# Lab Webservers

## Server OS / Operating Systems II

Bachelor Toegepaste Informatica
Graduaat Netwerken

Author : Geert Coulommier

Design, Technologie en Gezondheidszorg

# Table of contents

Table of contents ........ 2
1. Prerequisites ........ 3
2. How to use this lab ........ 3
3. Apache ........ 3
  3.1 Tutorial ........ 3
    3.1.1 Installation ........ 4
    3.1.2 Working with the httpd service ........ 4
    3.1.3 Configuring the firewall for HTTP and HTTPS ........ 6
    3.1.4 Virtual Hosts: creating the file structure and index.html ........ 7
    3.1.5 Virtual Hosts: creating the Virtual Host configuration ........ 8
  3.2 Exercise 1: Virtual Host in etc/httpd/conf.d ........ 11
  3.3 Exercise 2: Virtual Hosts with port redirection ........ 11
4. LAMP: Apache with MariaDB and PHP ........ 12
  4.1 Tutorial local installation ........ 12
  4.2 Exercise Docker Apache installation ........ 12
  4.3 Exercise Docker implementation ........ 12
5. NginX ........ 13
  5.1 NginX installation ........ 13
  5.2 PHP installation with PHP-FPM ........ 14
  5.3 Add Server Block ........ 15
  5.4 Exercise: Working with Server Blocks in Nginx ........ 15
  5.5 Exercise: OPTIONAL: Apache as webserver with NginX as reverse proxy ........ 16

# 1. Prerequisites

## What you need for this lab

- Internet connection
- A Linux operating system
  - sudo rights on your machine
  - Ports 80 and 443 accessible from outside the server and not in use by another service.
    - To check this, you can use the command below and see if ports 80 and/or 443 are listed. If so, another service is already listening on these ports.

      ```
      sudo lsof -i -P -n | grep LISTEN
      ```

      (You might need to install lsof if it is not present. Use the package manager for your distribution.)
  - CentOS: the remainder of this lab will use CentOS, as it is a very popular Linux distribution for this kind of deployment, and it leans closely to the default installation of Apache server.
  - Ubuntu: also very popular. Keep in mind that ubuntu deviates from the default Apache installation. For an overview on how to install and configure Apache on Ubuntu, you can use https://help.ubuntu.com/lts/serverguide/httpd.html or read the /etc/httpd/conf/httpd.conf carefully.

# 2. How to use this lab

This lab will use a combination of fully written out tutorials followed by additional exercises which will test your skills and knowledge acquired in the tutorials.

The tutorials will give a step by step tutorial to get you acquanted with all the concepts and commands. To get the most out of these tutorials, make sure to read the text carefully so that you understand what is said, and follow the commands and instructions to the letter so that you get the same final result. Although it is possible to just copy-paste most of the commands to make them execute in the cli-environment, it is advised to type the commands by hand to better memorize them.

The exercises will test your newly learned skills by presenting you with a desired end result, which you can achieve by using all of the material from the tutorial and previous chapters. This is more a personal playground, so feel free to experiment with different commands and approaches to attain the requested result. Also try to repeat the commands and exercises a few times to get a good feel for them.

# 3. Apache

## 3.1 Tutorial

Based on https://devconnected.com/how-to-install-apache-on-centos-8/

### 3.1.1 Installation

1. As usual, we start the installation of new software with an update of the existing software:

```
sudo yum update
```

2. With all software up-to-date, we can start the installation of the Apache package, which is aptly called "httpd".

```
sudo yum install httpd
```

### 3.1.2 Working with the httpd service

3. Next up is to start the httpd service. We use the systemctl command for this. For more information of the workings of systemctl, check out https://wiki.archlinux.org/index.php/Systemd

   Note: ArchLinux is another distribution which focusses on control for the user on how to configure the installation. In other words, most operations have to be done manually. This can give it a steeper learning curve, which is why they provide a very extensive and well-maintained documentation, also for concepts and items not specific for ArchLinux, such as systemd. This is the reason why this documentation is well regarded in the Linux community, also by users of other distributions.

```
sudo systemctl start httpd
```

4. To check whether the service started up correctly, we can again use the systemctl command:

```
sudo systemctl status httpd
```

output:

```
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled;
vendor preset: disabled)
   Active: active (running) since Sun 2020-02-23 15:11:04 CET; 2min 13s
ago
     Docs: man:httpd.service(8)
 Main PID: 4136 (httpd)
   Status: "Running, listening on: port 80"
    Tasks: 213 (limit: 11002)
   Memory: 24.7M
   CGroup: /system.slice/httpd.service
           ├─4136 /usr/sbin/httpd -DFOREGROUND
           ├─4137 /usr/sbin/httpd -DFOREGROUND
           ├─4138 /usr/sbin/httpd -DFOREGROUND
           ├─4139 /usr/sbin/httpd -DFOREGROUND
           └─4140 /usr/sbin/httpd -DFOREGROUND

Feb 23 15:11:04 CENTOS01.localdomain systemd[1]: Starting The Apache
HTTP Server...
```

```
Feb 23 15:11:04 CENTOS01.localdomain systemd[1]: Started The Apache HTTP
Server.
Feb 23 15:11:04 CENTOS01.localdomain httpd[4136]: Server configured,
listening on: port 80
```

Note the `active (running)`, stating it has started up successfully.

5. We now have a running httpd-service, but unless we enable it, it will not start up with the computer, which is what is usually required for a webservice. As we can see in the second line in the above output (Loaded), the service is not yet enabled:

   output:

   ```
   Loaded:  loaded  (/usr/lib/systemd/system/httpd.service;  disabled;
   vendor preset: disabled)
   ```

   Again, we use systemctl:

   ```
   sudo systemctl enable httpd
   ```

   We get this output, stating a symlink has been created which will be used for starting up the service:

   output:

   ```
   [Student@CENTOS01 ~]$ sudo systemctl enable httpd
   Created symlink /etc/systemd/system/multi-
   user.target.wants/httpd.service →
   /usr/lib/systemd/system/httpd.service.
   ```

   Again, we check the status, and should be getting an output similar to the one below, with this the time, the service enabled.

   output:

   ```
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled;
   vendor preset: disabled)
   ```

   To test the result, restart the server, and again check the status of the service.

   ```
   sudo reboot now

   [login into your server]

   sudo systemctl status httpd
   ```

6. To check the version of the httpd service:

```
httpd -v
```

output:

```
[Student@CENTOS01 ~]$ httpd -v
Server version: Apache/2.4.37 (centos)
Server built:   Dec 23 2019 20:45:34
```

7. Other useful commands to control the httpd-service, which should be self-explanatory, are:

```
sudo systemctl restart httpd
sudo systemctl stop httpd
sudo systemctl start httpd
sudo systemctl reload httpd
```

Restart will restart the whole service, while reload will only reload the configuration files. The latter is interesting to use when configuration files have changed for 1 site, but you don't want to restart another site on the same server while reconfiguring the first site.

Alternatively, you can use the apachectl command:

```
apachectl restart
apachectl stop
apachectl start
```

Test these commands, everytime checking the status with:

```
sudo systemctl status httpd
```

or

```
apachectl status
```

### 3.1.3 Configuring the firewall for HTTP and HTTPS

8. At this point, the webserver is running, but the firewall is still blocking access to it from outside the machine. You can test this by taking a webbrowser client application on another computer and trying the ip-address of your machine (see prerequisites). It should time-out.
To open the ports 80 (HTTP) and 443 (HTTPS) on the firewall, we use these commands:

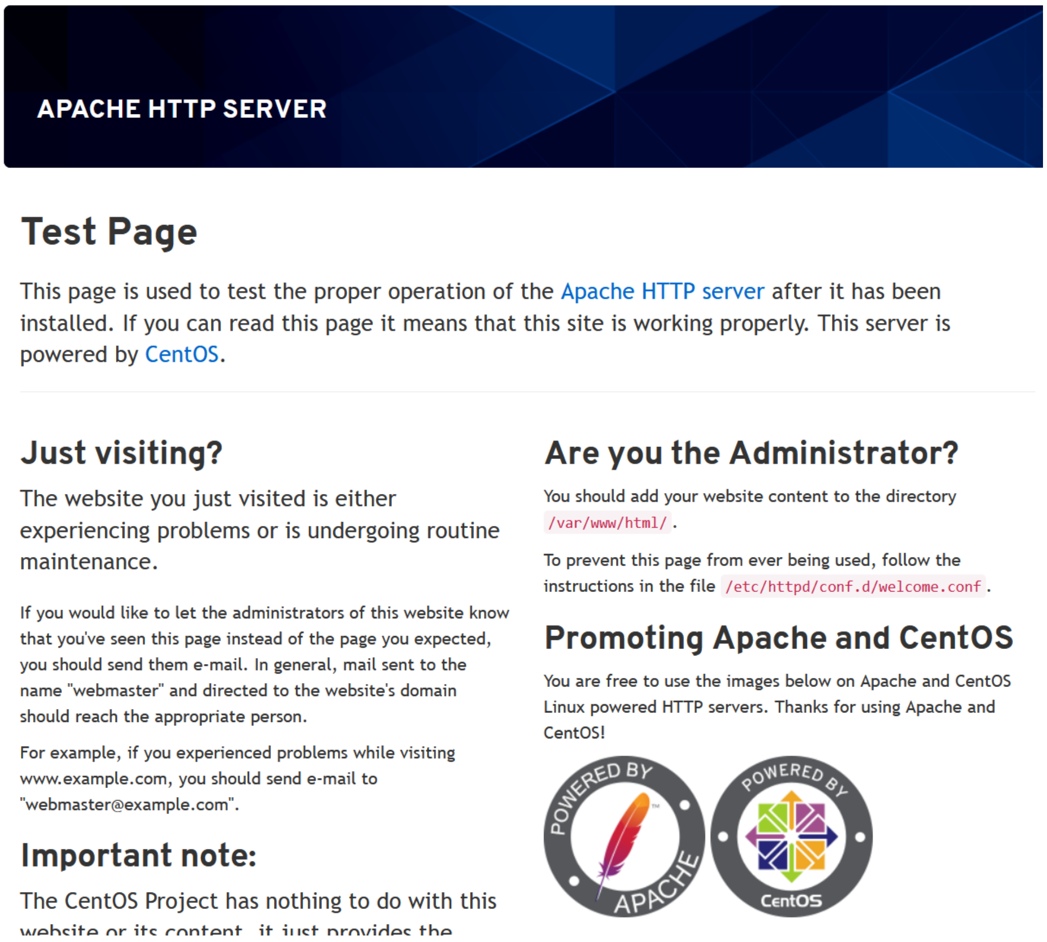
```
sudo firewall-cmd --permanent --zone=public --add-service=http
sudo firewall-cmd --permanent --zone=public --add-service=https
```

After this, we need to reload the firewall to activate these settings:

```
sudo firewall-cmd --reload
```

Try to access your webserver again through an external webbrowser. You should get the default page of Apache:

output:



APACHE HTTP SERVER

Test Page

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

Just visiting?

The website you just visited is either experiencing problems or is undergoing routine maintenance.

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

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

Important note:

The CentOS Project has nothing to do with this

Are you the Administrator?

You should add your website content to the directory `/var/www/html/`.

To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

Promoting Apache and CentOS

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

## 3.1.4 Virtual Hosts: creating the file structure and index.html

9. Working with Virtual Hosts gives us the ability to host multiple websites on one single webserver.
   By default, the files of your websites are stored in `/var/www/html`. To create an new virtual host, we will need a new directory to host the files for this site. We will call this new site website1.local. This is the new folderstructure we need:

```
/var/www/
├── html
├── website1.local
    ├── html
    ├── log
```

The content of our site will end up in the html subfolder, while the logfiles will end up in the logs subfolder.
You can easily create these folders with the following commands:

```
sudo mkdir -p /var/www/website1.local/html
sudo mkdir -p /var/www/website1.local/log
```

Check with the command ls to make sure you have the required folders and subfolders.

10. Next, we will need to create an `index.html` file for our website in the `/var/www/website1.local/html` folder. You can use a linux text editor like nano or vi to achieve this. Copy-paste the below content in this file and save as `index.html`.

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">

  <title>Website1.local</title>
  <meta name="description" content="Website1.local
Homepage">
  <meta name="author" content="Student">
</head>

<body>
  You have reached the indexpage of website1.local.
</body>
</html>
```

### 3.1.5 Virtual Hosts: creating the Virtual Host configuration

11. There are several methods of defining Virtual Hosts on a Apache server. One is to create an extra config file for each Virtual Host in `/etc/httpd/conf.d`. This is the default configuration in Apache as you can see by the line below in `/etc/httpd/conf/httpd.conf`

```
IncludeOptional conf.d/*.conf
```

This is a good time to check out the default configuration of Apache in `/etc/httpd/conf/httpd.conf`. Open it and try to find all the other major configuration directives in the file, such as `ServerRoot`, `Listen`, `Include`, `User`, `Group`, `ServerAdmin`, `DocumentRoot`.

12. Another popular method is to work with sites-available and sites-enabled directories, which we will use here.
    **sites-available** : contains the entire list of websites available on our web server.
    **sites-enabled** : contains the list of websites that are accessible to users. A symbolic link will be created in this directory in order to activate and desactivate websites on demand.

First we will need to create the directories:

```
sudo mkdir -p /etc/httpd/sites-enabled /etc/httpd/sites-available
```

13. Next, we need to change `/etc/httpd/conf/httpd.conf`. Find the line (probably at the bottom of the config file)

    ```
    IncludeOptional conf.d/*.conf
    ```

    And replace it with

    ```
    IncludeOptional sites-enabled/*.conf
    ```

    This will check the `/etc/httpd/sites-enabled` directory for config files, instead of `/etc/httpd/conf.d`.

14. When making changes to config files, it is always a good idea to check for errors before activating them. This can be done with the command:

    ```
    apachectl configtest
    ```

    You should get an output of `Syntax OK`. If not, check your configfile again for errors.

    Try this out! Deliberately make an error in your config file, save it and test with

    ```
    apachectl configtest
    ```

    You should get an output stating where the error is in the config file. Don't forget to repair the error and test again.

15. Now we can make a configuration file for our new website in /etc/httpd/sites-enabled. We will give it the name of our website for easy reference.

    ```
    sudo nano /etc/httpd/sites-available/website1.local.conf
    ```

    And paste the below content into it, defining the name-based Virtual Host.

    ```
    <VirtualHost *:80>
        ServerName website1.local
        ServerAlias www.website1.local
        DocumentRoot /var/www/website1.local/html
        ErrorLog /var/www/website1.local/log/error.log
    </VirtualHost>
    ```

    Again, test the syntax of your configuration file alteration with

    ```
    apachectl configtest
    ```

16. `sites-available` and `sites-enabled` is an easy way of working with multiple sites, because we can create config files for all sites in `sites-available`, and to activate them only need a symbolic link to these files in the sites-enabled. As only the `sites-enabled` folder will be checked for config files, only sites for which there is a symlink in this folder will enabled. To create a symlink for our newly created config file for website1.local, use the following command:

```
sudo ln -s /etc/httpd/sites-available/website1.local.conf
/etc/httpd/sites-enabled/website1.local.conf
```

Check the `sites-enabled` folder for the symlink.

17. CentOS is secured by SELinux, which is by default configured to work with the standard Apache configuration files. As we changed them, we need to enable custom directories in SELinux:

    ```
    sudo setsebool -P httpd_unified 1
    ```

18. Restart the webserver:

    .

    ```
    apachectl restart
    ```

19. Finally we can test the server. To this we need to have a system to resolve the dnsname website1.local to its ip-address. This can be done with a dns server or by altering the hosts file. For now, we will do the latter.
    To change a hosts file you will need administrator rights in Windows or sudo rights in Linux.
    For Windows, you can use this procedure:
    https://www.petri.com/easily-edit-hosts-file-windows-10

    If you are testing on the webserver, go add following line to `/etc/hosts`

    ```
     d
    ```

    If you are testing on another linux machine, replace `127.0.0.1` with the external ip-address of the webserver.

    If you are testing from another machine, we wil use a webbrowser. If you are testing from the webserver, we will use `curl`. This linux command allows us to download static webcontent over http from a website from the command line. The syntax of `curl` is:

    ```
    curl http://[name_or_ip_address_of_the_site]
    ```

    Try these names and ip-addresses in your browser or curl:

    **[the ip address of your webserver]:**
    you should get the default page of Apache
    **website1.local:**
    will display your newly created webpage
    **www.website1.local:**
    will display your newly created webpage (from the alias we configured in our Virtual Host config file)
    **website2.local:**
    should display an error or wrong website (if redirected to some online site with the same name)

output with curl for www.website1.local:

```
[Student@CENTOS01 etc]$ curl http://www.website1.local
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">

  <title>Website1.local</title>
  <meta name="description" content="Website1.local Homepage">
  <meta name="author" content="Student">
</head>

<body>
  You have reached the indexpage of website1.local.
</body>
</html>
```

## 3.2 Exercise 1: Virtual Host in etc/httpd/conf.d

- For this exercise, we will be using the `/etc/httpd/conf.d` folder.
- Create a new virtual host for website2.local and www.website2.local in the `/etc/httpd/conf.d` folder.
- To achieve this, you will also need to edit the `/etc/httpd/conf/httpd.conf` file.
- Make sure the website1.local and www.website1.local also work in this new configuration.
- Test

## 3.3 Exercise 2: Virtual Hosts with port redirection

- For this exercise, we will be using the `/etc/httpd/conf.d` folder
- Create a new virtual host for website3.local and www.website3.local in the `/etc/httpd/conf.d` folder, but this time, serve the website on port 443 (we will not be using https over this port, just regular http).
- For more information on how to configure the Virtual Host, take a look at https://httpd.apache.org/docs/2.4/vhosts/examples.html
- Test

# 4. LAMP: Apache with MariaDB and PHP

In this chapter, we will add MariaDB an PHP to our existing Apache installation on CENTOS, effectively creating a full LAMP stack.

## 4.1 Tutorial local installation

For this tutorial, we will use online resources where the installation of MariaDB and PHP are discussed.

Follow this tutorial from step 2 onwards:

https://www.ostechnix.com/install-apache-mariadb-php-lamp-stack-in-centos-8/

Notes:

- As we already have a running installation of Apache, you can skip step 1 of the tutorial.
- The use of php-fpm is advised from Apache 2.4, that is why we opt for the installation of this module. For more info: https://cwiki.apache.org/confluence/display/HTTPD/php
- It might be necessary to allow PHP in SeLinux. You can use this command to achieve this:

```
setsebool -P httpd_execmem 1
```

- The test at the end of the php configuration assumes a default installation of Apache, which we no longer have due to the previous exercises. Adjust accordingly.

## 4.2 Exercise Docker Apache installation

In the link below you can find a working Docker implementation of an Apache container:

https://www.tecmint.com/install-apache-web-server-in-a-docker-container/

Use this for inspiration to start a Docker Apache container with these specifications:

- Accessible through port 80 on the host
- A web page that is presented through index.html that displays the text "Apache in Docker". You might need to check previous Apache tutorials.
- Can you create a website1.local and website2.local in this container, similar to the configuration in exercise 1 in chapter 3.2? You might also need to check the information on working with volumes in Docker. Remember, we don't like to make changes inside the container, we will always prefer to create links to directories on the host through volumes, so we can persist the data.

## 4.3 Exercise Docker implementation

**NOTE: this exercise will NOT work in the current configuration as hosted on github. You can skip this exercise for now.**

For this tutorial, we will be creating a LAMP installation in Docker. Follow below tutorial:

https://gist.github.com/Beyarz/674b24d03614fde205a38f449800857a

Notes:

- You will need docker-compose to use this configuration. Look for online information on how to install this tool in Linux.
- Like every preconfigured package that you try to implement in your local environment, this will need to modified. Read the files and make changes where necessary to make it work in your environment. The previous exercise in 4.2 might give you some inspiration.
- If you encounter issues with the default page, try replacing it with a simpler php-file, like one with the below content:

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

- Can you troubleshoot the errors with the database?

# 5. NginX

## 5.1 NginX installation

The installation of NginX is very straightforward and quit similar to the installation of many other services, among which Apache. Try to install NginX on your CentOS vm using your knowledge acquired in previous exercises and your troubleshooting skills. The package is called nginx. If you don't remember all the steps, you can use the steps as described in the exercise for installing apache as an example. If you run into problems, you can follow the procedure below.

1. Before installation, it is a good idea to first check that there are no other services running that listening on ports 80 and/or 443. Again, this can be done with the command:

   ```
   sudo lsof -i -P -n | grep LISTEN
   ```

   If you found services that are using this port, make sure to stop them. It might also a good idea to disable them for the duration of the exercise. Also, watch out for Docker containers that might be listening on the same ports. Clean up if necessary.
1. As usual, we start the installation of new software with an update of the existing software:

   ```
   sudo yum update
   ```

2. With all software up-to-date, we can start the installation of the NginX package, which is aptly called "nginx".

   ```
   sudo yum install nginx
   ```

3. If not yet done in previous exercises, you will want to open up ports 80 and 443 on your firewall:

```
sudo   firewall-cmd   --permanent   --zone=public   --add-service=http
sudo firewall-cmd --permanent --zone=public --add-service=https
```

After this, we need to reload the firewall to activate these settings:

```
sudo firewall-cmd --reload
```

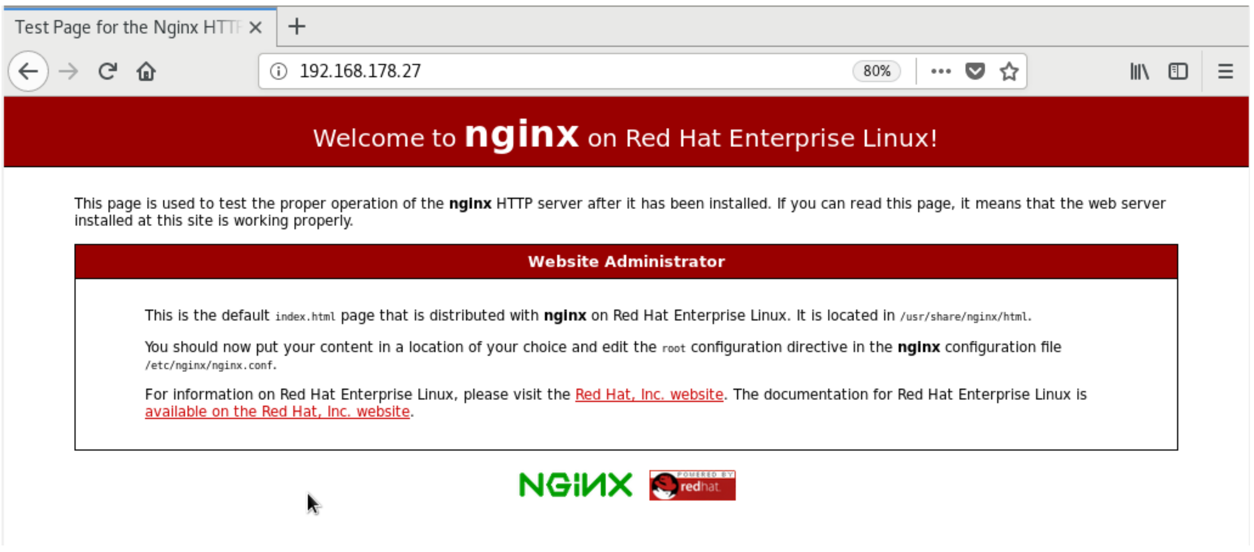
4. Next up is to start and enable the NginX service.

```
sudo systemctl start nginx
sudo systemctl enable nginx
```

5. Finally, check whether everything is running correctly with

```
sudo systemctl status nginx
```

At this stage you should also be able to display the default launch page of NginX. Open a webbrowser and go to `http://[IP_of_the_host]`. You should get a page like this:



Welcome to **nginx** on Red Hat Enterprise Linux!

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

**Website Administrator**

This is the default `index.html` page that is distributed with **nginx** on Red Hat Enterprise Linux. It is located in `/usr/share/nginx/html`.

You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

For information on Red Hat Enterprise Linux, please visit the Red Hat, Inc. website. The documentation for Red Hat Enterprise Linux is available on the Red Hat, Inc. website.

Alternatively, you can also use curl in the CLI of your host:

```
curl http:// [IP_of_the_host]
```

## 5.2 PHP installation with PHP-FPM

A very popular configuration of NginX is to run it with PHP-FPM, which increases performance of handling PHP dramatically. To add this functionality to your NginX installation, follow the steps 2 and 3 in the link below.

https://tecadmin.net/install-nginx-php-fpm-centos-8/

Notes:

As is often the case with online tutorials, especially from sources that are not as well established as the one we are using here, the information is not always completely correct and needs to be applied intelligently. The tutorial that we are using is no different. If you use it as-is, it will not work. Try to find the errors and correct where necessary. If you can't find them, you can get some inspiration below.

1. In Step 2 we need to install the remi repository. Unfortunately, the version mentioned here is no longer valid. Use version `remi-release-9.rpm` instead of `remi-release-8.rpm`.
2. When configuring PHP-FPM, the tutorial assumes default `user`- and `group`account for NginX of `www-data`. In the NginX installation that we are using, this might not be correct. Verify the `user` used by NginX in the general Nginx configuration file and replace if necessary.

## 5.3 Add Server Block

Just like we use Virtual Hosts in Apache, we use Server Blocks in NginX. Follow Step 4 and 6 (Step 5 should already be OK) from the same online tutorial in the previous exercise.
https://tecadmin.net/install-nginx-php-fpm-centos-8/

Notes:

Again, some alterations need to be made to the instructions in the tutorial. Try to solve these on your own, using the knowledge you acquired earlier (theory and labs). If you get stuck, you can try the tips below.

1. As long as we don't have a nameserver where we can create our own domains or an external computer where we can alter the `hosts` file, using a webbrowser in an external computer with name based Virtual Hosts or server blocks, such as this example, will not work. Only option is to change the hosts file on the host itself and use curl.
2. This lab uses a domainname of example.com. We have not yet configured this in our hosts file, so either you add it there, or you use website1.local from the previous exercise instead of example.com.
3. When testing the nginx configuration after adding the proposed example.com.conf file, an error might occur. Added to the error statement, which is rather vague in its description, a successive action is proposed to gather more information on the problem: "journalctl -xe". If you read the output of this command carefully, and combine it with reading the nginx.conf file, you should be able to find the conflict and resolve it. [hint: there can only be one default website].

## 5.4 Exercise: Working with Server Blocks in Nginx

Using the knowledge acquired in previous tutorials and exercises, build a similar configuration as described in 3.2 and 3.3 with website1.local, website2.local and website3.local, but this time in Nginx.

## 5.5 Exercise Docker nginx installation

Use this for inspiration to start a Docker nginx container with these specifications:

- Accessible through port 80 on the host
- A web page that is presented through index.html that displays the text "Nginx in Docker". You might need to check previous Apache tutorials.
- Can you create a website1.local and website2.local in this container, similar to the configuration in exercise 1 in chapter 3.2? You might also need to check the information on working with volumes in Docker. Remember, we don't like to make changes inside the container, we will always prefer to create links to directories on the host through volumes, so we can persist the data.

## 5.6 Exercise: OPTIONAL: Apache as webserver with NginX as reverse proxy

NginX is not only very popular as a webserver, but also as reverse proxy server for another webserver, such as Apache. Try to build a configuration with website1.local, website2.local and website3.local on Apache, but NginX as reverse proxy before the Apache webserver. You can find more information in these resources:

https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

https://www.nginx.com/resources/wiki/start/topics/examples/full/

Also, remember this is a CentOS distribution which is secured by SELinux. You will need open it up a bit to make it all work. You can do this with this command:

```
sudo setsebool -P httpd_can_network_connect true
```