Rewrite rules

https forcieren

Redirect "/" "https://www.miteinander-esslingen.de/"

canonical subdomain rewrite

```
Rewrite Engine on
RewriteCond %{HTTP_HOST} !^www.* [NC]
RewriteCond %{HTTP_HOST} ^owncloud\.miteinander-esslingen.de$
RewriteRule ^(.*) https://www.miteinander-esslingen.de/owncloud/ [L,QSA]
```

Benchmark Testing

Das Benchmark Tool ist Teil von apache2-utils

```
ab -n 100 -c 10 http://www.netzwissen.de/
```

```
-n requests Number of requests to perform
-c concurrency Number of multiple requests to make
-t timelimit Seconds to max. wait for responses
-p postfile File containing data to POST
-T content-type Content-type header for POSTing
-v verbosity How much troubleshooting info to print
-w Print out results in HTML tables
-i Use HEAD instead of GET
-x attributes String to insert as table attributes
-y attributes String to insert as tr attributes
-z attributes String to insert as td or th attributes
-C attribute Add cookie, eg. 'Apache=1234. (repeatable)
-H attribute Add Arbitrary header line, eg. 'Accept-Encoding: gzip'
Inserted after all normal header lines. (repeatable)
-A attribute Add Basic WWW Authentication, the attributes
are a colon separated username and password.
-P attribute Add Basic Proxy Authentication, the attributes
are a colon separated username and password.
-X proxy:port Proxyserver and port number to use
-V Print version number and exit
-k Use HTTP KeepAlive feature
-d Do not show percentiles served table.
-S Do not show confidence estimators and warnings.
-g filename Output collected data to gnuplot format file.
```

```
-e filename Output CSV file with percentages served
-h Display usage information (this message)
-Z ciphersuite Specify SSL/TLS cipher suite (See openssl ciphers)
-f protocol Specify SSL/TLS protocol (SSL2, SSL3, TLS1, or ALL)
```

Mit Authentikation:

```
ab2 -A auth-username:passwort -c 10 -n 100
http://www.netzwissen.de/gallery/main.php/v/thg82/
```

Apache Server Tuning

Quelle:
http://www.woktron.com/secure/knowledgebase/133/How-to-optimize-Apache-performance.html

```
Apache.conf
```

To start, open the Apache configuration file and locate the directives section: If you are using nano, vi or vim: once you open the file, you can find the directives by scrolling through the file. Using VI or VIM you can also search by typing forward-slash '/' and typing the exact string that you are looking for (search is case specific).

```
  Timeout
```

The Timeout setting is the number of seconds before data "sends" or "receives" (to or from the client) time out. Having this set to a high number forces site visitors to "wait in line" which adds extra load to the server. Lowering the 'Timeout' value too much will cause a long running script to terminate earlier than expected.

A reasonable value is 100 for Virtual Private Servers, or heavily loaded dedicated servers. For Dedicated Servers under normal load the default value of 300 is sufficient.

```
  KeepAlive
```

KeepAlive enables persistent connections on the web server. This setting should be On unless the server is getting requests from hundreds of IPs at once. High volume and/or load balanced servers should have this setting disabled Off to increase connection throughput.

```
  MaxKeepAliveRequests
```

This setting limits the number of requests allowed per persistent connection when KeepAlive is on. If it is set to 0, unlimited requests will be allowed. When using DirectAdmin, this directive can be found in: /etc/httpd/conf/extra/httpd-default.conf

It is recommended to keep this value at 100 for virtualized accounts like VPS accounts. On dedicated servers it is recommended that this value be modified to 150.

```
  KeepAliveTimeout
```

The number of seconds Apache will wait for another request before closing the connection. Setting this to a high value may cause performance problems in heavily loaded servers. The higher the timeout, the more server processes will be kept occupied waiting on connections with idle clients. When using DirectAdmin, this directive can be found in: /etc/httpd/conf/extra/httpd-default.conf

The default value of 10 seconds is a good value for average server performance. This value should be kept low as the socket will be idle for extended periods otherwise.It is recommended that this value be lowered to 5 on servers under heavy load.

```
StartServers
```

sets the number of child server processes created on startup. As the number of processes is dynamically controlled depending on the load there is usually little reason to adjust this parameter. This value should mirror what is set in MinSpareServers.

```
MinSpareServers
```

Sets the desired minimum number of idle child server processes. An idle process is one which is not handling a request. If there are fewer spareservers idle then specified by this value, then the parent process creates new children at a maximum rate of 1 per second. Setting this parameter to a large number is almost always a bad idea.

```
    Virtual Private Server 5
    Dedicated server with 1-2GB RAM 10
    Dedicated server with 2-4GB RAM 20
    Dedicated server with 4+ GB RAM 25
```

```
MaxSpareServers
```

sets the desired maximum number of idle child server processes. An idle process is one which is not handling a request. If there are more than MaxSpareServers idle, then the parent process will kill off the excess processes.

```
ServerLimit
```

is only used if you need to set MaxClients higher than 256 (default). Do not set the value of this directive any higher than what you might want to set MaxClients to.

```
MaxClients
```

sets the limit on the number of simultaneous requests that will be served. Any connection attempts over the MaxClients limit will normally be queued, up to a number based on the ListenBacklog directive. Once a child process is freed at the end of a different request, the connection will then be serviced.

For non-threaded servers (i.e., prefork), MaxClients translates into the maximum number of child processes that will be launched to serve requests. The default value is 256; to increase it, you must also raise ServerLimit. this and ServerLimit should be the same or very close with MaxClients never exceeding ServerLimit. For servers under high load this value should be increased. See below for more information on how to define the maxclients directive.

How to define the MaxClients directive

A simple calculation for MaxClients would be: (Total Memory – Critical Services Memory) / Size Per Apache process. I define Critical Services as services such as mySQL, Plesk, DirectAdmin; any service that is required for proper operation of your server.

I've used the following commands via shell to determine values for Total Memory, OS Memory, MySQL Memory, and Apache Process Size

```
TOTAL MEMORY
[root@vps httpd]# free -m
total       used       free     shared    buffers     cached
Mem:          1002        599        402          0         28        337
-/+ buffers/cache:        233        769
Swap:         2047        124       1922
```

```
MYSQL MEMORY
[root@vps httpd]# ps aux | grep 'mysql' | awk '{print $6}'
408
21440
704
```

```
APACHE PROCESS SIZE
[root@vps httpd]# ps aux | grep 'httpd' | awk '{print $6}'
22468
11552
41492
40868
41120
41696
39488
41704
15552
16076
16084
728
```

In this case the server has 1002Mb of memory allocated, xx used by the OS itself, 21Mb used by mySQL, and each Apache thread averages about 30Mb. MaxClients = (1002 – 21) / 30 therefore MaxClients = 32.7

```
MaxConnectionsPerChild
```

MaxConnectionsPerChild sets the limit on the number of connections that an individual child server process will handle. After MaxConnectionsPerChild connections, the child process will die. If MaxConnectionsPerChild is 0, then the process will never expire.

Setting MaxConnectionsPerChild to a non-zero value limits the amount of memory that process can consume by (accidental) memory leakage. See below for more information.

How to define the MaxConnectionsPerChild directive

A good calculation for MaxConnectionsPerChild would be: (total amount of daily requests / total number of daily processes)

Determining these values is a bit more complex as it requires some type of statistics package or thorough knowledge of interpreting Apache access logs.

As this does not adversely effect memory usage, only cpu time to cycle the process if you are unable to determine this information the standard 1000 should be used.

Thus a good configuration for this server would be:

```
StartServers       2
MinSpareServers    3
MaxSpareServers    3
ServerLimit       30
MaxClients        30
MaxRequestsPerChild  1000
```