD (policy ACCEPT)
target prot opt source       destination
DROP all -- anywhere anywhere

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

We need to ensure that the firewall rules are activated every time you start or restart our vps.

Create a new script with the following command:

```
sudo nano /etc/network/if-pre-up.d/firewall
```

Copy and paste the following lines in to the file you just created

```
#!/bin/sh
/sbin/iptables-restore < /etc/iptables.firewall.rules
```

Set the script’s permissions to executable by entering the following command

```
sudo chmod +x /etc/network/if-pre-up.d/firewall
```

APACHE

Before making any changes to any configuration file, make a backup copy first! It's just a safeguard in case disaster strikes. All optimization tweaks are based on a VPS with 1Gig of memory.

Installation

```
sudo apt-get install apache2
```

Hardening

```
cd /etc/apache2/conf-available/
sudo cp security.conf security.conf.bak
sudo nano security.conf
```

Change the following to send minimal information about the server in the event of an error.

```
ServerTokens Prod
ServerSignature Off
TraceEnable Off
```

Optimize

```
cd /etc/apache2/
sudo cp apache2.conf apache2.conf.bak
sudo nano apache2.conf
```

Add the following to the end of your apache2.conf file

```
<IfModule mpm_prefork_module>
StartServers 2
MinSpareServers 6
MaxSpareServers 12
MaxClients 30
MaxRequestsPerChild 3000
</IfModule>
```

In the apache.conf file, in the <Directory /var/www> section, change the AllowOverride option to All. This will allow us to use .htaccess files with WordPress.

```
<Directory /var/www/>
Options Indexes FollowSymLinks
AllowOverride All
Require all granted
</Directory>
```

We also need to disable directory browsing and enable mod rewrite for when we install WordPress later on.

```
sudo a2dismod autoindex
sudo a2enmod rewrite
```

Restart apache

```
sudo systemctl reload apache2
```

MySQL

Installation

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

Harden

Remove anonymous user accounts, disable remote root login, and remove the test database

```
sudo mysql_secure_installation
```

Rename the MySQL root user

```
mysql -u root -p
rename user 'root'@'localhost' to 'some_user'@'localhost';
flush privileges;
```

Optimize

The configuration of mysql on Ubuntu 16.04 is suitable, no further optimization is needed.

I'll be adding a lecture on using mysql-tuner shortly.

Installation

```
sudo apt-get install php libapache2-mod-php php-mcrypt php-curl php-gd
php-mbstring php-mcrypt php-xml php-xmlrpc
```

All the above mdules must be typed on a single line, as per the lectures.

After installation of the php modules

```
cd /etc/apache2/mods-available
ls
sudo cp dir.conf dir.conf.bak
sudo nano dir.conf
```

The file dir.conf will look like this

```
<IfModule mod_dir.c> DirectoryIndex index.html index.cgi index.pl
index.php index.xhtml index.htm </IfModule>
```

We need to move the index.php entry to the beginning. Just delete the existing index.php and add it to the beginning.

```
<IfModule mod_dir.c> DirectoryIndex index.php index.html index.cgi
index.pl index.xhtml index.htm </IfModule>
```

Restart apache

```
sudo systemctl reload apache2
```

At this stage we need to test the php installation

Firstly, lets create a file called info.php in the web root of the server

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

Add the following in the file info.php

```
<?php

phpinfo();

?>
```

Visit your site using your browser. You can use the IP address of your server, e.g. http://51.200.145.254/info.php, replace the IP with your IP address.

Harden and Optimize

```
cd /etc/php/7.0/apache2/
sudo cp php.ini php.ini.bak
sudo nano php.ini
```

Make the following changes in the php.ini file. Most of these settings will match the defaults, check all of them anyway! php.ini is a large file, use ctrl w and search for the entries. If you are going to create a forum, leave the open_basedir setting blank. Most forums will not work if you specify the open_basedir.

```
open_basedir=/var/www/:/tmp/
expose_php=Off
max_execution_time=30
max_input_time=60
max_input_nesting_levels=64
memory_limit=64M
error_reporting = E_All
display_errors=Off
log_errors=On
post_max_size=8M
upload_max_file_size=2M
allow_url_fopen=off
allow_url_include=Off
```

The upload_max_file=2M depends on the type of WordPress site you are going to run. if you are uploading high resolution photos or HD videos to your blog, increase this value and decrease it when you are done.

Opcache is enabled by default in 16.04.

```
sudo systemctl reload apache2
```

We need to test the php installation again. Visit your site again, you can use the IP address of your server, e.g. http://51.200.145.254/info.php

You must remove the file, info.php, back to terminal and delete it.

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

FAIL2BAN

Fail2ban is an intrusion prevention software framework which protects computer servers from brute-force attack.

Brute-force break-in attempts are quite frequent against an SSH server and other password protected internet-services (such as ftp,pop,...). Automated scripts try multiple combinations of username/password (brute-force, dictionary attack) and sometimes changing the port to something other than the default can't be done. Furthermore, scouring your log files yourself is not only time consuming, but can be difficult too. Fail2ban attempts to alleviate these issues by providing an automated way of not only identifying possible break-in attempts, but acting upon them quickly and easily in a user-definable manner.

If you run a WordPress site fail2ban is your best friend. As soon as someone scans your site, they are blocked for an amount of time you specify.

Installation

```
sudo apt-get install fail2ban
```

Configuration

Make a copy of the jail.conf file and call it jail.local. We can then edit the file using nano.

```
cd /etc/fail2ban/
sudo cp jail.conf jail.local
sudo nano jail.local
```

Now we can edit the jail.local file. We need to change a few values to configure what we need

```
# "bantime" is the number of seconds that a host is banned.
bantime = 600000

# A host is banned if it has generated "maxretry" during the last "findtime"
findtime = 600
maxretry = 3
```

The debian-defaults file: /etc/fail2ban/jail.d/

```
[apache-auth]
enabled = true

[apache-badbots]
enabled = true

[apache-noscript]
enabled = false

[apache-overflows]
enabled = true

[apache-nohome]
enabled = true

[apache-botsearch]
enabled = true

[apache-fakegooglebot]
enabled = false

[apache-shellshock]
enabled = true
```

Ensure these two jails are set to FALSE and NOT true. Ignore the status in the lectures, SET JAILS TO FALSE

The defaults in the jail.local file can be overridden in the debian-defaults file.

```
bantime = seconds
maxretry = attempts
```

After making any changes, you must restart fail2ban.

```
sudo systemctl restart fail2ban
```

VIRTUAL HOSTS

Apache Virtual Hosts

Using virtual hosts, you can run multiple websites on a single vps. As soon as one site gets lots of traffic and starts to consume resources, it's easy to move it to another vps or upgrade your vps with more CPU and RAM.

From a previous video, our web root is /var/www/html and it is owned by root. We are shortly going to disable the default apache site and give each site its own directory. Each site will be owned by our user and the group www-data. www-data is the web server group.

At this stage you should purchase a domain name or transfer an existing name. The next section will cover DNS settings and pointing your domain name(s) to your vps.

Please note that we are creating a directory domain.com/public_html/ All our site files will be inside public_html.

We need to create the directories for our sites.

I always create a public_html directory inside the site name directory. I do this so we can move wp-config.php one level back. That way wp-config.php is out of "public" access.

```
sudo mkdir -p /var/www/domain_name1.com/public_html
sudo mkdir -p /var/www/domain_name2.com/public_html
sudo mkdir -p /var/www/domain_name3.com/public_html
```

Change to the sites-available directory and then copy the 000-default apache site, creating a new file for each site.

```
cd /etc/apache2/sites-available

sudo cp 000-default.conf domain_name1.com.conf
sudo cp 000-default.conf domain_name2.com.conf
sudo cp 000-default.conf domain_name3.com.conf
```

Now we need to edit each of the domain_name.conf files, changing as the information as shown below.

```
sudo nano domain_name1.com.conf
sudo nano domain_name2.com.conf
sudo nano domain_name3.com.conf
```

Edit each .conf file you created and add the following, under the # However, to each, changing the domains as needed.

```
<VirtualHost *:80>


# However, you must set it for any further virtual host explicitly.

ServerName domain_name.com
ServerAlias www.domain_name.com
ServerAdmin webmaster@domain_name.com
DocumentRoot /var/www/domain_name/public_html


</VirtualHost>
```

Finally, we must enable the new sites and disable the default apache site.

When you enable or disable an apache module or site, it's easy to remember:
apache2 enable site - a2ensite
apache2 disable site - a2dissite
apache2 enable module - a2enmod
apache2 disable module - a2dismod

```
sudo a2ensite domain_name1.conf
sudo a2ensite domain_name2.conf
sudo a2ensite domain_name3.conf
sudo a2dissite 000-default.conf
```

Restart apache to enable the changes.

```
sudo systemctl reload apache2
```

At this stage you must change the @ value with your domain registrar. The videos on DNS will show you how.

htop

You need to install htop to monitor server resources

```
sudo apt-get install htop
```

To run htop, type htop at the prompt

```
htop
```

Log Files

The log files are owned by root and the group adm. You must add your user to the adm group to be able to view and download the log files. The log files are located in the /var/log directory and the apache logs are located in the /var/log/apache2 directory. Use filezilla to download your log files and view them with your text editor.

After using the usermod command you must logout and login to enable the change. An explanation: I have no idea what username you are using, so $USER will refer to the non root user you added. You can use $USER instead of typing your user. $USER is the same as typing the current non root username.

```
sudo usermod -a -G adm $USER
logout
```

Mail Utils

To view system mail messages, you must install mailutils.

```
sudo apt-get install mailutils
```

To view system mail messages, type mail at the prompt.

```
mail
```

SendMail

WordPress needs sendmail to be installed, we can install it now.

```
sudo apt-get install sendmail
```

MYSQL DATABASES

Creating a Database

You need to login into MySQL to manage your databases. Earlier I recommended you change the root MySQL username to something else. Use that name, I'll be using root in the next few examples.

What's important when typing MySQL commands is to remember that they end in a semi-colon ; If you forget the semi-colon, type it on the next line and press enter. The apostrophe is also important, don't forget it. The commands can also be typed in lowercase.

SHOW DATABASES, CREATE DATABASE, GRANT ALL PRIVILEGES, SHOW GRANTS, FLUSH PRIVILEGES and exit

Login into MySQL and type your root MySQL password

```
mysql -u root -p
```

To view the default databases

```
SHOW DATABASES;
```

To **create** the database needed for the first WordPress site

```
CREATE DATABASE db_name;
```

Now you need to **grant privileges** on the database your created to a user and identify that user using a password.

```
GRANT ALL PRIVILEGES ON db_name.* TO 'username'@'localhost' IDENTIFIED BY 'password';
```

To confirm the above command, we use the command show grants

```
SHOW GRANTS FOR 'username'@'localhost';
```

To **refresh** our databases, we need to flush them.

```
FLUSH PRIVILEGES;
```

To **exit** MySQL

```
exit
```

Quick and Easy WordPress Databases

As you have seen in the videos, type the commands in your text editor as follows:

```
CREATE DATABASE db_name; GRANT ALL PRIVILEGES ON db_name.* TO
'username'@'localhost' IDENTIFIED BY 'password'; FLUSH PRIVILEGES;
```

The commands with different values

```
CREATE DATABASE mylist; GRANT ALL PRIVILEGES ON mylist.* TO
'shopping'@'localhost' IDENTIFIED BY '23fbt54rhc'; FLUSH PRIVILEGES;
exit
```

By copying and pasting that line at the MySQL prompt, your database has been setup. You must obviously login to MySQL first before pasting.

If you would like to **delete** a database, use the DROP command.

```
DROP DATABASE DB_NAME;
```

Backup a Database - create a mysqldump file

We use the mysqldump command to backup a database. I'm going to add the commands here for reference. The video tutorials will show you in detail how to use the command.

Always ensure you are in your home directory before using the mysqldump command. Create a special directory in your home directory for database backups.

There is NO SPACE between the hyphen p and password.

To backup a **single** database:

```
mysqldump -u root -p db_name > backup_filename.sql
```

To backup **multiple** databases:

```
mysqldump -u root -p --databases db_name1 db_name2 > backup_filename.sql
```

To backup **all** databases:

```
mysqldump -u root -p --all-databases > backup_filename.sql
```

Backup a Database - compressed with gzip

If your databases are large, you can use gzip to compress the sql file. All you do is pipe the output to gzip

To backup and compress a **single** database:

```
mysqldump -u root -p db_name | gzip -9 > backup_filename.sql.gz
```

To backup and compress **multiple** databases:

```
mysqldump -u root -p --databases db_name1 db_name2 | gzip -9 > backup_filename.sql.gz
```

To backup and compress **all** databases:

```
mysqldump -u root -p --all-databases | gzip -9 > backup_filename.sql.gz
```

In the next section I'm going to show you how to create a bash script. I don’t recommend the multiple or the all database dump commands. Rather use a bash script that performs an automatic backup of each database separately.

Restore a Database - mysqldump and gzip file

Before restoring a database, please make sure you know the name of the database that you want to restore.

If you are working with WordPress, check the wp-config.php file for the database information and create that database and then restore the dump file to that database.

```
mysql -u root -p db_name < backup_filename.sql
```

If the file has been compressed using gzip

```
gunzip < backup_filename.sql.gz | mysql -u root -p db_name
```

Uncompress a GZIP File

If you would like to gunzip a file that has been compressed with gzip.

```
gunzip name_of_file.sql.gz
```

When you run that command, the original gzip file will be erased. If you would like to keep the gzip file, use the -k flag

```
gunzip -k name_of_file.sql.gz
```

Database Backups Using a Bash Script

It's an easy task to setup a bash file to backup your databases. You don't need to type any commands, just run the bash script and its done. Later we will setup these scripts to run automatically.

In your home directory, create a directory to hold the backup files

```
cd
mkdir db_backups
```

Using nano, we must create the bash script in our home directory.

```
nano backup_db_sitename1.sh
```

The bash script contents:

```
#!/bin/bash

db_name=
db_user=
db_password=
backup_filename=site_name1-`date +%F_%H%M`
mysqldump -u $db_user -p$db_password $db_name | gzip >
/home/user/db_backups/$backup_filename.sql.gz
```

NB: The last command mysqldump is typed on a single line and the is NO space before or after the equals sign
db_name = database name
db_user = root mysql username
db_password = root mysql password NO space between the -p and the password
backup_filename= site name, date and time, NB note the BACKTICK `

After you exit nano, the script must be made executable

```
chmod +x backup_db_sitename1.sh
```

To run the script, type a period, then a forward slash then the filename

```
./backup_db_sitename1.sh
```

Your backup is done!

To create a script for additional sites, copy the original file

```
cp backup_db_sitename1.sh backup_db_sitename2.sh
```

Then edit the file using nano and change the following: the database name and the site_name in the line containing backup filename.

```
nano backup_db_sitename2.sh
```

In this fashion, create backup scripts for all your databases. We will create scheduled backups later.

Installing WP-CLI on your VPS

First check if curl is installed, by typing curl at the prompt. If it says command not found, install curl, otherwise you can move on.

```
sudo apt-get install curl
```

Change to your home directory. To install wp-cli, firstly we need to download wp-cli using curl, notice the UPPERCASE letter O, then make wp-cli executable and finally move wp-cli into our path so we can type wp-cli commands from within any directory.

```
cd
curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar
chmod +x wp-cli.phar
sudo mv wp-cli.phar /usr/local/bin/wp
```

To test wp-cli, type wp --info at the prompt

```
wp --info

PHP binary:     /usr/bin/php5
PHP version:    5.6.14-0+deb8u1
php.ini used:   /etc/php5/cli/php.ini
WP-CLI root dir:        phar://wp-cli.phar
WP-CLI global config:
WP-CLI project config:
WP-CLI version: 0.22.0
```

Updating WP-CLI

Whenever a major **new version of WordPress is released**, I recommend you first check if wp-cli needs to be updated **before** using it to update your WordPress sites.

The command is identical to when you first installed wp-cli.

Let's create a bash script to do it for us. Change to your home directory first.

```
cd
nano wpcli-update.sh
```

```
#!/bin/bash

cd
curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar
chmod +x wp-cli.phar
sudo mv wp-cli.phar /usr/local/bin/wp
```

Make the script executable and then run the script. The script will stop, at the sudo mv and ask you to type your password.

```
chmod +x wpcli-update.sh
./wpcli-update.sh
```

To test wp-cli, type wp --info at the prompt

```
wp --info

PHP binary:     /usr/bin/php5
PHP version:    5.6.14-0+deb8u1
php.ini used:   /etc/php5/cli/php.ini
WP-CLI root dir:        phar://wp-cli.phar
WP-CLI global config:
WP-CLI project config:
WP-CLI version: 0.22.0
```

All good, let's move on to installing WordPress…

INSTALLING WORDPRESS USING WP-CLI

Ownership of /var/www

Currently our web root, /var/www is owned by root. We need to change the ownership of the /var/www/ directory as we don’t serve pages as the root user.

This is a temporary ownership to allow wp-cli to write to the directories. We will change it again later once we have installed WordPress. The -R means recursive.

```
sudo chown -R $USER:$USER /var/www/
```

We also need to add our user to the www-data group, that is the web server group. You must logout and then login again to enable the group change.

```
sudo usermod -a -G www-data $USER
logout
```

It's always easiest to change to the directory where you want to install WordPress. When we setup apache virtual hosts, we created the various site directories.

The directories are located in the /var/www directory. We also created the domain_name1.com/public_html directories inside the /var/www. The directory that we want to install WordPress in is the public_html directory. Let's change to that directory.

```
cd /var/www/domain_name1.com/public_html
```

Install the First WordPress Site

Before installing WordPress, ensure that you have created the database and changed to the directory where you want to install WordPress.

It’s a three stage process to install WordPress using wp-cli.

WP-CLI will download WordPress from wordpress.org, then the wp-config.php file will be created and finally WordPress will be installed. When installing WordPress, there is NO space after the equal sign. Fill in all the values after the equal sign, nine in total.

wp core download

wp core config --dbname**=** --dbuser**=** --dbpass**=** --dbprefix**=**

wp core install --url**=** --title**=** --admin_user**=** --admin_password**=** --admin_email**=**

Use your text editor to setup the wp-cli install command. The above three lines are then combined with a double ampersand.

The title, if it's more than one word it must be enclosed in quotation marks.

NEVER use **wp_** as the WordPress database **prefix** or **admin** as the WordPress **username**.

Here is what the final command will look like, I'm just using generic values

```
wp core download && wp core config --dbname=db_name1 --dbuser=db_username1
--dbpass=db_password1 --dbprefix=prefix1_ && wp core install --
url=http://www.mydomainname.com --title="Welcome To My Site" --
admin_user=Gu43myemdiLDT --admin_password=83dTc6kRz53ECIN --
admin_email=webmaster@mydomainname.com
```

Once you have provided all the information, select, then copy the information and paste into terminal or putty and press enter.

```
Downloading WordPress …
Success: WordPress downloaded
Success: Generated wp-config.php file.
Success: WordPress installed successfully.
```

WordPress has been successfully installed.

At this stage we can create the .htaccess file. This file is located inside the /var/www/domain_name1/public_html directory.

```
touch .htaccess
```

Now we need to correct the ownership of the directory.

```
sudo chown -R $USER:www-data /var/www/domain_name1.com/
```

You may now visit your site in your browser, this time using the domain name. You can also login into WordPress admin, by appending /wp-admin/ to your site address.

Install the Second WordPress Site

The steps are identical, except, don't change ownership of /var/www and there is no need to add your user to the www-data group. It's done already.

You will need to provide all the details wp-cli needs, use your text editor with strong random names and passwords.

```
wp core download && wp core config --dbname=db_name2 --dbuser=db_username2
--dbpass=db_password2 --dbprefix=prefix2_ && wp core install --
url=http://www.my2nddomainname.com --title="Welcome To My 2nd Site" --
admin_user=ecd45gfSCH6iB --admin_password=pXvPr57CWAP8465 --
admin_email=webmaster@my2nddomainname.com
```

Once you have provided all the information, select, then copy the information and paste into terminal or putty and press enter.

```
Downloading WordPress …
Success: WordPress downloaded
Success: Generated wp-config.php file.
Success: WordPress installed successfully.
```

WordPress has been successfully installed. Now we need to correct the ownership of the directory.

```
sudo chown -R $USER:www-data /var/www/domain_name2.com/
```

You may now visit your site in your browser, this time using the domain name. You can also login into WordPress admin, by appending /wp-admin/ to your site address.

WordPress Security Checklist

We need to ensure a few things are correct at this stage.

The file and directory ownership, file and directory permissions, spend some time on wp-config.php and .htaccess. Then we need to remove plugin and theme version numbers.

Ownership

The ownership should be correct, as we changed the ownership immediately after installing WordPress.

```
sudo chown -R $USER:www-data /var/www/domain_name1.com/
```

Permissions

We need to change the permissions for the directories and files. We are going to use the find command to perform this task.

I recommend you add this command to your text file that contains all you site data, it's easy just to paste the command. BUT, for practice, I recommend you type it out.

The first line will correct the permissions for directories and the second line the files.

```
sudo find /var/www/domain_name1.com/ -type d -exec chmod 775 {} \;
sudo find /var/www/domain_name1.com/ -type f -exec chmod 664 {} \;
```

wp-config.php

We need to move wp-config.php back one level and delete the wp-config-sample.php file. Ensure you are in your sites public_html directory.

```
mv wp-config.php ../
rm wp-config-sample.php
```

Navigate back on level and edit wp-config.php using nano

```
cd ../
nano wp-config.php
```

We need to add the following at the end of the wp-config.php file.

```
/** Allow Direct Updating Without FTP */
define('FS_METHOD', 'direct');
/** Disable Editing of Themes and Plugins Using the Built In Editor */
define('DISALLOW_FILE_EDIT', 'true');
/** Allow Automatic Core Updates */
define('WP_AUTO_UPDATE_CORE', 'true');
```

The first line allows direct updates of themes and plugins without being prompted for FTP details. The second line prevents the built in editor from making changes to any files. The third line allows for the WordPress core to be updated automatically.

The permissions of wp-config.php should be 440, but…

There are many plugins, for example wp-super-cache and ithemes security that need to be able to write to the wp-config.php file. I recommend you set the initial permission at 660 and when you have finished setting up your caching and security plugins, change the permissions to 440.

So, initially 660, finish setting up your security and caching plugins, then change the permissions of wp-config.php to 440.

.htaccess

The permissions of .htaccess should be 440, but…

There are many plugins, for example wp-super-cache and ithemes security that need to be able to write to the wp-config.php file. I recommend you set the initial permission at 660 and when you have finished setting up your caching and security plugins, change the permissions to 440.

So, initially 660, finish setting up your security and caching plugins, then change the permissions of .htaccess to 440.

Remove Plugin and Theme Version Numbers

This section helps to reduce WordPress information leakage. The main security problem with WordPress comes from badly coded plugins and themes. Keeping your plugins and themes up to date is extremely important.

We can setup a scheduled job to check and if necessary update our plugins and themes every x number of hours, but some plugin updates can cause havoc with a WordPress site.

What we do is remove the theme and plugin version numbers, so if anyone is scanning your site they will only know you are using a certain plugin, they won't know what version it is. This step creates frustration and confusion as they would need to try every version and every exploit of that version. In the mean time you have moved on and the plugin has already been updated.

We will again use the find command. The first line will print the list of txt files on the screen. You can use the cat command to view the contents of any of the files.

The second command will delete all the txt files. Don’t forget the quotation marks.

```
sudo find /var/www/domain.com/ -type f -name "*.txt" -print
sudo find /var/www/domain.com/ -type f -name "*.txt" -delete
```

WP-CLI IMPORTANT NOTE

Before initiating any wp-cli command, always ensure you are in the correct directory.

When we setup virtual hosts, the path to our sites was specified as follows:

/var/www/site_name1/public_html/ and /var/www/site_name2/public_html/

Before using any wp-cli command, change to the correct directory

```
cd /var/www/site_name1/public_html/
```

Then type the wp-cli commands. Check which directory you are in by looking at the prompt, the path is displayed to the left of the dollar sign.

```
:/var/www/site_name1/public_html$ wp theme install theme_name
```

If you are not in the directory, you will need to specify the path. A lot of extra typing!!

```
:~$ wp theme install --path=/var/www/site_name1/public_html theme_name
```

USING WP-CLI TO ADMINISTER WORDPRESS

We can use wp-cli to administer all of our WordPress sites on the VPS. We can list, install, activate, deactivate and update both themes and plugins. If you need to import images, wp-cli is the ideal tool for the purpose. The easiest way to create a child theme is to use wp-cli.

The commands for themes and plugins are identical

Using the wp-cli commands in a bash script makes life even easier. You must specify the path to your WordPress files in the bash script.

wp-cli and themes

Always change to the public_html directory of the WordPress site you want to work with, otherwise you must specify the --path=

The command is wp theme followed by the action:

After typing wp theme leave a space and type an action					
activate	delete	install	list	update	search

With the search action, you can specify how many results you'd like to be displayed.

Search for the theme and then install the theme using the 'slug' name. You can install multiple themes, just search for each theme, make note of the slug and then after the install command, separate them with a space.

```
wp theme search --per-page=30 shop

Success: Showing 30 of 72 themes.

+----------------------+----------------------+--------+

| name                 | slug                 | rating |

+----------------------+----------------------+--------+

| TheShop              | theshop              | 100    |

| Shophistic Lite      | shophistic-lite      | 0      |

| WShop                | wshop                | 0      |

| Dark Shop lite       | dark-shop-lite       | 0      |
| e-Shopper           | e-shopper           | 0      |

wp theme install theshop
wp theme install wshop e-shopper

wp theme list

+----------------+----------+--------+---------+

| name           | status   | update | version |

+----------------+----------+--------+---------+

| 2016           | active   | none   | 0.1.0   |

| hueman-master  | inactive | none   | 2.2.5   |

| twentyfifteen  | inactive | none   | 1.4     |

| twentyfourteen | inactive | none   | 1.6     |

| twentysixteen  | parent   | none   | 1.1     |

+----------------+----------+--------+---------+
```

The themes are all download from wordpress.org. if your theme is in a zip file, upload it to your server using filezilla or scp and then specify the path to the zip file. The first theme file is located in my home directory and the second in /var/www/mysite.

```
wp theme install ~/mytheme.zip
wp theme install /var/www/mysite/mytheme.zip
```

The --all option is useful to update all themes.

```
wp theme update --all
Success: Updated 0/0 themes.
```

The wp theme list will inform you if there is an update(s) available, then run the above command.

wp-cli and plugins

Always change to the public_html directory of the WordPress site you want to work with, otherwise you must specify the --path=

The command is wp plugin followed by the action:

After typing wp plugin leave a space and type an action					
activate	delete	Install	list	update	search

With the search action, you can specify how many results you'd like to be displayed.

```
wp plugin list

+---------------------------+----------+--------+---------+

| name                      | status   | update | version |

+---------------------------+----------+--------+---------+

| akismet                   | active   | none   | 3.1.7   |

| contact-form-7            | active   | none   | 4.3.1   |

| crayon-syntax-highlighter | active   | none   | 2.8.0   |

| google-sitemap-generator  | active   | none   | 4.0.8   |

| better-wp-security        | active   | none   | 5.1.1   |

| regenerate-thumbnails     | active   | none   | 2.2.6   |

| responsive-lightbox       | active   | none   | 1.6.6   |
```

To activate all your plugins, use --all

```
wp plugin activate --all
```

To update all your plugins, use --all

```
wp plugin update --all
Success: Updated 0/0 plugins.
```

The wp plugin list will inform you if there is an update(s) available, then run the above command. The plugins are all download from wordpress.org. if your plugin is in a zip file, upload it to your server using filezilla or scp and then specify the path to the zip file. The first plugin file is located in my home directory and the second in /var/www/mysite.

```
wp plugin install ~/plugin_00.zip
wp plugin install /var/www/mysite/plugin_00.zip
```

```
wp plugin install hello-dolly
Installing Hello Dolly (1.6)
Downloading install package from
https://downloads.wordpress.org/plugin/hello-dolly.1.6.zip...
Unpacking the package...
Installing the plugin...
Plugin installed successfully.
Success: Translations updates are not needed for the 'English (US)'
locale.


wp plugin delete hello-dolly
Success: Deleted 'hello-dolly' plugin.
```

Search for the plugin and then install the plugin using the 'slug' name. You can install multiple plugins, just search for each plugin, make note of the slug and then after the install command, separate them with a space.

```
wp plugin search --per-page=100 woo
Success: Showing 100 of 380 plugins.

+----------------------+---------------------+--------+

| name                 | slug                | rating |

+----------------------+---------------------+--------+

| Woo Button Text      | woo-button-text     | 100    |

| Woo Question         | woo-question        | 0      |

| Woo Custom Emails    | woo-custom-emails   | 80     |

wp plugin install woo-button-text
wp plugin install woo-question woo-custom-emails
```

Working with images and wp-cli

It's easy to import images using wp-cli. You can import a few to thousands of images.

The easiest way is to upload your images using FileZilla, then import the images into your WordPress library using wp-cli media import.

What I normally do is create a directory in the /var/www directory and upload all my image files to that directory.

```
cd /var/www
mkdir temp_img
```

At this stage I'll use FileZilla to upload all my image files to the /var/www/temp_img directory.

To import the images into the WordPress library, use the wp media import. Ensure your are in the correct WordPress site directory or use the --path= option.

```
wp media import /var/www/temp_img/*
```

The above command will import all image files from the temp_img directory into your WordPress library. One of the advantages of wp media import is that we can define the alternate text as well as the title.

```
wp media import /var/www/temp_img/* --title"My Images" --alt="Alternate
Text"
```

Add the --title= and --alt= will save you an incredible amount of time when importing images.

You can also import images from a web site using wp media import.

```
wp media import url --title"My Images" --alt="Alternate Text"

wp media import http://www.abc.com/image.jpg --title"My Images" --
alt="Alternate Text"
```

Using wp media import is an excellent way to import images without needing to add another plugin.

Always try and reduce your dependency on plugins.

Creating a child theme using wp-cli

No theme is perfect! There is always something that you may want to change.

You should never edit the original theme files as when that theme gets updated, all your changes will be lost.

Creating a child theme is quick and easy using wp-cli. It's probably the quickest way of doing it.

```
wp theme list

+----------------+----------+--------+---------+

| name           | status   | update | version |

+----------------+----------+--------+---------+

| 2016           | active   | none   | 0.1.0   |

| twentyfifteen  | inactive | none   | 1.4     |

| twentyfourteen | inactive | none   | 1.6     |

| twentysixteen  | parent   | none   | 1.1     |

+----------------+----------+--------+---------+
```

As you can see above, I want to create a child theme, called 2016, from the twentysixteen theme.

The command is wp scaffold child-theme

```
wp scaffold child-theme 2016 --parent_theme=twentysixteen --
theme_name='2016 Child' --author='Author Name' --author_uri='Author url' -
-theme_uri=http://websiteaddress.com

Success: Created /var/www/sitename.com/public_html/wp-content/2016.
```

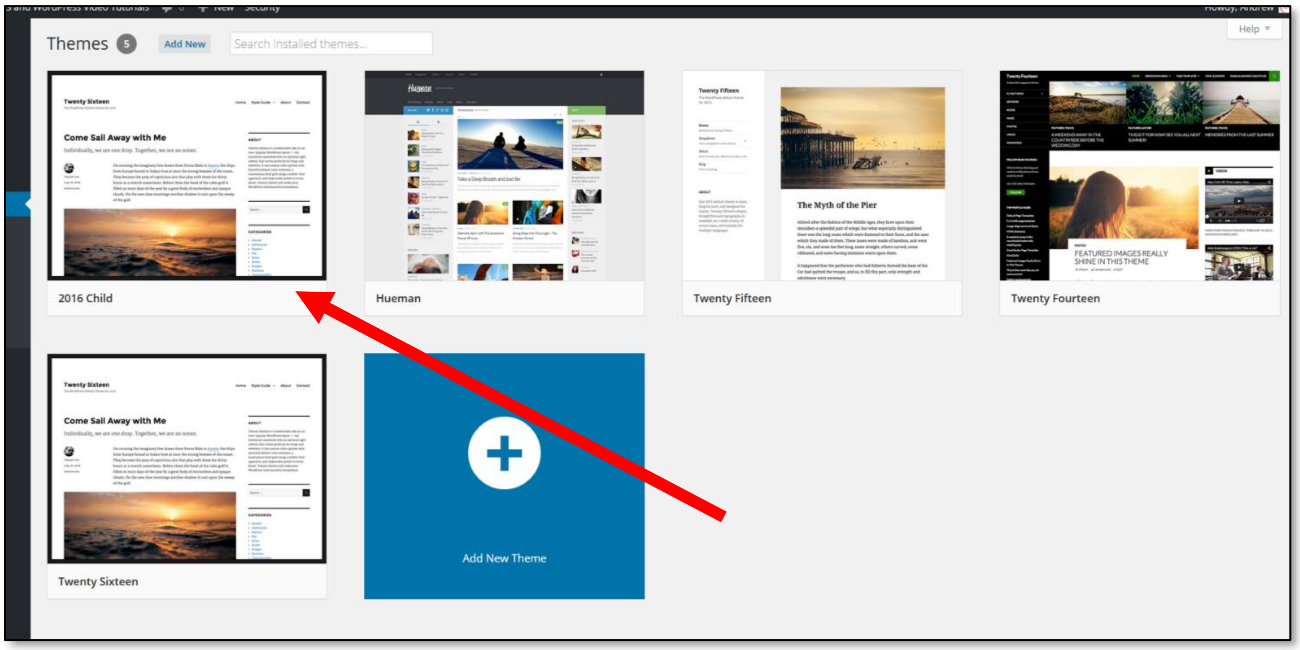
Explanation:

2016	folder to create to store child theme files
--parent-theme	theme you want to make a child from
--theme_name	your child theme name
--author	your name
--author_uri	your site address or any site address if you prefer
--theme_uri	your site address

Your child theme will appear in the WordPress Dashboard. To add an image, copy the original themes screenshot.png file to the 2016 directory.

Change to the public_html directory of the site, then change to the directory of the parent theme.

```
cd /var/www/sitename/public_html/
cd wp-content/themes/twentysixteen/
cp screenshot.png ../2016
```



The end result is a perfect child theme. The 2016 directory will contain the files style.css and functions.php. Any files you want to edit, for example the header.php, copy the original file to the child themes directory and then edit the file. You can edit any php or css file using nano.

WORDPRESS FILE BACKUPS

When it comes to WordPress backups, there are only two files that we need to concern ourselves with. The database dump file and the site backup.

In a previous section[20], I covered backing up the database. In this section we are going to cover backing up the site files.

It's a nightmare trying to download all the WordPress site files individually, there are thousands of files and directories. Even if your broadband connection is fast, it's extremely slow due to the thousands of files. Windows users can also run into another problem in that the paths become too long and Windows will not be able to read or write to those directories. This is not a problem for MACOSX and LINUX users.

It's much easier to compress the WordPress files into a single file and download that.

The easiest way to backup your site files is to create bash script for each WordPress site. Once you have created the bash script, it will take a few seconds to backup your site files. It will take longer if you have thousands of high res images and HD videos. We are going to use tar, as it is extremely efficient at compressing thousands of files.

Before creating a bash script, always ensure you are in your home directory. Create a directory to hold your site backups, then use nano to create the bash script.

```
cd
mkdir site_backups
nano domain_name_site1_backup.sh
```

Please follow my naming convention in the video tutorials.

Now the bash script, please note the back ticks then exit nano.

```
#! /bin/bash

destination_folder=/home/user/site_backups/
archive_file=domain_name1-`date +%F_%H%M`.tar.gz
/bin/tar -czvf $destination_folder/$archive_file /var/www/domain_name1/
```

tar options:	c = create a new file
	z = compress automatically
	v = display files on screen while being processed
	f = archive file

You must make the script executable and you can run the script.

```
chmod +x domain_name_site1_backup.sh
./domain_name_site1_backup.sh
```

Create a bash script like this for all your sites, each time changing the domain_name on line two and domain name on line three.

Like we did with the database backup script, make of copy of the first one and then just make the changes using nano.

```
cp domain_name_site1_backup.sh domain_name_site2_backup.sh
nano domain_name_site2_backup.sh
```

The changes for the second site

```
#! /bin/bash

destination_folder=/home/user/site_backups/
archive_file=domain_name2-`date +%F_%H%M`.tar.gz
/bin/tar -czvf $destination_folder/$archive_file /var/www/domain_name2/
```

After running the bash script, you will find a tar file, in ~/site_backups, that contains your WordPress site files.

WORDPRESS FILE RESTORE

Just off your home directory in site_backups you have a tar file that contains your WordPress site backup.

Firstly, you need to extract the WordPress files from the tar file

```
tar xf name_of_the_backup_file.tar.gz
ls -l
```

You will notice a directory, call var in your home directory. When tar compresses it stores the full path. Change to each of the directories until you have changed to the domain name directory, then type ls -l.

```
cd var
cd www
ls
cd domain_name
ls -l
```

There is the directory public_html and your wp-config.php file.

What you need to do now is copy the files and directories to the desired location.

```
cp -R * /var/www/domain_name/
```

Your site will not work at this stage as the permissions and ownership are wrong.

```
sudo chown -R $USER:www-data /var/www/domain_name.com/
sudo find /var/www/domain_name1.com/ -type d -exec chmod 775 {} \;
sudo find /var/www/domain_name1.com/ -type f -exec chmod 664 {} \;
```

Please check permissions on wp-config.php and .htaccess. Both must be set to 440.

Your site is now working…

AUTOMATED BACKUPS

We are going to use cron jobs to automate and schedule our database backups and site backups.

Be a little careful with the times that you run your backups. If your sites are large and you have large databases, running a few of these commands concurrently may affect your server's performance. Create a separate backup file for each of your sites and databases and add them individually, separating the times where needed. Use FileZilla to download the backup files to your pc or mac. You never know when disaster may strike.

It's pretty straight forward to setup a cron job. There are two commands you need to know:

crontab -l	to list current cron jobs
crontab -e	to create a cron job

CRON - Understanding the Schedule

```
* * * * * /bin/sh command >/dev/null 2>&1
```

*	*	*	*	*	/bin/sh	command	>/dev/null 2>&1
MIN	HOUR	DOM	MON	DOW	/bin/sh	command	>/dev/null 2>&1

Cron Tab Fields and Allowed Entries

FIELD	DESCRIPTION	ALLOWED VALUE
MIN	Minutes	0 - 59
HOUR	Hours	0 - 23
DOM	Day of the Month	1 - 31
MON	Month	1 - 12
DOW	Day of the Week: Sun = 0 to Sat = 6	0 - 6

You need to change the asterisks to specify when you want the scheduled event to run. Let's say we want a command to run at **5:30am** every day. We also want a command to run at **11:25pm every Saturday** and a command at **10:00am every Thursday**. The last command we want to run **every 6 hours**.

*	*	*	*	*	/bin/sh	command	>/dev/null 2>&1
MIN	HOUR	DOM	MON	DOW	/bin/sh	command	>/dev/null 2>&1
30	**5**	*****	*****	*****	/bin/sh	command	>/dev/null 2>&1
25	**23**	*****	*****	**6**	/bin/sh	command	>/dev/null 2>&1
0	**10**	*****	*****	**4**	/bin/sh	command	>/dev/null 2>&1
0	***/6**	*****	*****	*****	/bin/sh	command	>/dev/null 2>&1

Let's look at the other entries we need to know. /bin/bash follows the asterisks. Then you type the full path to the command.

Every time you run a cronjob, mail is sent to the user who setup and ran the job. That's why we installed mailutils earlier. Mailutils will allow us to read system mail. The problem is that it gets annoying to receive system mail every time we login to our server. The >/dev/null 2>&1 redirects the mail to an empty location. We will never receive mail regarding that particular cron job.

CRON - Setting the Schedule

Earlier we setup two bash scripts to backup our databases and backup the WordPress files. The files are backup_db_sitename1.sh and domain_name_site1_backup.sh. Both these files are in our home directory.

You should backup databases every day and the site files once a week.

Let's do this, the command is crontab -e. After typing the command scroll to the bottom of the file.

Scheduled Automated Database and WordPress Backups

We are going to **backup the databases every day, starting at 04:00am** and the **WordPress files every Sunday, starting at 07:00am**.

One additional Allowed Value is the /x, where a command will be repeated every x number of hours.

```
crontab -e
```

```
# m h  dom mon dow   command
0 4 * * * /bin/sh /home/user/backup_db_sitename1.sh >/dev/null 2>&1
0 7 * * 0 /bin/sh /home/user/domain_name_site1_backup.sh >/dev/null 2>&1
```

MOVING WORDPRESS

There are three situations that we need to consider when moving WordPress from server to server.

1. The domain name remains the same - VPS to VPS
2. The domain name changes - VPS to VPS
3. Moving from a shared host to a VPS

Before you even consider moving WordPress to a different server

1. You must ensure the new server has been setup and configured correctly, following all the steps on setting up a VPS.
2. Thereafter setup Apache Virtual Hosts, creating the relevant site directories.
3. Then you must install wp-cli as per the course instructions.

Moving WordPress - Same Domain Name

Moving WordPress from server to server is easy. You need three files:

1. The wp-config.php file
2. The database dump - the sql file
3. The WordPress site backup file

From the wp-config file, take note of the database name, the database username and the database password.

On the new server, create the same database as contained in the wp-config.php file.

Copy the database dump file and the WordPress site backup file to your new server. You can copy the files to your home directory on your new server.

Restore the database dump file, as per the instructions on Page **21**

```
mysql -u root -p db_name < backup_filename.sql
```

Extract the WordPress files as per the instructions on Page **41** and proceed with changing the ownership and permissions. Follow all the instructions on Page **41**.

Finally, you can login to your preferred Domain Name Registrar and change the DNS settings to point to your new server.

The A Record [Host] - the @ sign - must point to your VPS IP address and the CName [Alias] www must point to the @ sign.

If you are unsure, ask your Domain Name Registrar for help or send me a screenshot.

Watch the videos on moving WordPress and you will see how it's done using godaddy.com

Moving WordPress - Different Domain Name

The procedure is identical as if you are moving with the same domain name, except we need to change certain strings in the database to reflect the change in the domain name.

Do NOT use a text editor and perform a search and replace. You will mess things up as some entries are serialized. When you watch the corresponding videos, you will see that a text editor does not find all the entries.

We will use wp search-replace to perform the search and replace that's needed.

As with most steps, **change to the public_html directory of the WordPress site or use the --path= option**. Add the --dry-run option before performing the actual search and replace. The search and replace strings are typed between two apostrophe's

```
wp search-replace 'old' 'new' --dry-run
wp search-replace 'old' 'new'
```

```
wp search-replace 'old_domain_name' 'new_domain_name' --dry-run
```

To run the actual search and replace, remove --dry-run

```
wp search-replace 'old_domain_name' 'new_domain_name'
```

Let's say the **old** domain is www.abc.com and the **new** domain is www.xyz.com

Both domains are .com, makes life easy

```
wp search-replace 'abc' 'xyz' --dry-run
wp search-replace 'abc' 'xyz'
```

Let's say the **old** domain is www.abc.net and the **new** domain is www.def.com

```
wp search-replace 'abc.net' 'def.com' --dry-run
wp search-replace 'abc.net' 'def.com'
```

Refresh your site and it's all done, 100% correctly.

Moving WordPress - Shared Hosting to a VPS

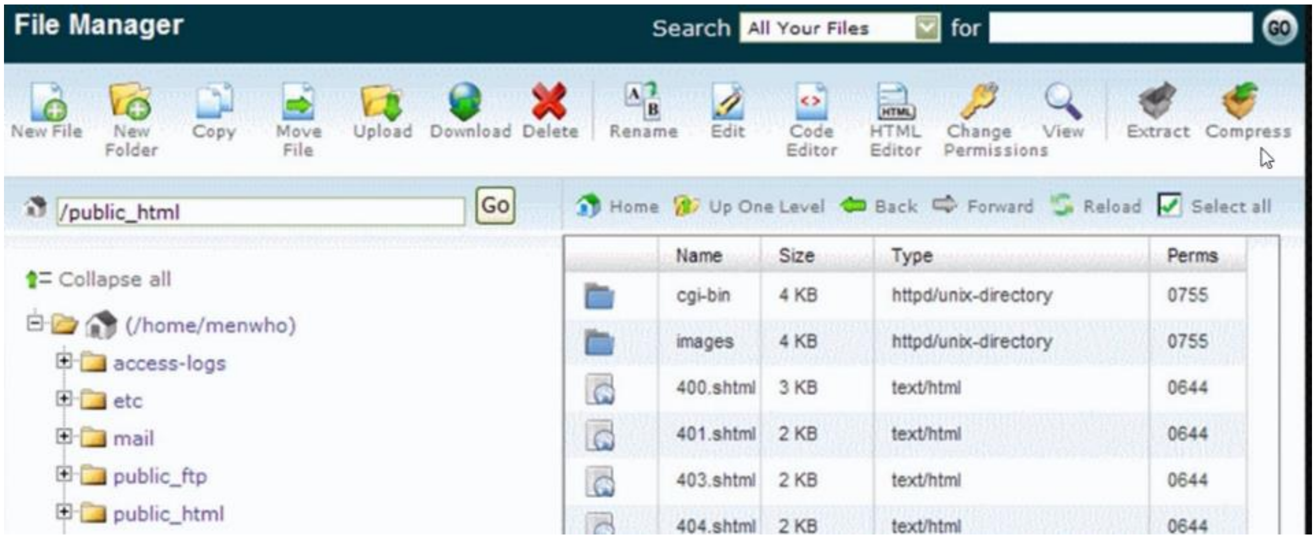
Finally, you have decided to skip shared hosting, let's move your WordPress sites asap.

You need three files:

1. The wp-config.php file
2. The database dump - the sql file
3. The WordPress site backup file

Use your **cPanel File Manager** and download the wp-config.php file. Once you have downloaded wp-config.php, open it and take note of the database name, the database username and the database password.

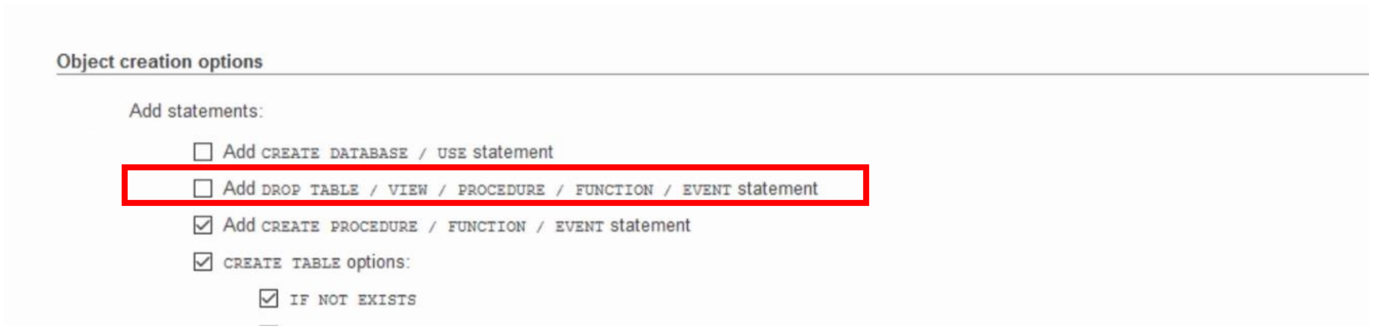
Using the **cPanel File Manager** browse to the directory that contains the WordPress files you want. Select all the files and then select Compress on the toolbar. It will offer you various types of compression, select tar, then download the file. compressing.



Name	Size	Type	Perms
cgi-bin	4 KB	httpd/unix-directory	0755
images	4 KB	httpd/unix-directory	0755
400.shtml	3 KB	text/html	0644
401.shtml	2 KB	text/html	0644
403.shtml	2 KB	text/html	0644
404.shtml	2 KB	text/html	0644

Using **cPanel again, open phpmyadmin**. On the left hand side, click the database name you made note of from the wp-config.php file. Then click export and the export method must be custom.

Select all tables, then **scroll down till the heading Object creation options. YOU MUST SELECT THE Add DROP TABLE / VIEW / PROCEDURE / FUNCTION / EVENT statement**. Click go and download the database dump file.



Object creation options

Add statements:

- [ ] Add CREATE DATABASE / USE statement
- [ ] Add DROP TABLE / VIEW / PROCEDURE / FUNCTION / EVENT statement
- [x] Add CREATE PROCEDURE / FUNCTION / EVENT statement
- [x] CREATE TABLE options:
  - [x] IF NOT EXISTS

Finally, you are ready to move. Just ensure you have the wp-config.php file, the WordPress tar file and the database sql file.

You can now follow the videos, Moving WordPress - Same Domain Name or Moving WordPress - Different Domain Name.

CHOOSING A VPS HOST

If your site is a static site, i.e. just html, css, image and video files then a shared hosting environment may be suitable but you can also host hundreds of static sites on a single $3.49 VPS. If you are using WordPress, Joomla or Drupal, a VPS is the ideal hosting platform.

You need to ask yourself a few questions and do a little research before you decide on a VPS.

How much space do you need?

A minimum of 10 Gigabytes is needed. Your server OS installation will take around a 1 gig of space and after updates and various other software installations, you are looking at around 1.5 gigs. Let’s be conservative and make it 2 gigs to get your VPS up and running, serving content to your visitors. What type of content will you be hosting? Is your site mainly text with a few images? Is your site image based with a little text? Are you going to serve videos, images and text? The last aspect to think about is the SSD vs HD. SSD wins every time, but expect to pay an additional premium for the speed of an SSD.

How much RAM do you need?

You can never have enough RAM. The more RAM you have the better your server will run. It’s that simple. Do you want an unmanaged or managed VPS? Unmanaged means you are fully responsible for the VPS. Expect NO help from the host. The host is responsible for the hardware and network connection. Managed VPS hosting is very close to shared hosting, but, its expensive. An unmanaged VPS will run perfectly on 512MB – 1GB of RAM. A managed VPS needs more RAM due to the Control Panel. A Control Panel will need at least 512MB – 1GB resources on its own.

How many CPU cores do you need on a VPS?

Start out with a single core and as your site gets busy you can upgrade the VPS. Its recommend that at a minimum for a fairly busy site you have with 1GB of RAM per core, but preferably 2GB per CPU core. It depends on how busy your site is.

What connection speed would be adequate for your sites?

The ideal speed is 1Gb/s. Most VPS hosts will offer you an unmetered 100Mb/s connection. 100Mb/s is fine if you are starting a new site, but as your site gets busy you will need the 1Gb/s burst speed. After you have completed this course, you will have the knowledge to move your site(s) to a different server quickly, easily, error free and most importantly PLUGIN FREE.

How easy is it to upgrade a VPS?

Most VPS web hosts provide a one click upgrade option, an instant upgrade. If the VPS host you decide on does not offer this option, go somewhere else. You can use htop to monitor your server resources.

What server operating system must you choose?

For your first VPS, select UBUNTU 14.04 LTS. The LTS designation (Long Term Support Release) means you don't have to worry about a major server upgrade until April 2019. It's the easiest server distribution to start with. Please ensure that the host gives you root access to the VPS.

How many sites can I host on a single VPS?

If you VPS is setup correctly and optimized, you can comfortably host 8 WordPress sites per gig of RAM. If you are setting up static sites, that number is almost unlimited.

What VPS hosts do I recommend?

I can only speak from experience, there are only 2 hosts I have used and I'm happy with both of them. I start my new sites out on ovh.com and as the traffic grows I move them over to DigitalOcean.

OVH: ovh.com are one of the largest hosting companies around. Their prices are rock bottom for a VPS and they are reliable. I run a large number, last count was 103, of WordPress sites from a number of ovh.com VPS's.

DigitalOcean: digitalocean.com is probably the best VPS host today. As soon as a site becomes popular and starts to make demands on my server, I move it to DigitalOcean. Why do I do that? DigitalOcean offers 1Gb/s connection and an SSD drive.

FREE VPS Hosting for TWO MONTHS

If you sign up with DigitalOcean using this link, you will receive a $10 credit free.

You DO NOT need to provide any type of payment information, just sign up and enjoy the benefits of hosting your sites on a VPS.

If you choose a 512 MB RAM VPS, that's 2 months free hosting.

You can use this link to sign up with DIGITALOCEAN

COMMANDS

adduser	add a user to the vps
apt-get	install or remove software on the vps, update, upgrade and dist-upgrade are also used.
cat	display and combine files
cd	change to the current users home directory
cd ..	go back one directory level, can be used to go back multiple levels: cd ../../
cd /	change to the "root directory" of the vps
chmod	change file and directory permissions
chown	change file and directory ownership
cp	copy
logout	logout of the vps
ls	list files and directories, use with -l for a detailed list or -a to display all, e.g. ls -l or ls -a or ls -la
mkdir	create a directory
mv	move or rename a file or directory
nano	a basic editor
passwd	change the current users password
pwd	print working directory, display current directory path
rm	remove or delete a file or directory
rmdir	delete 'remove' a directory
service	to start, stop and restart services that are running on our vps
sudo	invoke root privileges, sudo is typed before the command
touch	create a blank new file
usermod	modify groups user belongs to, must be used with the -a -G flags. lowercase a UPPERCASE G
visudo	edit the sudoers file

This is a very basic list of commands, commands hardly ever used are not included in this list. All commands are typed in lowercase.

VPS CHECKLIST

ROOT USER SECURE

- ☐ change root password
- ☐ add new non root user
- ☐ give non root user root privileges
- ☐ prevent root login

LOGIN AS NON ROOT USER

- ☐ update vps, including dist-upgrade
- ☐ create .ssh directory in the /home/user directory

CREATE SSH KEY PAIR

- ☐ create ssh key pair on local pc or mac
- ☐ copy public key to server
- ☐ secure public key on server
- ☐ prevent password login
- ☐ setup pageant or create a config file to make VPS access easier

SETUP FIREWALL

- ☐ add rules to block all ports except http, https, ssh and ping
- ☐ check if rules are enabled after reboot

APACHE

- ☐ install
- ☐ configure
- ☐ harden
- ☐ optimize
- ☐ test web server

MYSQL

- ☐ install
- ☐ configure
- ☐ harden
- ☐ optimize

PHP

- ☐ install
- ☐ configure
- ☐ harden
- ☐ optimize
- ☐ test php

FAIL2BAN

- ☐ install
- ☐ configure

APACHE VIRTUAL HOSTS

- ☐ setup virtual hosts for each site that is going to be hosted on VPS
- ☐ remove default apache site

ADMINISTRATION

- ☐ install mailutils and sendmail
- ☐ keep VPS updated
- ☐ log files
- ☐ resource monitor - htop

COURSE SUPPORT

If you have any questions or queries, please don't hesitate to contact me on the Udemy course support page.