# LEMP : Install Multiple PHP Versions with Nginx on Ubuntu

# [Install Nginx](https://blog.containerize.com/2021/05/21/how-to-install-multiple-php-versions-with-nginx-on-ubuntu/" \l "nginx)

* [Install Multiple PHP Versions](https://blog.containerize.com/2021/05/21/how-to-install-multiple-php-versions-with-nginx-on-ubuntu/" \l "php)
* [Create Multiple Websites with PHP](https://blog.containerize.com/2021/05/21/how-to-install-multiple-php-versions-with-nginx-on-ubuntu/" \l "web)
* [Configure Nginx to Run Different Versions for Websites](https://blog.containerize.com/2021/05/21/how-to-install-multiple-php-versions-with-nginx-on-ubuntu/" \l "configure)

# To install Nginx, use following command:

[ sudo apt update ]

[ sudo apt install nginx ]

After installation is done you can open “https:// ip” in your browser and you have all setup.

# Install Multiple PHP Versions

[ sudo apt-get install software-properties-common -y ]

[ sudo add-apt-repository ppa:ondrej/php ]

[ sudo apt-get update -y ]

Once the repository is up to date, install PHP 7.4, PHP 8.1 and PHP-FPM with the following command:

[ apt-get install php7.4 php7.4-fpm php8.1 php8.1-fpm -y ]

Once the installation has been completed, check the status of PHP-FPM with the following command:

[ systemctl status php7.4-fpm ]

[ systemctl status php8.1-fpm ]

# Create Multiple Websites with PHP :

mkdir /var/www/html/project1.com

mkdir /var/www/html/project2.com

Next create a sample index.php file for project1:

[ vi /var/www/html/project1.com/index.php ]

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Hello, Nginx! project1</title>

</head>

<body>

<h1>Hello, Nginx! project11</h1>

<p>We have just configured our Nginx web server on Ubuntu Server!</p>

<?php

phpinfo();

?>

</body>

</html>

Now create a sample index.php file for project2:

[ vi /var/www/html/project2.com/index.php ]

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Hello, Nginx! project2</title>

</head>

<body>

<h1>Hello, Nginx! project2</h1>

<p>We have just configured our Nginx web server on Ubuntu Server!</p>

<?php

phpinfo();

?>

</body>

</html>

# Then, change the ownership of both websites to www-data:

[ chown -R www-data:www-data /var/www/html/project1.com ]

[ chown -R www-data:www-data /var/www/html/project2.com ]

# Configure Nginx :

Next, you will need to create an Nginx virtual host file for domain project1.com that uses PHP 7.4 And another for project2.com that uses PHP 8.2

[ vi /etc/nginx/sites-available/project1com.conf ]

server {

listen 80;

root /var/www/html/project1.com/;

index index.php;

server\_name project1.com;

location / {

try\_files $uri $uri/ =404;

}

location ~ \.php$ {

try\_files $uri =404;

fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;

fastcgi\_pass unix:/var/run/php/php7.4-fpm.sock;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

}

Now create the second file for project2.com :

[ vi /etc/nginx/sites-available/site2.containerize.com.conf ]

server {

listen 80;

root /var/www/html/project1com/;

index index.php;

server\_name project1.com;

location / {

try\_files $uri $uri/ =404;

}

location ~ \.php$ {

try\_files $uri =404;

fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;

fastcgi\_pass unix:/var/run/php/php8.1-fpm.sock;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

}

Then, enable both virtual host files with the following command:

[ ln -s /etc/nginx/sites-available/project1.com.conf /etc/nginx/sites-enabled/ ]

[ ln -s /etc/nginx/sites-available/project2.com.conf /etc/nginx/sites-enabled/ ]

Finally, restart Nginx and PHP-FPM service to apply all the configuration changes:

[ systemctl restart nginx ]

[ systemctl restart php7.4-fpm ]

[ systemctl restart php8.1-fpm ]

### Create a Self-Signed SSL Certificate for Nginx in Ubuntu

Step 1 — Creating the SSL Certificate

You can create a self-signed key and certificate pair with OpenSSL in a single command:

[ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt ]

“The most important line is the one that requests the Common Name (e.g. server FQDN or YOUR name). You need to enter the domain name associated with your server or, more likely, your server’s public IP address.”

While using OpenSSL, you should also create a strong Diffie-Hellman (DH) group, which is used in negotiating[**Perfect Forward Secrecy**](https://en.wikipedia.org/wiki/Forward_secrecy)with clients.

You can do this by typing:

[ sudo openssl dhparam -out /etc/nginx/dhparam.pem 4096 ]

“This will take a while, but when it’s done you will have a strong DH group at

[/etc/nginx/dhparam.pem] that will be used during configuration.”

Step 2 — Configuring Nginx to Use SSL

Creating a Configuration Snippet Pointing to the SSL Key and Certificate

To properly distinguish the purpose of this file, name it self-signed.conf

[ sudo nano /etc/nginx/snippets/self-signed.conf ]

/etc/nginx/snippets/self-signed.conf

ssl\_certificate /etc/ssl/certs/nginx-selfsigned.crt;

ssl\_certificate\_key /etc/ssl/private/nginx-selfsigned.key;

The parameters you set can be reused in future Nginx configurations, so you can give the file a generic name:

[ sudo nano /etc/nginx/snippets/ssl-params.conf ]

Add the following into your ssl-params.conf snippet file:

/etc/nginx/snippets/ssl-params.conf

ssl\_protocols TLSv1.3;

ssl\_prefer\_server\_ciphers on;

ssl\_dhparam /etc/nginx/dhparam.pem;

ssl\_ciphers EECDH+AESGCM:EDH+AESGCM;

ssl\_ecdh\_curve secp384r1;

ssl\_session\_timeout 10m;

ssl\_session\_cache shared:SSL:10m;

ssl\_session\_tickets off;

ssl\_stapling on;

ssl\_stapling\_verify on;

resolver 8.8.8.8 8.8.4.4 valid=300s;

resolver\_timeout 5s;

# Disable strict transport security for now. You can uncomment the following

# line if you understand the implications.

#add\_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload";

add\_header X-Frame-Options DENY;

add\_header X-Content-Type-Options nosniff;

add\_header X-XSS-Protection "1; mode=block";

Adjusting the Nginx Configuration to Use SSL

Before moving forward, back up your current configuration file:

[ sudo cp /etc/nginx/sites-available/project1.com.conf /etc/nginx/sites-available/project1.com.conf.bak ]

Now, open the configuration file to make adjustments:

[ sudo nano /etc/nginx/sites-available/project1.com ]

Inside, your server block probably begins similarly to the following:

/etc/nginx/sites-available/your\_domain

server {

listen 80;

listen [::]:80;

root /var/www/your\_domain/html;

index index.html index.htm index.nginx-debian.html;

server\_name your\_domain www.your\_domain;

location / {

try\_files $uri $uri/ =404;

}

}

Note: Use a 302 redirect until you have verified that everything is working properly. After, you will change this to a permanent 301 redirect.

In your existing configuration file, update the two listen statements to use port 443 and ssl, then include the two snippet files you created in previous steps:

/etc/nginx/sites-available/your\_domain

server {

listen 443 ssl;

listen [::]:443 ssl;

include snippets/self-signed.conf;

include snippets/ssl-params.conf;

root /var/www/your\_domain/html;

index index.html index.htm index.nginx-debian.html;

server\_name your\_domain.com www.your\_domain.com;

location / {

try\_files $uri $uri/ =404;

}

}

Next, add a second server block into the configuration file after the closing bracket (}) of the first block:

/etc/nginx/sites-available/your\_domain.com

server {

listen 80;

listen [::]:80;

server\_name your\_domain.com www.your\_domain.com;

return 302 https://$server\_name$request\_uri;

}

# Step 3 — Adjusting the Firewall

[ sudo ufw app list ]

Output

Available applications:

Nginx Full

Nginx HTTP

Nginx HTTPS

OpenSSH

You can check the current setting by typing sudo ufw status:

[ sudo ufw status ]

It will probably generate the following response, meaning that only HTTP traffic is allowed to the web server:

Output

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

Nginx HTTP ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Nginx HTTP (v6) ALLOW Anywhere (v6)

To allow HTTPS traffic, you can update permissions for the “Nginx Full” profile and then delete the redundant “Nginx HTTP” profile allowance:

[ sudo ufw allow 'Nginx Full' ]

[ sudo ufw delete allow 'Nginx HTTP' ]

After running sudo ufw status, you should receive the following output:

[ sudo ufw status ]

Output

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

Nginx Full ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Nginx Full (v6) ALLOW Anywhere (v6)

This output confirms the adjustments to your firewall were successful and you are ready to enable the changes in Nginx.

# Step 4 — Enabling the Changes in Nginx

First, check that there are no syntax errors in the files. You can do this by typing sudo nginx -t:

[ sudo nginx -t ]

Output

nginx: [warn] "ssl\_stapling" ignored, issuer certificate not found for certificate "/etc/ssl/certs/nginx-selfsigned.crt"

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok

nginx: configuration file /etc/nginx/nginx.conf test is successful

If your output matches our example, your configuration file has no syntax errors. If this is the case, then you can safely restart Nginx to implement changes:

[ sudo systemctl restart nginx ]

# Step 5 — Testing Encryption

Now, you’re ready to test your SSL server.

Open your web browser and type https://followed by your server’s domain name or IP into the address bar:

https://project1.com

Depending on your browser, you will likely receive a warning since the certificate you created isn’t signed by one of your browser’s trusted certificate authorities:

This warning is expected and normal. We are only interested in the encryption aspect of our certificate, not the third-party validation of our host’s authenticity. Click “ADVANCED” and then the link provided to

At this point, you should be taken to your site. In our example, the browser address bar shows a lock with an “x” over it, which means that the certificate cannot be validated. It is still encrypting your connection. Note that this icon may differ, depending on your browser.

If you configured Nginx with two server blocks, automatically redirecting HTTP content to HTTPS, you can also check whether the redirect functions correctly:

[http://project1.com](http://project1.com/)

[http://project2.com](http://project2.com/)

If this results in the same icon, this means that your redirect worked correctly.

# Step 6 — Changing to a Permanent Redirect

If your redirect worked correctly and you are sure you want to allow only encrypted traffic, you should modify the Nginx configuration to make the redirect permanent.

Open your server block configuration file again:

sudo nano /etc/nginx/sites-available/project1.com.conf

When you’re ready, restart Nginx to make the redirect permanent:

[ sudo systemctl restart nginx ]

After the restart, the changes will be implemented and your redirect is now permanent.