Webservers

Server OS

Contents

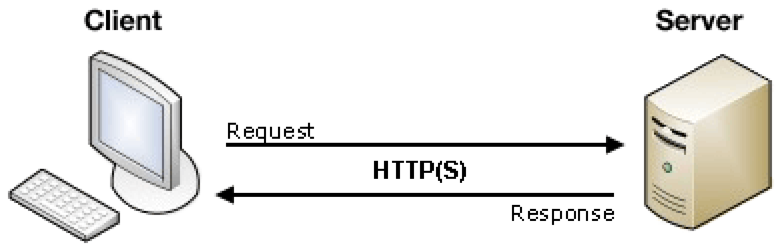
- HTTP(S)
- Proxy
- Webserver overview
- Apache
- NginX

HTTP(S)

- When talking about the internet, most people refer to HTTP(S).
  - The internet is obviously a lot more.
  - It does show the importance of the HTTP(S) protocol.
- Hypertext Transfer Protocol (Secure)
- Developed by Tim Berners-Lee
- World Wide Web

HTTP(S)

- Client – server / request – response model
  - Clients: webbrowsers
    - GET the content from the webserver
    - e.g. Chrome, Firefox, Edge, Internet Explorer, Safari,…
  - Servers: webservers
    - Provide the content to the client
    - e.g. Apache, NginX, IIS,…



HTTP(S)

- Application layer protocol
- Uses TCP in the transport layer: sessions
- HTTP: unsecured
  - Default port 80
- HTTPS: secured
  - Default port 443
  - Uses SSL (older version) or TLS (newer version) as encryption protocol
    - HTTP over SSL or HTTP over TLS
  - Secure: authentication, encryption of communication, protection against man-in-the-middle attacks

HTTP(S)

- Frequent HTTP(S) error messages
  - 4xx client errors:
    - 400: Bad request
      - Request is using invalid syntax
      - e.g. corrupt cookie, faulty browser
    - 401: Unauthorized
      - Request requires authentication which has not been provided or was invalid
    - 403: Forbidden
      - Request is understood by the server but the client does not have (file)permission or it is a prohibited action
        - e.g. on linux server, make sure the user account of the webserver (often www-data) has Read permission on the webfiles
        - e.g. missing index.html file
    - 404: Not Found
      - Requested resource is not available on server

https://www.digitalocean.com/community/tutorials/how-to-troubleshoot-common-http-error-codes

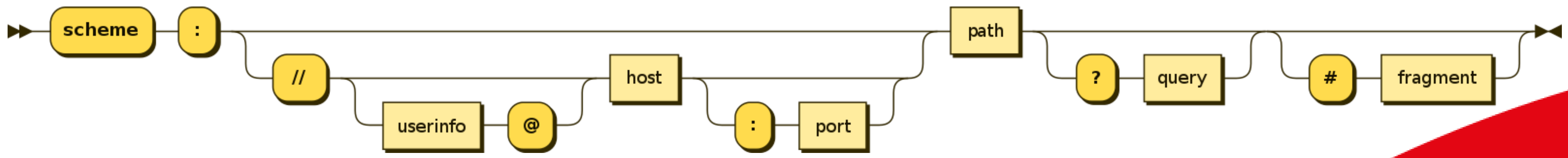
HTTP(S)

- Frequent HTTP(S) error messages
  - 5xx server errors:
    - 500: Internal Server error
      - Badly configured server
    - 502: Bad Gateway
      - Request sent to a proxy server, but the proxy server is not receiving a valid response from the backend webserver(s)
    - 503: Service Unavailable
      - Server is overloaded or under maintenance
      - e.g. DDOS attack
    - 504: Gateway Timeout
      - Request sent to a proxy server, but the proxy server is not receiving a valid response from the backend webserver(s) within the allowed time period

https://www.digitalocean.com/community/tutorials/how-to-troubleshoot-common-http-error-codes

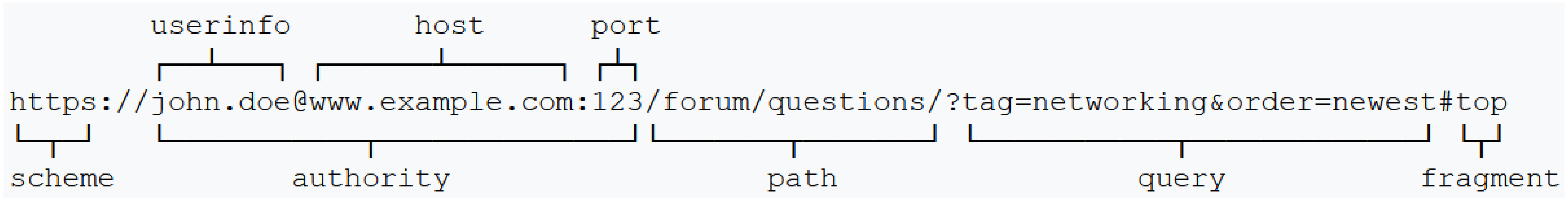
HTTP(S)

- URL
  - Uniform Resource Locator (URI with location of resource/file)
  - Scheme : http, https, ftp, mailto, file, data, and irc
  - Userinfo: (optional): username
  - Host: DNS-name or IP
  - Path: Path to resource on server
  - Query and fragment: optional



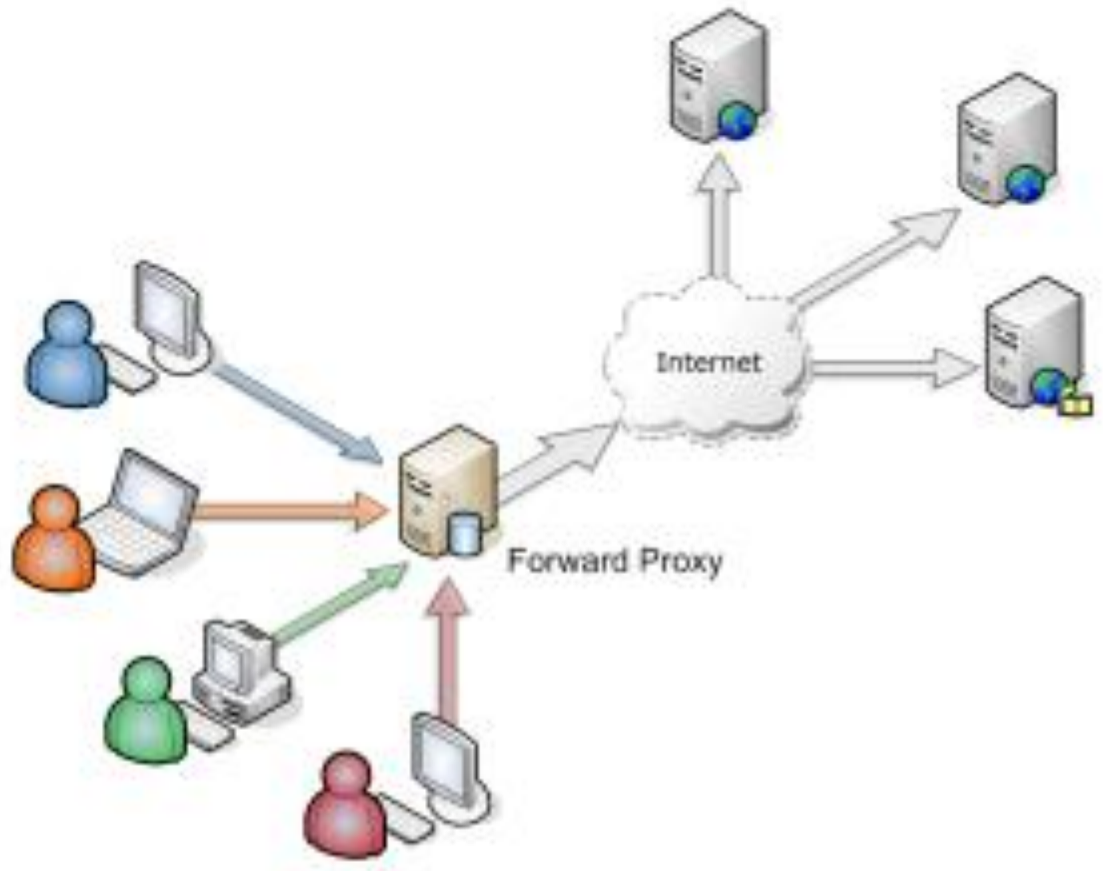
HTTP(S)

- URL
  - HTTPS example


userinfo
host
port
https://john.doe@www.example.com:123/forum/questions/?tag=networking&order=newest#top
scheme
authority
path
query
fragment

Proxy

- Forward proxy



https://community.pulsesecure.net/t5/Pulse-Secure-vADC/Using-Stingray-Traffic-Manager-as-a-Forward-Proxy/ta-p/28747

Proxy

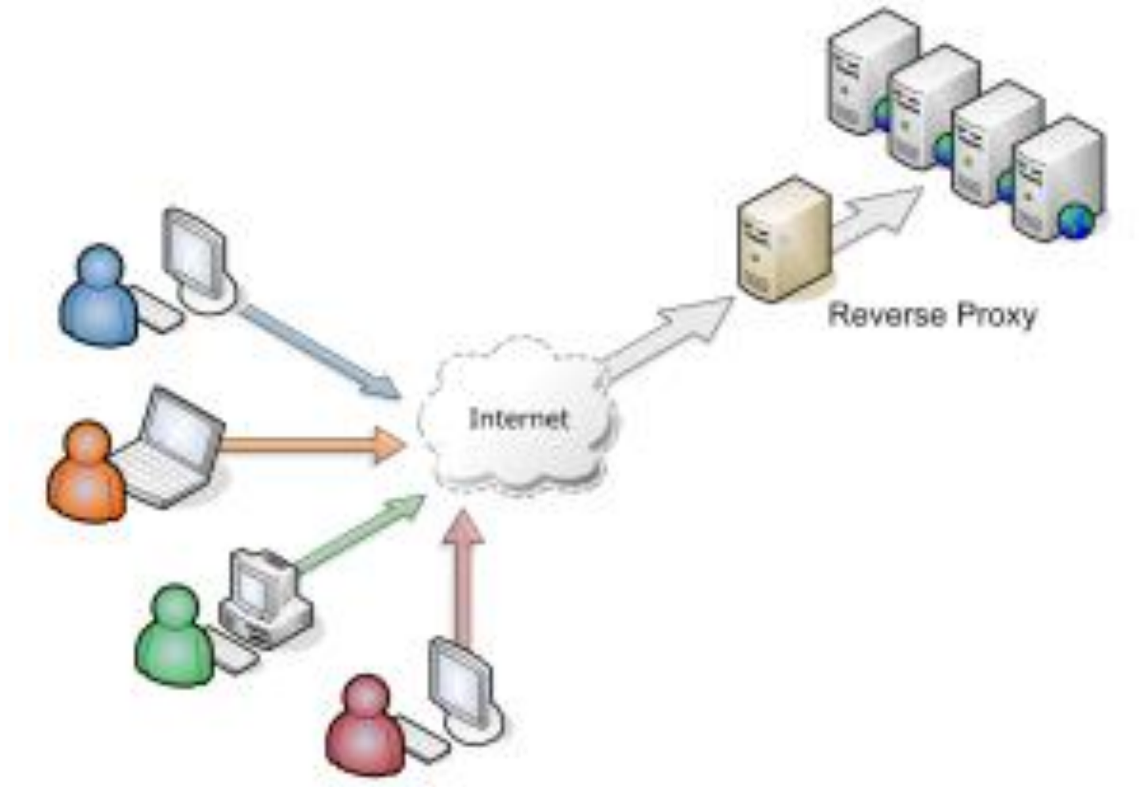
- Forward proxy
  - Are used for filtering and caching
  - Popular forward proxy software:
    - Apache, squid, IIS (Windows), …
  - An HTTP request from inside the network to an outside webserver is sent to the proxy server instead of directly to the outside webserver.
  - Direct HTTP requests to the internet are blocked by the firewall, making the proxy server the only gateway to the www.

Proxy

- Forward proxy
  - The proxy can perform filtering:
    - Certain URI's are not forwarded (filtered)
  - The proxy server will get the content from the webserver for the webclient and cache the content.
  - If a new request is sent to the proxy server, it will serve the content directly to the client from its local cache.
    - Speed gain (traffic over lan versus wan)
    - Less bandwidth usage over the wan internet connection

Proxy

- Reverse proxy



https://community.pulsesecure.net/t5/Pulse-Secure-vADC/Using-Stingray-Traffic-Manager-as-a-Forward-Proxy/ta-p/28747

Proxy

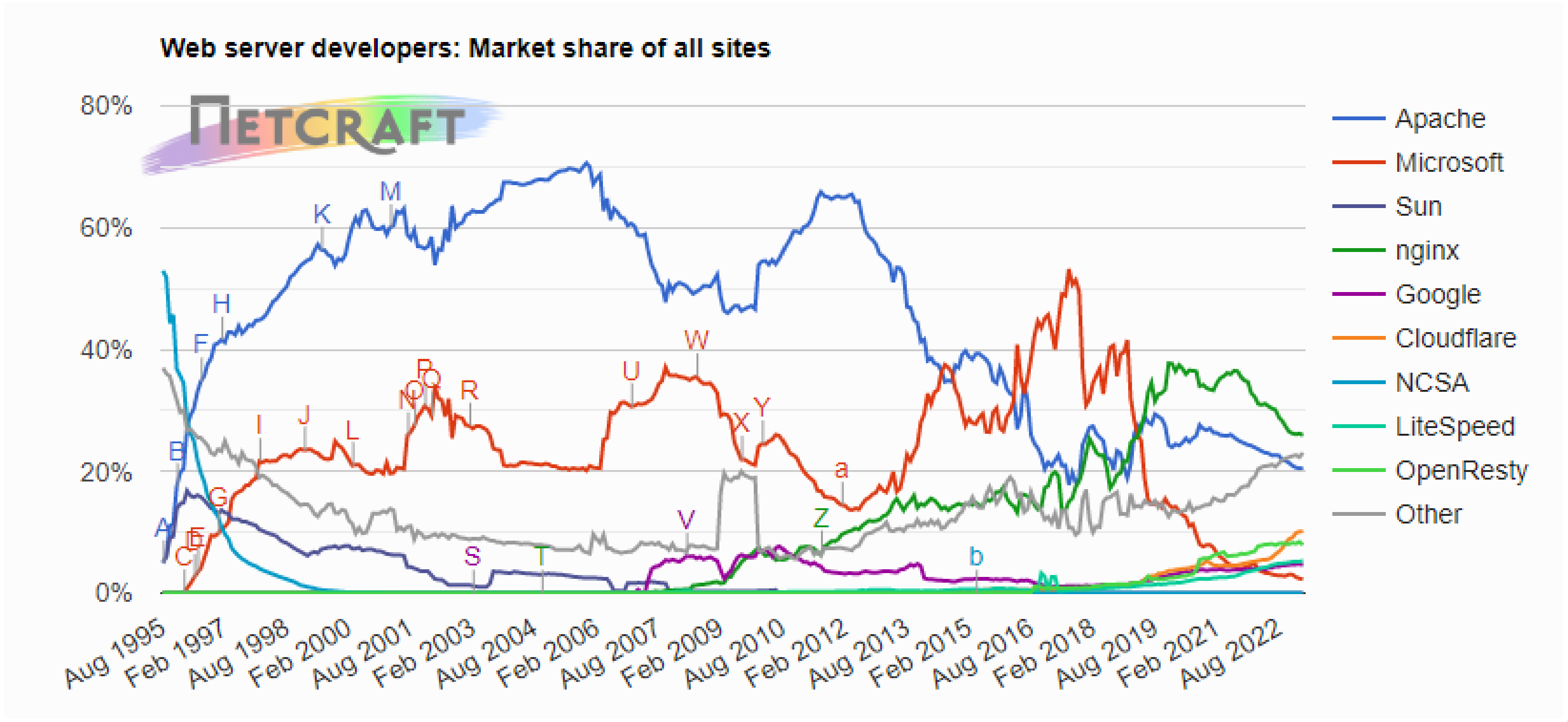
- Reverse proxy
  - On the other side of the HTTP-communication, webservers can also use proxy servers.
    - Reverse proxy
  - Popular reverse proxy software:
    - Haproxy, Squid, NginX, Varnish…
  - The reverse proxy will also be positioned between the internet and the inside of the LAN, this time the webservers.
  - It will take all incoming GET requests from outside webclients and forward them to the the actual webservers and sent the response to webclient.

Proxy

- Reverse proxy
  - In doing so, several tasks can be handled by the proxy, diminishing the load on the webservers:
    - Static content might be cached, and delivered directly to the client:
      - Web acceleration
    - Content compression
    - TLS encryption/decryption for HTTPS
    - Authentication
    - Load-balancing
  - Also helps protecting the webservers:
    - Hidden behind the proxy
    - DOS attacks

Webservers overview

- Many different webserver software packages are available, with changing popularity over the years.



https://news.netcraft.com/archives/category/web-server-survey/

Webservers overview

- Apache
  - Since 1999
  - Open-source
  - Free
  - Cross-platform: Linux, Unix, BSD, but also Windows
  - Vast majority on Linux
  - Often in LAMP setup: **L**inux + **A**pache + **M**ySQL/**M**ariaDB + **P**HP/**P**ython/**P**earl
  - Initial design aimed towards webserver

Webservers overview

- Apache
  - Characteristics
    - Extensible through extensive library of modules , e.g. PHP
      - Inserted in the work processes
      - Dynamically loaded while running the server
    - Can handle dynamic content internally itself through modules
      - Simpler configuration
    - Directory based configuration through `.htaccess` files
      - Allows decentralized configuration of websites
      - Are read with every request: allows changing configuration of a site without restarting the service
    - Web requests are mostly interpreted as physical requests to files
      - The filesystem hierarchy is the same as the available document tree
      - See also the use of the `.htaccess` files

https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations

Apache

- Architecture
  - Apache server: httpd
    - Answers request and provides the content
    - In itself can only deliver static content
  - Modules
    - Apache can be extended by adding modules for added functionality
      - e.g. authentication, security, php, compression,…
      - Dynamically loaded
      - Only add modules you need
        - Less resources
        - Smaller attack surface

Apache

- Architecture
  - Virtual Hosts
    - One Apache server can deliver multiple different websites by creating them as virtual hosts
  - Security
    - Several components are provided for securing the website, like access control, authentication and encryption
  - Logging

Apache

- Configuration files
  - Apache uses different plain-text files for its configuration:
  - Apache wide main configuration file:
    - `/etc/httpd/conf/httpd.conf`
  - Additional configuration files:
    - Extra configuration files can be added and will be read by Apache if placed in:
    - `/etc/httpd/conf.d/*.conf`
  - Module configuration files:
    - `/etc/httpd/conf.modules.d/*.conf`

https://httpd.apache.org/docs/2.4/configuring.html

Apache

- Configuration files
  - Example configuration files:
    - Apache provides several example configuration files when installing httpd:
    - `/usr/share/doc/httpd/`
      - `httpd-default.conf`
      - `httpd-vhosts.conf`
      - `proxy-html.conf`

https://httpd.apache.org/docs/2.4/configuring.html

Apache

- Directives
  - Rules, instructions and settings for how the webserver should behave.
  - General directives: for the whole server
  - Scoped directives: for a specific part of the server
    - Directories, files, modules and virtual hosts
    - Default in `httpd.conf` is block all, meaning explicit overrides need to be defined

```
<Directory />
    AllowOverride none
    Require all denied
</Directory>
```

    - Follows syntax as in the example above

https://httpd.apache.org/docs/2.4/configuring.html

Apache

- Commonly used directives
  - `ServerRoot`
    - top-level directory of your Apache configuration
    - `/etc/httpd`
    - configuration files, log files and also your modules
  - `Listen`
    - Default is (port) 80
    - Default all IP addresses of the server
    - Can be a different port or specific IP
    - https://httpd.apache.org/docs/2.4/bind.html

Apache

- Commonly used directives
  - `Include`
    - Which config files to include
    - e.g. `/etc/httpd/conf.d`
  - `User / Group`
    - Which user and which group to run Apache as
    - Good practice to create a specific user for Apache
  - `ServerAdmin`
    - Email address where errors are sent to

Apache

- Commonly used directives
  - `DocumentRoot`
    - Default directory for the actual web documents
    - By default `/var/www/html`
    - https://httpd.apache.org/docs/2.4/urlmapping.html
  - `Options`
    - Controls which server features are available for a particular directory
  - `AllowOverride`
    - Additional directives can be created per directory
    - `.htaccess` file in the directory itself
    - Often used for scoped directives

Apache

- Apache commands
  - `apachectl`
    - The main command used for managing Apache
      - `stop`
      - `start`
      - `restart`
      - `status`
      - `configtest`
        - validate configuration files
    - Often just a wrapper for the `systemctl` command
  - `httpd`
    - Command that can be used to test config files of virtual hosts
      - `httpd -t -D DUMP_VHOSTS`
        - Displays all configured virtual hosts

Apache

- Virtual Hosts
  - Run multiple sites from one server
  - Name-based
    - Uses the domain name in the request to forward to the correct virtual host
  - IP-based
    - Uses the IP-address in the request to forward to the correct virtual host
  - Name-based are far more common than IP-based
  - Often configured in separate configuration files inside `/etc/httpd/conf.d/*.conf`

https://httpd.apache.org/docs/2.4/vhosts/

Apache

- Virtual Hosts
  - Name-based virtual host example

```
<VirtualHost *:80>
    DocumentRoot "/www/example1"
    ServerName www.example.com

    # Other directives here
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot "/www/example2"
    ServerName www.example.org

    # Other directives here
</VirtualHost>
```

https://httpd.apache.org/docs/2.4/vhosts/examples.html

Apache

- Virtual Hosts
  - Name-based virtual host example with ports

```
<VirtualHost 172.20.30.40:80>
    ServerName www.example.com
    DocumentRoot "/www/domain-80"
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
    ServerName www.example.com
    DocumentRoot "/www/domain-8080"
</VirtualHost>
```

https://httpd.apache.org/docs/2.4/vhosts/examples.html

Apache

- Apache and SELinux
  - Security Enhanced Linux
  - Additional layer of security on many Linux distributions
  - Requires extra configuration for Apache
- Service on different port
  - `sudo semanage port -a -t http_port_t -p tcp [port_number]`
  - `sudo semanage port –l`
    - List all ports that are usable through selinux
- Allow different content folders
  - `sudo setsebool -P httpd_unified 1`

https://wiki.centos.org/HowTos/SELinux

Apache

- Modules

Add functionality to your server

Many modules available

Load only what's needed

Write your own modules

https://app.pluralsight.com/library/courses/https-advanced-services-linux-lfce/table-of-contents

Apache

- Modules
  - Install through the package manager of your Linux distribution.
  - List of all modules that come with Apache:
    - httpd.apache.org/docs/current/mod/
  - Location of modules:
    - `/etc/httpd/modules`
      - Symlink to external path for easier configuration with relative paths towards `ServerRoot`
  - Location of configuration files:
    - `/etc/httpd/conf.modules.d/`

Apache

- Modules
  - Syntax for loading modules:
    - `LoadModule [Module_name] [path_to_module_binary]`
    - e.g. `LoadModule alias_module modules/alias.so`
  - Display list of all loaded modules:
    - `httpd –D DUMP_MODULES`

Apache

- Modules
  - Frequently used Apache modules

Module Name	Purpose
mod_ssl	Implements SSL and TLS
mod_alias	Simple URL remapping
mod_rewrite	Rule based remapping of URLs
mod_status	Provides information on server activity and performance
mod_deflate	Compression before content is delivered to the client
mod_cgi	Application execution as defined in the script
mod_cgid	Application execution using an external CGI daemon

https://app.pluralsight.com/library/courses/https-advanced-services-linux-lfce/table-of-contents

Webservers overview

- NginX
  - Since 2004
  - Open-source
  - Free
  - Cross-platform: Linux, Unix, BSD, but also Windows
  - Vast majority on Linux
  - Initial design aimed towards webserver and proxy

Webservers overview

- NginX
  - Charasteristics
    - Light on resources
      - Great under heavy load
    - Extensible through modules, but often more complex configuration
    - Great for delivering static content
    - (Reverse) proxy
    - Load balancer
    - No directory based configuration
      - Faster
      - Centralized control of configuration is often more secure
    - Web request are handled mainly based on the URI

https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations

NginX

- Configuration
  - The NginX configuration was largely based on the Apache configuration
  - Many directives from Apache with the same name in NginX
  - Admins with knowledge in Apache adapt easily to NginX

NginX

- Configuration files
  - NginX wide main configuration file:
    - `/etc/nginx/nginx.conf`
  - Additional configuration files:
    - Extra configuration files can be added and will be read by Nginx if placed in:
    - `/etc/nginx/conf.d/*.conf`
  - Module configuration files:
    - `/usr/share/nginx/modules/*.conf`

https://httpd.apache.org/docs/2.4/configuring.html

NginX

- Directives
  - Rules, instructions and settings for how the webserver should behave.
  - Simple directives:
    - name and parameters separated by spaces and ends with a semicolon (`;`)
  - Blocked directives:
    - same structure as a simple directive
    - instead of the semicolon it ends with a set of additional instructions surrounded by curly brackets (`{ }`)
    - can have other directives inside braces: context
  - Directives placed in the configuration file outside of any contexts
    - Considered to be in the main context
    - `events` and `http` directives: main context
      - `server` in `http`
        - `location` in `server`

http://nginx.org/en/docs/beginners_guide.html

NginX

- Commonly used directives
  - `include`
    - Which config files to include
    - e.g. `include /etc/nginx/conf.d/*.conf;`
  - `user`
    - Which user and which group to run NginX as
    - Default `nginx` in CentOS

http://nginx.org/en/docs/http/ngx_http_core_module.html#directives

NginX

- Commonly used directives
  - `http`
    - Main context for the HTTP server directive
  - `server`
    - Used to define a `server` block (see Server Blocks)
  - `location`
    - Used to define a `location` block (see Location Blocks)
    - Located within a `server` block

http://nginx.org/en/docs/http/ngx_http_core_module.html#directives

NginX

- Commonly used directives
  - `listen`
    - Sets the address and port for IP, or the path for a UNIX-domain socket on which the server will accept requests
    - Default port 80
    - Default all IP addresses of the server
      - `Listen 80;               #Listen on all IPv4-addresses on port 80`
      - `Listen [::]:80;          #Listen on all IPv6-addresses on port 80`
      - `Listen 10.3.50.50:80   #Listen on Listen on IPv4-address 10.3.50.50 on port 80`
    - Many other configuration options for listen directive available

http://nginx.org/en/docs/http/ngx_http_core_module.html#directives

NginX

- Commonly used directives
  - `root`
    - Path to the folder that contains the actual web documents
  - `server_name`
    - Used to define the domain name that will answer the incoming request

http://nginx.org/en/docs/http/ngx_http_core_module.html#directives

NginX

- NginX commands
  - `sudo systemctl`
    - `start nginx`
      - Starts the NginX service
    - `status nginx`
      - Command to check the status of the NginX service
        - Running (`active (running)`)
        - Start with the Operating System `(enabled)`

NginX

- NginX commands
  - `nginx`
    - The main command used for managing NginX
    - `-s`
      - `stop` – fast shutdown
      - `quit` – graceful shutdown
      - `reload` – reloading the configuration file
      - `reopen` – reopening the log files
    - `-t`
      - Test the configuration file(s)
    - `-v`
      - Display the installed version of NginX

NginX

- Server Blocks
  - Configuration blocks containing separated configuration contexts
  - Hierarchically structured
  - Every request runs through the hierarchy to determine which configuration block is applied
    - First selected on the `listen` directive, then the `server_name` directive
    - In case of competing directives, the one that is more specific will win
    - e.g. `listen 192.168.1.10;` will win over `listen 80;`, even if the second one has a nameserver defined, while the first one doesn't.

https://www.keycdn.com/support/nginx-location-directive
https://www.digitalocean.com/community/tutorials/understanding-nginx-server-and-location-block-selection-algorithms

NginX

- Server Blocks
  - `server` block
    - Subset of configurations that define a virtual server
    - Multiple server blocks are possible to decide which block will handle the request based on domain name, IP address and port
    - Similar to Virtual Hosts in Apache: run multiple sites from one server
  - `location` block
    - Located within a server block
    - Defines how requests are processed for different URIs and resources

https://www.keycdn.com/support/nginx-location-directive
https://www.digitalocean.com/community/tutorials/understanding-nginx-server-and-location-block-selection-algorithms

NginX

- Server Block examples
  - Name-based Server Blocks, serving static files

```
server {
  server_name www.domain1.com;
  access_log logs/domain1.access.log main;

  root /var/www/domain1.com/htdocs;
}

server {
  server_name www.domain2.com;
  access_log  logs/domain2.access.log main;

  root /var/www/domain2.com/htdocs;
}
```

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

NginX

- Server Block examples
  - Port-based Server Blocks, serving static files

```
server {
  listen 80;
  server_name www.domain.com;
  access_log logs/domain.access-80.log main;

  root /var/www/domain.com-80/htdocs;
}

server {
  listen 8080;
  server_name www.domain.com;
  access_log  logs/domain.access-8080.log main;

  root /var/www/domain.com-8080/htdocs;
}
```

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