


Website Setup

Prerequisites

Build a Cloud Linux VPS

 [Ubuntu 22.04 VPS Base Config](#)

Register a Domain

While there are many options, this guide will leverage [Namecheap](#)

Find a WordPress Theme

While there are many options, this guide will leverage [Envato Elements](#)

Add DNS Records



DNS A Records are a fundamental part of how the internet works, serving as the directory for the World Wide Web. A Records, also known as Address Records, link a domain to the physical IP address of a server, which is the machine where the website or service associated with that domain is hosted.

When a user types a URL into their browser, their computer queries the DNS servers to find the corresponding IP address for that domain. This process is known as DNS resolution. The A Record is what provides the necessary IP address to complete this resolution, thus allowing the browser to reach the correct server and load the requested website or service. By adding and managing A Records for your domain, you can control which server hosts your website or service and ensure that users are directed to the correct location when they enter your URL.

1. Access DNS Management Console:

- Log into your hosting provider's control panel.
- Navigate to the DNS management section. This section might be labeled as "DNS Settings," "DNS Zone," or something similar.

2. Select Record Type:

- Look for an option to add a new record. This is usually marked by a button like "Add Record" or "Create New Record."
- Select "A" from the list of record types. An A Record points a domain or subdomain to an IPv4 address.

3. Add the @ Record:

- **Name/Host:** Use "@" to signify the root domain (e.g., yourdomain.com).
- **Value/Points to:** Enter the IPv4 address you want the root domain to point to.
- **TTL (Time To Live):** If given the option, choose the shortest period offered (often 300 seconds). This setting determines how long DNS information is cached.

4. Add the www Record:

- **Name/Host:** Enter "www" to point the subdomain (e.g., www.yourdomain.com) to the same or different IPv4 address.
- **Value/Points to:** Enter the IPv4 address you want the www subdomain to point to.
- **TTL:** Again, choose the shortest period available.

By following these steps, you can successfully add DNS A Records for your domain, ensuring that your domain and subdomains point to the correct IP addresses. The process is similar across various hosting providers, making these steps broadly applicable.

Install LEMP Stack

A LEMP stack is a group of open source software typically installed together to enable a server to host dynamic websites and web apps. This acronym represents the Linux operating system, with the Nginx web server (pronounced Engine-x, hence the 'E'). The backend data is stored in the MariaDB database, and dynamic processing is handled by PHP.

In the context of our WordPress installation, the LEMP stack provides a robust and capable environment for running WordPress efficiently. Linux provides a secure and stable operating system. Nginx, being a high-performance HTTP server, ensures that our WordPress site can handle large amounts of traffic without compromising speed. MariaDB handles the storage and retrieval of our website data including posts, pages, user profiles, settings etc. Lastly, PHP processes all our WordPress PHP files and converts them into static HTML pages that can be served to the end-user. Hence, all these components work together to ensure that our WordPress website runs smoothly and efficiently.

```
sudo apt install -y nginx mariadb-server php-fpm php-mysql
```

Secure MariaDB Installation

MariaDB is an open-source database management system that defaults to a secure mode. This approach ensures that our databases are as secure as possible to prevent unauthorized access and data breaches, given the sensitive nature of the data that might be stored in these databases.

Unlike MySQL, MariaDB doesn't require running a separate script for securing the installation as it comes secure by default. This design leaves less room for human error and potential security risks. It's crucial to ensure that unless explicitly changed, MariaDB installations would always be secure.

Set Up MariaDB for WordPress

1. Drop into a MySQL shell

```
sudo mariadb
```

2. Create a database for WordPress to use:

⚠️ Replace `wordpress` with whatever database name you want or leave it as is.

```
CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8 COLLATE
```

3. Create a user for WordPress to use.

⚠️ Replace `wordpressuser` with the username you want or leave it as is.
Replace the `password` with an actual strong password that you'll need to jot down for later use.

```
CREATE USER 'wordpressuser'@'localhost' IDENTIFIED BY 'password'
```

4. Grant the new user access to the `wordpress` database created above.

```
GRANT ALL ON wordpress.* TO 'wordpressuser'@'localhost';
```

5. You can now exit from the database by typing `exit;`

Allow Nginx Through UFW

```
sudo ufw allow 'Nginx Full'
```

Obtain TLS Certificates

Install Certbot

```
sudo apt install -y snapd && sudo snap install core && snap refi
```

Stop Nginx Service

It's essential to secure standalone certificates for your site before setting up Nginx. This upfront security measure guarantees that as soon as your site goes live, it will already be HTTPS-enabled, enhancing its security. To facilitate this, we'll employ Certbot. However, before initiating this process, it's crucial to stop the Nginx service, as Certbot requires access to port 80 to procure the certificate, and currently, this port is occupied by the Nginx service.

```
sudo systemctl stop nginx
```

Obtain Certs

```
sudo certbot certonly --standalone -d yourdomain.com -d www.yourdomain.com
```

Configure Diffie-Hellman Parameters

Diffie-Hellman parameters (dhparam) play a crucial role in establishing secure HTTPS connections. They are used in the Diffie-Hellman key exchange, an algorithm that allows two parties, in this case, a client and a server, to establish a shared secret key over an insecure network.

In the context of our Nginx config, using dhparam improves the security of the website by enhancing the SSL/TLS encryption. This enhancement is achieved by generating a stronger DHE (Diffie-Hellman Ephemeral) parameter for the Diffie-Hellman key exchange process.

Key benefits of using dhparam in Nginx config include:

- **Enhanced Security:** On generating a unique dhparam file, you are creating a strong encryption system for key exchange that is hard to crack, thus securing your data in transit.

- **Perfect Forward Secrecy (PFS):** In case a server's private key is compromised, previous communications encrypted with that key will still be safe due to the nature of the Diffie-Hellman key exchange. PFS ensures that even if a key is compromised in the future, past communications remain secure.



Remember, it's essential to generate a unique dhparam file for each server to ensure maximum security.

```
sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 4096
```

Configure Nginx

In Nginx, the `sites-available` and `sites-enabled` directories play a vital role in managing different server blocks for various services. These directories provide an organized way to manage multiple configuration files for different servers, making the process of adding, removing, or updating server configurations much simpler and more manageable.

The `sites-available` directory is a repository for all your server block configuration files, whether they're active or not. Each server, for instance, `yourdomain.com` & `www.yourdomain.com`, should have a distinct configuration file in this directory. Additional applications, such as `app.yourdomain.com`, should also have their own separate configuration files. This structure lets you adjust and fine-tune server configurations without impacting the servers that are currently live.

On the other hand, the `sites-enabled` directory contains symbolic links to the configurations in the `sites-available` directory. Only the sites that are currently enabled (i.e., live) should have their configuration files linked in the `sites-enabled` directory.

This separation between available and enabled sites provides a clear and effective system for managing server configurations. It allows you to prepare and test configurations before going live and to disable sites easily without deleting their configuration.

Having separate configuration files for each service can greatly aid in managing and troubleshooting your servers. If a problem arises, you can isolate it to a specific server by looking at its individual configuration file. Furthermore, if you need to make changes to a specific server, you can do so without worrying about inadvertently affecting your other services.

So, if you're running multiple services, it's a good practice to have separate configuration files for each one. It keeps your configurations clean, organized, and easy to handle, making your administrative tasks much more manageable.

Rate-Limit Requests

Add the following within the `/etc/nginx/nginx.conf` file inside the `http{...}` block:

```
limit_req_zone $binary_remote_addr zone=mylimit:10m rate=10r/s;
```

Build Site Config File

Use `nano` to create a new config file. It's best practice to name it after the domain or service it's responsible for (e.g., `/etc/nginx/sites-available/yourdomain.com`). File contents:



Update `yourdomain.com` with your registered domain name.

▼ Toggle Nginx Site Config File

```
server {
    listen 80;
    server_name supergenlabs.com www.supergenlabs.com;

    # Redirect all HTTP requests to HTTPS
    return 301 https://$server_name$request_uri;
}

server {
```

```
listen 443 ssl http2;
server_name supergenlabs.com www.supergenlabs.com;

root /var/www/supergenlabs.com;
index index.php index.html index.htm;

# SSL Certificate Configuration
ssl_certificate /etc/letsencrypt/live/supergenlabs.com/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/supergenlabs.com/privkey.pem;

# SSL Protocols and Ciphers
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers HIGH:!aNULL:!MD5;
ssl_prefer_server_ciphers on;

# SSL Session Settings
ssl_session_cache shared:SSL:10m;
ssl_session_timeout 10m;

# DH Parameters for Forward Secrecy
ssl_dhparam /etc/ssl/certs/dhparam.pem;

# OCSP Stapling for Improved SSL Performance
ssl_stapling on;
ssl_stapling_verify on;
ssl_trusted_certificate /etc/letsencrypt/live/supergenlabs.com/ocsp-response.pem;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;

# Client Body Size Limit
client_max_body_size 1G;

# Security Headers
add_header X-Content-Type-Options nosniff;
add_header X-Frame-Options SAMEORIGIN;
add_header X-XSS-Protection "1; mode=block";
```



```
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";
add_header Referrer-Policy "no-referrer-when-downgrade";

# Content Security Policy (CSP)
add_header Content-Security-Policy "default-src 'self' http://example.com https://example.com";

# Additional Security Headers
add_header X-Permitted-Cross-Domain-Policies none;
add_header Expect-CT "max-age=7776000, enforce";

# Rate Limiting Configuration
location / {
    limit_req zone=mylimit burst=20;
    try_files $uri $uri/ /index.php?$args;
}

# Favicon Handling
location = /favicon.ico {
    log_not_found off;
    access_log off;
}

# Robots.txt Handling
location = /robots.txt {
    log_not_found off;
    access_log off;
    allow all;
}

# Static File Handling with Caching
location ~* \.(css|gif|ico|jpeg|jpg|js|png|woff|woff2|ttf|eot)$ {
    expires max;
    log_not_found off;
    access_log off;
}
```

```

# PHP File Handling
location ~ /\.php$ {
    include fastcgi_params;
    fastcgi_pass unix:/run/php/php8.1-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param SCRIPT_NAME $fastcgi_script_name;
    fastcgi_index index.php;
    fastcgi_buffers 16 16k;
    fastcgi_buffer_size 32k;
}

# Custom Error Pages
error_page 404 /404.html;
location = /404.html {
    internal;
}

error_page 500 502 503 504 /50x.html;
location = /50x.html {
    internal;
}

# Access and Error Logs
access_log /var/log/nginx/supergenlabs.com_access.log;
error_log /var/log/nginx/supergenlabs.com_error.log;

# Gzip Compression
gzip on;
gzip_types text/plain text/css text/javascript application/javascript;
gzip_proxied any;
gzip_vary on;

# Caching for Static Files
location ~* \.(css|js|jpg|jpeg|png|gif|ico|svg|woff|woff2|ttf|eot)$ {
    expires 30d;
    add_header Cache-Control "public, no-transform";
}

```

```
}  
}
```

Enable Site

Use the following command to create the symbolic link to `sites-enabled`

```
sudo ln -s /etc/nginx/sites-available/yourdomain.com /etc/nginx.
```

Test Nginx Configuration

```
sudo nginx -t
```

Restart Nginx

Assuming that the test succeeded, we need to restart the Nginx service.

```
sudo systemctl restart nginx
```

Install & Configure WordPress

Protect WordPress Installation Page

When you first install WordPress, it becomes immediately accessible to the internet. By default, a setup wizard presents an initial installation and admin configuration page. This page is vital as it allows you to set up your website's basic details and create your admin user account. However, this page can pose a significant security risk if it's exposed to the internet.

Malicious actors can potentially find this page and try to compromise your site before you've even had a chance to set it up. Therefore, it's crucial to restrict access to this setup page when you first install WordPress.

To manage this, we leverage the Uncomplicated Firewall (UFW). UFW is a user-friendly front-end for managing iptables firewall rules. It provides a simple way to create and manage firewall rules on Ubuntu servers.

We can configure UFW to restrict access to our WordPress installation page to only our Public IPv4 Address. This way, only you can access the setup page and create your admin user in a secure environment.

After completing the installation and admin setup, we will then adjust the UFW settings once again to allow access from any location. This step opens up your website to the world, allowing users to visit your site while ensuring that the initial setup was done securely.

Remove 'Nginx Full' Access in UFW

```
sudo ufw delete allow 'Nginx Full'
```

Add Your Public IPv4 to UFW

```
IP=$(who am i | awk '{print $5}' | tr -d '()') && sudo ufw allow
```

Install PHP Extensions for WordPress

```
sudo apt install -y php-curl php-gd php-intl php-mbstring php-so
```

Create Website Root Directory

We have configured Nginx to serve our website from `/var/www/yourdomain.com/` so this directory must exist, and this is where we will install WordPress.

```
sudo mkdir /var/www/yourdomain.com
```

Download WordPress

```
cd /var/www/yourdomain.com && curl -LO https://wordpress.org/lat
```

Extract WordPress & Clean Up

```
tar xzf latest.tar.gz && rm latest.tar.gz
```

Configure WordPress

1. WordPress relies on a configuration file called `wp-config.php` but one does not yet exist. Move into the newly extracted `wordpress/` directory and make a copy of the sample config file, but name it `wp-config.php`

```
cd wordpress/ && sudo cp wp-config-sample.php wp-config.php
```

2. Modify the `wp-config.php` file to get WordPress set up.

```
sudo nano -l wp-config.php
```

- Update the `DB_NAME` with the database name you created earlier.
- Update the `DB_USER` with the username you created earlier.
- Update the `DB_PASSWORD` with the password you created for that user.

▼ Toggle Screenshot

```
GNU nano 6.2 wp-config.php *
10 *
11 * * Database settings
12 * * Secret keys
13 * * Database table prefix
14 * * ABSPATH
15 *
16 * @link https://wordpress.org/documentation/article/editing-wp-config-php/
17 *
18 * @package WordPress
19 */
20
21 // ** Database settings - You can get this info from your web host ** //
22 /** The name of the database for WordPress */
23 define( 'DB_NAME', 'wordpress' );
24
25 /** Database username */
26 define( 'DB_USER', 'wordpressuser' );
27
28 /** Database password */
29 define( 'DB_PASSWORD', ' ' );
30
31 /** Database hostname */
32 define( 'DB_HOST', 'localhost' );
33
34 /** Database charset to use in creating database tables. */
35 define( 'DB_CHARSET', 'utf8' );
36
37 /** The database collate type. Don't change this if in doubt. */
38 define( 'DB_COLLATE', '' );
39
```

- Replace each of the Authentication Keys and Salts with unique values. You can then save this file and exit nano.

▼ Toggle Screenshot

```
GNU nano 6.2 wp-config.php *
39
40 /**#@+
41 * Authentication unique keys and salts.
42 *
43 * Change these to different unique phrases! You can generate these using
44 * the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}.
45 *
46 * You can change these at any point in time to invalidate all existing cookies.
47 * This will force all users to have to log in again.
48 *
49 * @since 2.6.0
50 */
51 define( 'AUTH_KEY', 'put your unique phrase here' );
52 define( 'SECURE_AUTH_KEY', 'put your unique phrase here' );
53 define( 'LOGGED_IN_KEY', 'put your unique phrase here' );
54 define( 'NONCE_KEY', 'put your unique phrase here' );
55 define( 'AUTH_SALT', 'put your unique phrase here' );
56 define( 'SECURE_AUTH_SALT', 'put your unique phrase here' );
57 define( 'LOGGED_IN_SALT', 'put your unique phrase here' );
58 define( 'NONCE_SALT', 'put your unique phrase here' );
59
60 /**#@-*/
```

- You can use this to get strong unique values.



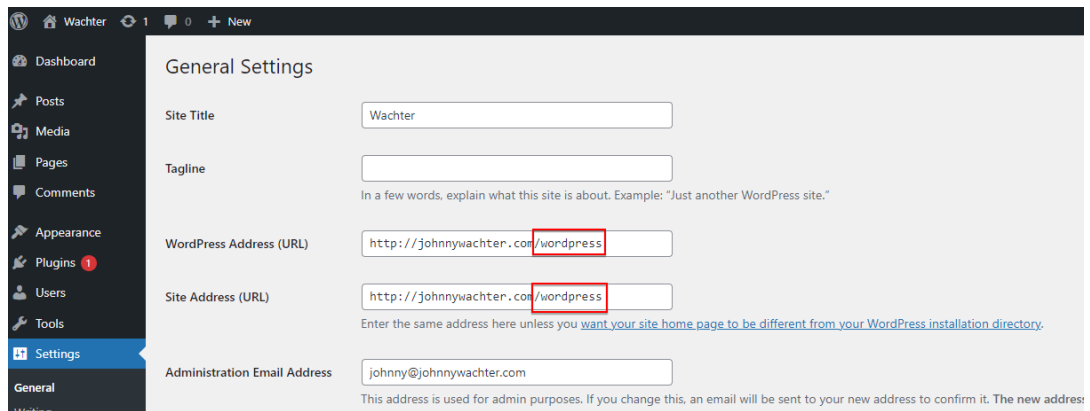
Remember that in `nano` the key combo `CTRL + K` cuts text, so you can use this to easily remove each of the lines shown in the above screen shot. You can then use `CTRL + SHIFT + V` to paste those values copied from the WordPress Secret Key & Salt API linked above.

3. Change ownership recursively of the `/var/www/` directory to `www-data`


```
chown -R www-data:www-data /var/www
```

4. You can now access your WordPress site via `http://yourdomain.com/wordpress` where you can continue the installation/configuration.
5. Once logged in, let's change the way that our site is accessed. Currently, because everything lives in the root directory `/var/www/yourdomain.com/wordpress/` that means that users will be accessing your website at the same domain you did in the previous step; this looks bad. We can adjust it now.
 - a. Begin by navigating to `Settings` then removing `/wordpress` from both the fields shown in the following screenshot:

▼ Toggle Screenshot



- b. Scroll down and `Save Changes`

 You will be redirected to a page that looks like an error. This is expected and will be resolved by completing the next steps.

- c. With this change applied, we now need to move files on our server to reflect this.

```
sudo mv /var/www/yourdomain.com/wordpress/* /var/www/yourd
```

Increase Site Limits

If your server specifications support it, you can increase size limits on your server by modifying the `/etc/php/8.1/fpm/php.ini` file. Find and increase the value for each of these variables:

```
upload_max_filesize = 1024M
post_max_size = 1024M
memory_limit = 1024M
```

Restart PHP FPM

```
sudo systemctl restart php8.1-fpm
```

Reopen Site to the Internet

Now that we've completed the WordPress installation, we can reopen the site to the internet.

```
sudo ufw delete allow from YOUR_PUBLIC_IPv4_ADDRESS to any port
```

Access the WordPress Admin Dashboard

You should now be able to access the WordPress Admin Dashboard at any time by visiting <https://yourdomain.com/wp-admin>

Remove Default WordPress Plugin "Hello Dolly"

1. Navigate to the 'Plugins' section on the left-hand navigation menu.
2. Here you will see a list of installed plugins. Locate the "Hello Dolly" plugin.
3. You will see the option to 'Deactivate' the plugin if it's activated. Click on 'Deactivate'.
4. Once the plugin is deactivated, the 'Delete' option will appear. Click 'Delete' to completely remove the plugin from your WordPress installation.

Please note that deleting a plugin will also delete all its data. Make sure you won't need any of the data before deleting a plugin.

Activating the Akismet WordPress Plugin

1. Navigate to 'Plugins' on the left-hand navigation menu.
2. Look for the 'Akismet' plugin in the list of installed plugins.
3. Click on 'Activate' to activate the plugin. Upon activation, a notice will appear at the top of your screen saying that Akismet is almost ready and that you need to enter your Akismet API key.
4. Click on the link in the notice to enter your API key. If you don't have an API key, you'll be guided to get one.

Getting a Free Akismet API Key

1. Navigate to the [Akismet Plans](#) page.
2. Choose the 'Personal' plan for a free account. Click on 'Get Personal'.
3. On the next page, adjust the slider to '0' to get it for free. Remember, this is for personal blogs only, not for commercial sites.
4. Fill in your account details and click on 'Continue'.
5. You now have an Akismet account and an API key. This key will be emailed to you, but it will also be displayed on the next page.

Entering the Akismet API Key in WordPress

1. Go back to your WordPress site.
2. Navigate to 'Settings' → 'Akismet Anti-Spam'.
3. Paste your API key into the 'Key' field and click 'Connect with API key'.
4. Your Akismet plugin is now activated and will start protecting your site from spam.

Installing a WordPress Theme via the Web UI

1. Navigate to Appearance → Themes on the left-hand navigation menu.
2. Click on the 'Add New' button at the top of the page.
3. You can now search for the theme you want to install. Use the search bar at the top right of the page.
4. Once you've found the theme you want, click the 'Install' button.
5. After the theme is installed, the 'Install' button will change to 'Activate'. Click 'Activate' to start using your new theme.

Removing Default WordPress Themes

1. Still within the Appearance → Themes section, you will see all your installed themes.
2. Hover over the theme you want to remove. Some details will appear, along with a 'Theme Details' button. Click this button.
3. A window will open displaying more details about the theme. In the bottom right corner of this window, there is a 'Delete' link. Click this link to remove the theme.
4. A confirmation window will appear. Click 'OK' to confirm the deletion.

Please note that deleting a theme will also delete all its data, including any customizations or settings you have made. Be sure you want to delete a theme before following these steps.