Zusammenfassung Tag 28

Apache2-Installation und Serverdienst verwalten

- Installationsbefehl: `apt install apache2`
- Versionsprüfung und weitere Informationen: `apache2 -V`
- Status des Serverdienst prüfen: `systemctl status apache2`
- Elternprozess forkt mehrere Kindprozesse, aus Sicherheitsgründen mit unprivilegiertem Benutzer
- Server-Root-Verzeichnis: `/etc/apache2`
- Hauptkonfigurationsdatei: `apache2.conf`
    - Beinhaltet Dokumentation
    - Diverse vorbereitete Direktiven
    - Allgemeine Einstellungen zum Verhalten des Servers
    - Zahlreiche Include-Direktiven, die diverse Dateien und Verzeichnisse einbinden
    - Apache-typischen Tags, ähnlich wie bei HTML, wobei eine Sektion immer durch ein öffnendes Tag gestartet und ein schließendes beendet wird
- Verzeichnis-Hierarchie von der Apache-Debian-Installation, die auch in allen Debian-Derivaten zum Einsatz kommt. Es gibt aktivierte Sites, Module und Zusatzkonfiguration
- Für jeden Ressourcentyp gibt es einmal ein Verzeichnis `available`, also verfügbar und zum anderen `enabled`, also aktiv
- In den Available-Verzeichnissen sind die eigentlichen Ressourcen beheimatet
- In den Enabled-Verzeichnissen dagegen Symlinks auf die entsprechenden Ressourcen in den analogen Available-Verzeichnissen

Apache2-Installation unter CentOS

- Auf CentOS wird der Apache-Webserver nicht über das Paket *apache2* installiert, wie bei Debian-Derivaten, sondern über *httpd*
- Installationsbefehl: `yum install httpd`
- Versionsprüfung und weitere Informationen: `httpd -V`
- Server-Root-Verzeichnis: `/etc/httpd`
- Hauptkonfigurationsdatei: `httpd.conf`
    - Viele Aspekte, die bei Debian in eigene Dateien ausgelagert wurden, sind hier bereits enthalten
    - Das ServerRoot-Verzeichnis wird noch einmal explizit gesetzt

  - Die Direktive `Listen` besagt, auf welchem Port der Webserver lauschen soll
  - Wird keine Schnittstelle angegeben, bindet sich der Server an alle Schnittstellen
  - Die Direktive `DocumentRoot` ist das Hauptverzeichnis für die Webinhalte, die Apache bereitstellt. Jeder virtuelle Host hat ein eigenes DocumentRoot-Verzeichnis
  - Für das DocumentRoot-Verzeichnis können verschiedene Optionen und Einstellungen getroffen werden, z.B. Zugriffsrechte, Darstellung, etc.
  - Es wird bei CentOS fast alles zentral geregelt, wobei auch einige Bereiche ausgelagert wurden, wie in der Include-Direktive zu sehen: `../conf.d`

- Das Unterverzeichnis `conf.d` ist für die Einbindung weiterer Konfigurationsdateien vorgesehen, um die Hauptkonfigurationsdatei etwas schlanker zu halten
- In `welcome`.conf ist festgelegt, was passiert, wenn keine Startdatei zur Verfügung steht, z.B. Anzeige von `/usr/share/httpd/noindex/index.html`

Eine eigene Webseite erstellen

Hinweis:
CentOS hat per Default noch eine Firewall, die verhindert, dass eine Anfrage aus dem Netzwerk den Server erreicht. Diese muss ggf. deaktiviert werden: `systemctl stop firewalld` (kann in der Praxis natürlich auch durch Anpassen des Regelwerks geschehen)

- Wenn ein Webbrowser eine Verbindung mit dem Webserver auf dessen FQDN aufbaut (z.B. www.gulugulu.org), dann spricht er per Default Port 80/tcp an und fragt nach dem Inhalt des DocumentRoot-Verzeichnis, stellt also einen HTTP-Request `GET /`
- Wird die IP des Webservers im Browser eingegeben, fragt dieser ebenfalls nach /, also dem Inhalt aus dem DocumentRoot-Verzeichnis
- In beiden Fällen liefert der Webserver die Startdatei, z.B. index.html, index.htm, index.php, start.asp, o.ä.
- CentOS:
  - DocumentRoot-Verzeichnis CentOS: `/var/www/html`
  - Ist hier keine Datei namens index.html , start.php oder etwas ähnliches vorhanden, daher wird dargestellt, was in `/etc/httpd/conf.d/welcome.conf` konfiguriert wurde
  - Per Default stammt der Inhalt aus: `/usr/share/httpd/noindex/index.html`
- Ubuntu:
  - In der Datei `apache2.conf` findet sich keine Direktive zum DocumentRoot-Verzeichnis. Stattdessen werden diese in `/etc/apache2/sites-enabled` in den jeweiligen Website-Dateien festgelegt (siehe nächster Abschnitt)
  - Die Default-Website wird in `000-default.conf` beschrieben.
  - Hier finden wir eine DocumentRoot-Direktive, die auf das Hauptverzeichnis Verzeichnis zeigt: `/var/www/html`
  - Dort befindet sich eine `index.html`

VirtualHosts erstellen

- *VirtualHosts* geben die Möglichkeit, auf einem Webserver mehrere Webpräsenzen zu hosten
- Meistens werden *namensbasierte* VirtualHosts eingerichtet
- Das Konzept basiert auf HTTP 1.1, hier wurde der Host-Header eingeführt
- Dieses Feld im HTTP-Header enthält den DNS-Namen, der in der URL enthalten ist
- Der Browser fragt konkret nach der URL und darauf kann der Server reagieren und an die entsprechende Webpräsenz weiterleiten
- Damit das funktioniert, muss die Namensauflösung einrichtet sein. Wird im Browser eine IP-Adresse als Ziel eingegeben, funktioniert dies so nicht, da der Webserver die Webpräsenzen nach DNS-Namen unterscheidet

Konfiguration eines VirtualHosts unter Ubuntu:

- Im Verzeichnis `/etc/apache2/sites-available` wird eine neue Datei für Webpräsenz erstellen:
z.B. `001-gulugulu.conf`:
    - `<VirtualHost *:80>` - erstellt eine VirtualHost-Sektion
    - `ServerName www.gulugulu.local` - legt den DNS-Namen fest, der im Hostheader stehen muss
    - `ServerAlias gulugulu.local` - legt weitere Aliase fest, auf denen der Server reagieren soll
    - `DocumentRoot /var/www/gulugulu` - Verzeichnis für Content der Webpräsenz festlegen
- Einstellungen für das DocumentRoot-Verzeichnis:
    - `<Directory /var/www/gulugulu>` - erstellt eine Directory-Sektion für das Verzeichnis
    - `options FollowSymLinks` - Server Symlink im Verzeichnis folgt zum Ursprung
    - `AllowOverrride None` - global gültige Einstellungen können nicht überschrieben werden. Grundsätzlich können über eine Datei namens `.htaccess` in einem Veröffentlichungsverzeichnis noch einmal separate Einstellungen getroffen werden
    - `Require all granted` - alle Benutzer haben Zugriff auf diese Webpräsenz. Hier könnte der Zugriff auch auf bestimmte Benutzer oder IP-Adressen beschränkt werden
    - `</Directory>` - schließt die Sektion und die Einstellungen für das Verzeichnis
    - `</VirtualHost>` - beendet die VirtualHost-Konfiguration
- Im Anschluss muss ein Symlink für diese Konfigurationsdatei im Verzeichnis `sites-enabled` erstellt werden
- Alternativ dient der Befehl `a2ensite 001-gulugulu.conf`, damit wird der VirtualHost ebenfalls aktiviert

Konfiguration eines VirtualHosts unter CentOS:

- Entweder kann eine eigene Datei unter `/etc/httpd/conf.d` erstellt werden, die die Endung `.conf` hat und diese entsprechend befüllt werden, oder aber die VirtualHost-Definition kann direkt in `httpd.conf` geschrieben werden
- Bevorzugt sollten externe Dateien genutzt werden zur besseren Übersicht
- Wenn direkt die IP eingegeben wird, wird auch der spezifische VirtualHost angezeigt. Besser ist es daher, eine weitere VirtualHost-Konfiguration ohne ServerName-Direktive zu erstellen, um hier eine Default-Webspräsenz zu erstellen

Apache-Module

- Die Funktionalität von Apache lässt sich durch die Einbindung von Modulen umfassend erweitern
- Module bringen eine bestimmte Funktionen ein, z.B. SSL-Unterstützung oder URL-Rewrite- oder Proxy-Funktionalität
- Im Verzeichnis `/etc/apache2/mods-available` befinden sich in Debian-Derivaten alle derzeit bekannten und verfügbaren Module
- Für jedes Modul existieren in der Regel zwei Dateien: `.conf` und `.load`
- In der .conf-Datei befindet sich die Konfiguration des Moduls
- In der .load-Datei befindet sich der Name und den Speicherort des Moduls
- Um ein Modul einzubinden, kann entweder ein manueller Symlink in `mods-enabled` erstellt werden, oder es kann der Befehl `a2enmod` verwendet werden
- Durch `a2enmod` werden Abhängigkeiten automatisch geprüft und ggf. Tipps zur Konfiguration ausgegeben
- Der Befehl `apachectl -t -D DUMP_MODULES` zeigt alle aktiven Module

PHP-Modul einbinden

- Eine der wichtigsten serverseitige Programmiersprachen ist PHP
- Um PHP in Apache zu integrieren, muss das PHP-Modul eingebunden werden
- Suche nach PHP-Modul für Apache: `apt search libapache2-mod-php`
- Installation des PHP-Moduls für Apache: `apt install libapache2-mod_php7.2`
- Abhängigkeiten werden aufgelöst und Modul wird in Apache integriert (PHP-Version muss ggf. angepasst werden)
- Installation von PHP unter CentOS inkl. Apache-Modul: `yum install php`

MySQL für LAMP bereitstellen

- Heutige Webanwendungen sind dynamisch und erhalten ihre Daten in fast allen Fällen aus einer Datenbank

- Eine der gängigsten Datenbanken unter Linux ist MySQL bzw. deren freier Klon MariaDB
- MySQL und MariaDB sind funktional fast identisch
- Für eine voll funktionsfähige Plattform für Webanwendungen werden folgende Komponenten benötigt:
    - ein Betriebssystem – typischerweise Linux
    - ein Webserver in Form von Apache2
    - eine Datenbank in Form von MySQL bzw. MariaDB
    - eine serverseitige Skriptsprache für die Erstellung dynamischer Webinhalte – PHP
- Dieses Konzept nennt sich LAMP – von Linux, Apache, MySQL und PHP
- Damit eine in PHP geschriebene Webanwendung die Daten aus der Datenbank nutzen kann, muss eine Schnittstelle zwischen PHP und MySQL vorhanden sein
- Dazu muss das Paket `php-mysql` installiert sein. Es enthält die PHP-Funktionen, die die Kommunikation mit MySQL implementieren
- Auch der MySQL-Server selbst muss über das Paket `mysql-server` installiert sein. Alle notwendigen Abhängigkeiten werden wieder automatisch aufgelöst
- Es gibt kein Apache-MySQL-Modul. Nur PHP selbst kommuniziert mit der Datenbank, nicht der Webserver
- Prüfung, ob PHP also Apache-Modul funktionsfähig ist:
    - Datei `phpinfo.php` unter `/var/www/html` erstellen mit folgendem Inhalt:

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

    - Die Datei kann im Browser über folgende Adresse aufgerufen werden: `http://localhost/phpinfo.php`
    - PHP-Konfiguration im Detail wird angezeigt
- Installation von MySQL und PHP-MySQL-Funktionen:
  `apt install php7.2-mysql mysql-server`
- Es gibt noch zahlreiche MySQL-Zusatzfunktionen, wie z.B. *PHP-MyAdmin* für die webbasierte Verwaltung von MySQL usw.
- Verwaltungsoberfläche von mysql aufrufen: `mysql`
- Befehle in der MySQL-Shell werden immer mit Semikolon abgeschlossen
- Vorhandene Datenbanken anzeigen: `show databases;`
- MySQL ist die Verwaltungsdatenbank des Systems, Wechsel in diesen Kontext: `use mysql;`
- Der Befehl `show tables;` zeigt verfügbare Tabellen
- Benutzer, die von MySQL verwaltet werden, stehen in der Tabelle `user`

- Passwort des Users „root“ ändern:
```
update user set authentication_string=PASSWORD('Pa$$w0rd') where user='root';
update user set plugin='mysql_native_password' where user='root';
```
- Privilegien zurücksetzen mit dem Befehl: `flush privileges;`
- Der Befehl `mysql -p` aktiviert die Passwortabfrage
- Es ist empfehlenswert, für jede Datenbank von MySQL explizit User anzulegen, die ganz bestimmte Zugriffsprivilegien haben, nicht generell den User root nutzen, das ist ein großes Sicherheitsrisiko