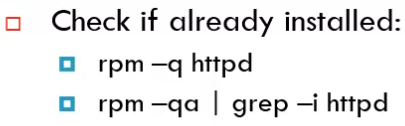
**Basics of Apache Webserver**

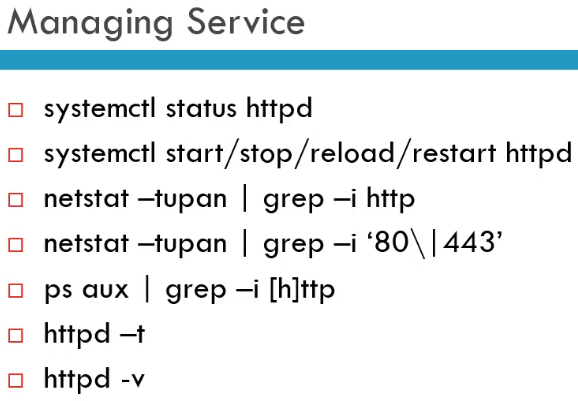


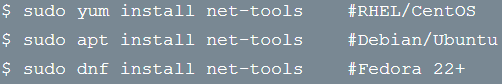






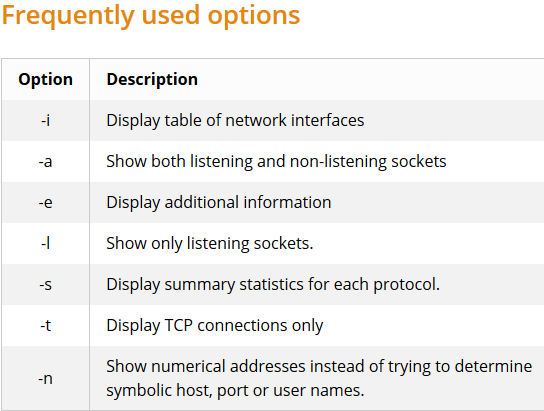
RedHat Package Manager maintains a database of all the installed packages in the system. We can query the db with the -q flag.





You need **net-tools** in order to use **netstat**

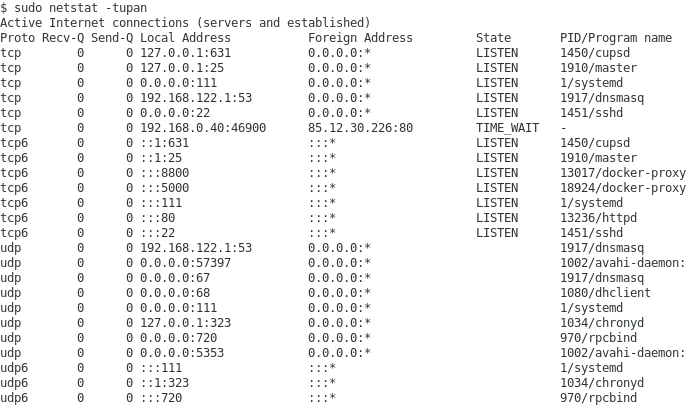
-> available on both Linux and Windows

**Netstat**

**netstat** command is a useful command to reveal a network status of your system. It allows a system administrator to keep track of any network connections by querying their status, origin and destination.

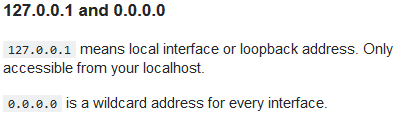
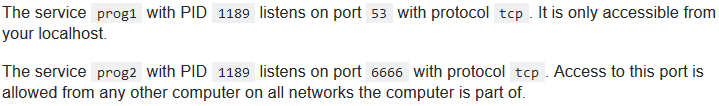
Furthermore, netstat is the all-in-one networking monitoring tool as it can also be used to display route tables, interface statistics, masquerade connections, as well as multicast memberships. **ss** command is a future successor of the **netstat** command.

* For most of the functions an **administrative** **privileges** are required to execute the netstat command

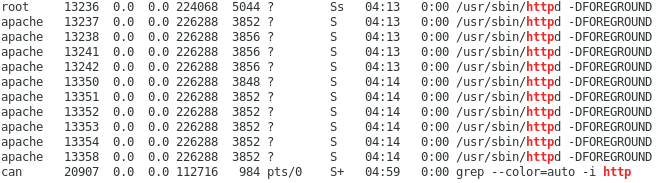








**ps aux | grep -i http**





**httpd -t**

When you change the configuration, you can make sure whatever you changed had the correct syntax before restarting or reloading the server.



**httpd Options**

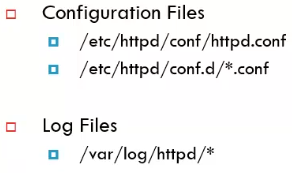


**A screenshot of a cell phone

Description automatically generatedhttpd -S**

the **apache2ctl** -S or **httpd -S** commands provide a report on currently running virtual hosts, containing the port that the host is listening on, the name of the virtual host (i.e. the domain), and information about the location of the site’s configuration settings including file names and line numbers.



The main config is **httpd.conf**

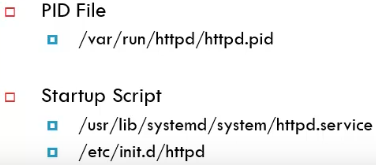
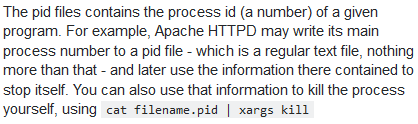
This file also includes anything in the **conf.d/** directory that ends with **\*.conf**







All log files from every host will land here(?), you can change that.



In older systems it is contained in the init.d directory

**Start Service on Boot**:

* systemctl enable httpd

# IncludeOptional won't give an error if the folder/files are not found

#**IncludeOptional conf.d/\*.conf**

# This will throw an error when trying to stop or restart the server

**Include /usr/local/etc/httpd/conf.d/\*.conf**

**FOR MAC OS**:

The main configuration file for apache was in **/usr/local/etc/httpd/httpd.conf**

I tried to write in the **httpd.conf**:

A screenshot of a cell phone

Description automatically generated**IncludeOptional conf.d/\*.conf** but it searched the ServerRoot folder for a conf.d folder and wasn’t able to find it. But it also did not throw an error.

(**ServerRoot** **/usr/opt/etc/httpd/**)

* The default configuration of apache also gives the full path when they use **Include**:

Apache Web Server

Table

Description automatically generatedTable

Description automatically generated

**Apache Virtual Hosts**

When you start a web server software on a server (like Apache or Nginx), that software is going to binds itself to one or more network ports on the server. Usually for a web server, it is going to be port 80 (for http traffic) or 443 (for https traffic with TLS). When a new request or package comes to the server, the operating system looks at the port that the packet was sent to, the destination port. It’s going to use that destination port to determine which the application this package should be forwarded to. This means that your server can only have one application which can receive packages at port 80 or 443. Let’s say we want to host 2 or more different websites on the same web server. The server needs to be able to tell when a packet comes to the server, requesting a website, it needs to be able to tell what website was requested. Since the port will be the same for every single http or https packet which comes to the server, the port number alone is not enough to determine which website the client tries to access.

All that the operating system/networking stack is going to do is send the packet to apache, which will be bound to the ports. From there Apache decides what the user wants and what it sends back.

For this purpose, Apache has the concept of Virtual hosts. With virtual hosts Apache can serve different websites:

* Based on the **IP address** that’s being requested
* Based on the **domain name** which is requested

The IP address version is for example when a server has multiple public IP addresses / network cards, or if its configured in a way so it talks on several different IP addresses. This is a very unusual setup.

* If client requested 1.2.3.4:80 then send Website-A,
* If request was for 2.2.2.2:80 then send Website-B

The named virtual hosts is what is generally used because most servers have/need a single public IP address and they can have multiple domain names in the DNS lookup tables which resolves all to the same IP address. This way once the server gives the packet to Apache, Apache can look inside the HTTP-Headers to see to which Host the packet is being sent to (Host: <host>:<port>) and then decide which directory it will serve.

We can’t have Apache and Nginx both having virtual hosts listening on the same port. Because before the packet gets to either of those webservers, the operating system needs figure out which software it needs to forward it, depending on the port number. If we do want to run both on the same server, one of them needs to see every packet first before redirecting it to the other, that will be a **reverse proxy**. Lets say Apache receives first a certain server name, its going to turn around and internally send the request to Nginx on a different port. Nginx responds to Apache, Apache responds to the client.

**Example Configuration**

Text

Description automatically generatedOur goal now is to setup and configure Apache, so that we will be hosting 3 different websutes on the same server. With the help of virtual hosts, we will serve different websites depending on the requested host name.

* I’ve added 3 domain names to /etc/hosts. All of these domain names will resolve to the same IP address, which is 127.0.0.1
* Our Vagrant with Ubuntu runs at host port 8282 (will go to port 80 in Ubuntu)

Installing apache: **sudo apt install apache2**

* On Redhat based distros it is named **httpd** and on Debian based distros the package is named **apache2**. Apache is the name of the organization and httpd is the name of the software, see the ‘Apache Basics’ block for more info.

Checking status: **systemctl status apache2** we can see that it is running

All of the 3 websites are returning the same webpage now. Let’s start to configure Apache.

The main configuration folder of Apache is **/etc/apache2** which has the following content inside:

Table

Description automatically generated

The main configuration file is the **apache2.conf** file. This is read when apache looks for the config. Here we can set global settings which then can be overridden by website specific config files.

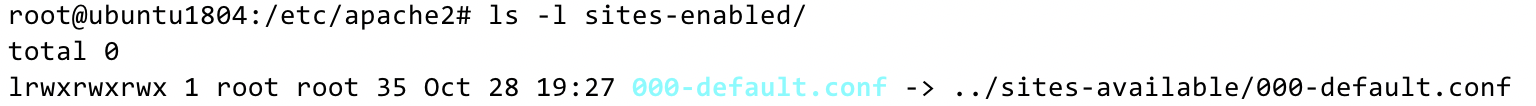
When we look at the bottom of the config file, we see the following lines (left Ubuntu, right CentOs).

Graphical user interface, text

Description automatically generated



The idea is: after all the global configurations are done, include now specific config files which then add or override the global configurations. This is mainly done so the main config file shouldn’t be touched often and each directory or each website can define its own configurations. These folders are not special at all, these can be changed to any other directory and any other wildcard.



The **000-default.conf** looks like this:

Text, letter

Description automatically generated

<VirtualHost **\*:80**>

* the wildcard is ONLY for IP addresses, not for host names.

13:30

Text

Description automatically generated

Todo: show vhost with domain name instead of IP address

Text, letter

Description automatically generatedText, letter

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

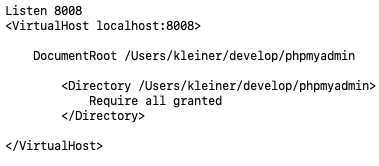
A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Setup Php Myadmin with Apache**

The Listen directive is needed. Now I can access phpMyAdmin under localhost:8008

A screenshot of text

Description automatically generated

**Setting up different projects to different Urls paths**

localhost/phpmyadmin will open php myadmin site

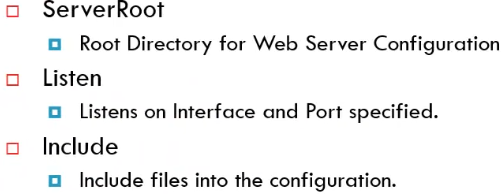
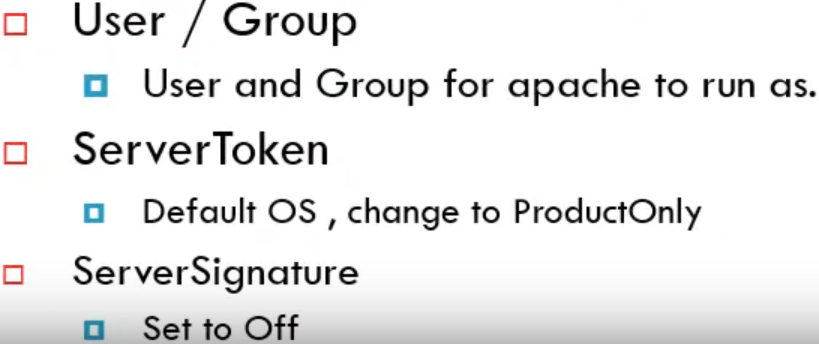
localhost/dodp/typo3 will open the BE login site.

This approach was problematic, maybe the subpaths can be redirected or overridden by mod\_rewrite or smth.

A picture containing text

Description automatically generatedI had some problems with typo3, when there is a subpath like ‘/dodp/ then certain things didn’t work.

This is one of the ways to fix it and map different ports to different projects.



**Directory**: is used to set per directory basis permissions and overwrite configurations to a particular directorry

**IfModule**: includes additional modules. For example the DirectoryIndex module will load the index.html file if a user didn’t specify any file name

**/etc/hosts File**

Before the Domain Name System (DNS) computers were addressed by their IP address and a hosts file were used to store domain names mapped to those IP adresses in order to make life easier. With the introduction of DNS, this file became less important.

Every request **first** **looks** in the **/etc/hosts** file and if it finds a match then it sends the request to that IP. If not then a DNS lookup is done.

This is useful for local development where developers don’t want tot type the IP address in their code and just use the future domain name of the website so that it will work without any changes when deployed.

/etc/… resolver.