Projektdokumentation Homeserver

Thilo Wendt

25. Oktober 2019

Zusammenfassung

Inhaltsverzeichnis

1 Einleitung

Träumt nicht jeder an Informationstechnik interessierte Mensch von der allumfassenden sicheren und unbegrenzten Cloud-Lösung für einen intuitiven Workflow über alle Geräte und Plattformen hinweg? Der Autor möchte sich diesen Traum erfüllen und seinen eigenen Webserver betreiben, um dort verschiedene Dienste anbieten zu können. Aber Halt! Ist es nicht viel einfacher, diese Lösung an extern zu vergeben und die Dienste von Apple, Google und wie sie nicht alle heißen zu nutzen? Einfacher in jedem Fall, doch möchte man seine Daten nicht mit einem internationalen Konzern teilen, so führt kein Weg an der selbst gebauten Lösung vorbei. Des Weiteren ist es mit den heute verfügbaren Open-Source-Werkzeugen einfach geworden, einen eigenen Webserver zu betreiben und diesen universell zu nutzen. Beispielsweise ist auch die Gestaltung eines Webauftritts mit WordPress einfach realisierbar.

Im Folgenden wird zunächst das Problem näher erläutert und die möglichen bereits vorhandenen Lösungen mit der geplanten selbst betriebenen Lösung verglichen. Darauf aufbauend wird eine grobe Systemarchitektur skizziert, sowie das Projekt in Arbeitspakete unterteilt.

2 Motivation

Spätestens seitdem wir angefangen haben, täglich mehrere Computer verschiedener Größe vom Handy bis zur Full-Size-Workstation zu nutzen, haben wir uns bezüglich des Austausches von Informationen und Dateien einige Probleme eingehandelt: Ohne die Nutzung einer Cloud-Lösung ist es schwierig, den automatisierten Austausch zwischen den Geräten zu realisieren. Das Ergebnis ist oft doppelte und inkonsistente Datenhaltung auf unterschiedlichen Medien (USB-Stick, Handy, Laptop etc.). Die grundlegenden Probleme, zu denen im vorliegenden Projekt Lösungen erarbeitet werden sollen, lassen sich auf die folgenden drei Stichpunkte reduzieren.

1. Konsistenz: Wie kann ein einheitlicher Versionsstand zwischen den Geräten hergestellt werden?
2. Verfügbarkeit: Wie kann jedem Gerät zu jedem Zeitpunkt die vollständige Datenbasis zur Verfügung gestellt werden?
3. Vertraulichkeit: Es ist nicht gewünscht, persönliche Daten mit multinationalen Konzernen zu teilen. Wie kann dieses Ziel erreicht werden?

Die Lösung für Problem 1 und 2 scheint wie bereits angedeutet einfach: Die Nutzung einer Cloud-Lösung als verbindende Instanz zwischen allen Geräten. Hierzu gibt es bereits fertige Lösungen wie z.B. Google Drive, iCloud, Dropbox, OneDrive und viele mehr. All diese Fertiglösungen haben jedoch den Nachteil, dass der gesamte persönliche Datenbestand auf einen externen (amerikanischen) Server ausgelagert wird und die dahinter stehenden Unternehmen damit machen können, was sie möchten. Ist dies für den Nutzer akzeptabel steht der Nutzung einer solchen Cloud-Lösung nichts mehr im Wege. Ist man jedoch daran interessiert, selbst über seine Daten zu verfügen, muss etwas mehr Aufwand getrieben werden.

Auch für Home-Clouds gibt es fertige Lösungen. Zu nennen ist hierbei die Serverhardware von QNAP und Synology, welche eine Plug-and-Play-Lösung darstellen. Gegen die Nutzung einer solchen Plattform sprechen jedoch folgende Gründe:

- Preis-Leistungs-Verhältnis: Ein Synology Network-Attached-Storage-Server (NAS) mit einem Dual-Core Prozessor von Intel und 2GB RAM kostet ca. 450€. Für den

selben Preis ist es möglich einen Server aus gebrauchten Teilen mit einem Hexa-Core-Xeon Prozessor und 64GB RAM zusammenzustellen.

- Unflexibel: Die Lösungen von QNAP und Synology sind *ausschließlich* als NAS konzipiert. Ein selbst aufgesetzter Server kann z.B. auch als Plattform für ein GitLab ein WordPress-Blog oder Ähnliches dienen.
- Begrenzte Erweiterbarkeit: Die Baugröße eines typischen Plug-and-Play-NAS erlaubt keine größeren Upgrades bezüglich Prozessor, RAM und Massenspeicher.

Die Motivation, eine eigene Plattform aufzusetzen, ist damit begründet. Im folgenden Kapitel wird kurz die Hard- und Software-Architektur beschrieben.

3 Projektdurchführung

3.1 Hardware

Die Serverhardware umfasst folgende Komponenten:

- Fujitsu D3128-B25 Mainboard mit Intel C602 Server Chipsatz
- 8 x 8 GB DDR3-RDIMM ECC Arbeitsspeicher
- Intel XEON E5-2560V2 Octa-Core Server-CPU mit Hyperthreading
- Effizientes 450 W Modell Straight Power 11 von be quiet!
- Phanteks Enthoo Pro Tower PC-Gehäuse
- WD-Blue 250 GB SSD Festplatte
- ARCTIC Freezer 12 CO CPU-Kühler
- 4 x Seagate ST4000VN008 IronWolf 4 TB HDD Festplatte

All diesen Komponenten ist gemeinsam, dass sie auf Server-Anforderungen wie z.B. Dauerbetrieb optimiert sind. Des Weiteren sind viele der Komponenten gebraucht sehr günstig verfügbar.

3.2 Verwendete Basis-Software

3.2.1 Betriebssystem

Bevor ein sinnvolles Arbeiten auf dem Server möglich ist, steht die Entscheidung für ein bestimmtes Betriebssystem und dessen Installation an. Der Autor hat sich an dieser Stelle für den Einsatz eines UNIX-basierten Betriebssystems entschieden, da für solche Systeme die benötigte Software frei verfügbar ist. Als konkrete Distribution wurde Ubuntu Server 16.04 ausgewählt.

Die Installation des Betriebssystems gestaltet sich bei dem gewählten Mainboard etwas aufwändiger, weil dieses keine Integrierte Grafikschnittstelle bietet. An dieser Stelle ergeben sich zwei Möglichkeiten: Entweder es wird zeitweise eine Grafikkarte verwendet, welche über eine PCI-Express Schnittstelle angeschlossen werden kann, oder es wird eine *vollautomatische* Installation des OS durchgeführt. Im vorliegenden Projekt wurde die letztere Variante durchgeführt. Hierfür sind folgende Schritte notwendig:

1. **Erstellung einer Preseed-Datei:** Während der normalen interaktiven Installation werden dem Benutzer verschieden Fragen zur Konfiguration des Betriebssystems gestellt. Diese Fragen können *vor der Installation* in einer Datei beantwortet werden. Canonical stellt eine recht umfangreiche Dokumentation bereit, wie eine Preseed-Datei zu erstellen ist [3]. Die in diesem Projekt verwendete Preseeding-Datei stammt aus [1] und wurde an die Bedürfnisse angepasst.

2. **Bereitstellung der Datei:** Auch hier gibt es mehrere Möglichkeiten: Die Datei kann über einen FTP Server, auf dem Installationsdatenträger oder im initrd bereit gestellt werden. Die Bereitstellung über FTP bietet sich bei der Installation über PXE an. In diesem Fall erschien die einfachste Möglichkeit, die Datei auf dem Installationsmedium bereit zu stellen. Hierfür wird z.B. mit dem Programm **Cubic** ein angepasstes ISO-Image erstellt. Die Datei muss dann nur an eine beliebige Stelle in dem Image abgelegt werden, wobei sich der Ordner ,,preseed" anbietet.

3. **Bekanntmachung der Datei für den Bootloader:** Der Bootloader muss noch erfahren, dass er die Preseed-Datei nutzen soll, anstatt den Benutzer nach den Optionen zu fragen. Die folgenden Dateien stellen den Inhalt des Menüs dar, welches die Optionen beim starten der Installation entgegen nimmt. Der Option ,,Install Ubuntu Server" wird der Pfad der Preseed-Datei mit der Option `file=/cdrom/path/to/file` hinzu gefügt. Je nach BIOS-Version müssen unterschiedliche Dateien editiert werden.
   - Legacy: Die Datei `/isolinux/txt.cfg` muss editiert werden.
   - UEFI: die Datei `/boot/grub/grub.cfg` muss editiert werden.

4. **Weitere Optionen:** Es muss eine Standard-Option mit einem Timeout definiert werden, damit die Installation automatisch Startet. In der Datei `/isolinux/txt.cfg` geschieht das über den Eintrag `default <label>` und in der Datei `/boot/grub/grub.cfg` so wie in [2] beschrieben. Des Weiteren müssen noch die Optionen für `locale` z.B. `en_us` und `keyboard-configuration/layoutcode` z.B. mit `us` vorbelegt werden, da diese abgefragt werden, *bevor* der Bootloader die Preseed-Datei liest, wo diese Optionen ggf. auch spezifiziert wurden.

Es empfiehlt sich, die Installation vorher einmal in einer virtuellen Maschine zu testen, um festzustellen, ob diese dann auch vollautomatisiert durch läuft. Grundsätzlich sollte die Preseed-Methode auch für Ubuntu 18.04 funktionieren [3], jedoch hatte der Autor massive Probleme mit deren Ausführung und es wurde somit auf Ubuntu 16.04 zurück gegriffen. Ist es unabdingbar eine neuere Distribution zu nutzen, kann nach der Installation unkompliziert auf die neue Version upgegraded werden.

3.2.2 Containervirtualisierung mit Docker

Ist die Installation des Betriebssystems überstanden, eröffnen sich nun die Gestaltungsmöglichkeiten des eigentlichen Webservers. Dem Autor sind zwei grundsätzliche Herangehensweisen bekannt:

„Harte" Installation der Komponenten auf der Maschine Es ist möglich, die gewünschte Software (z.B. einen Webserver, nextcloud, WordPress usw.) und die benötigten Abhängigkeiten (z.B. eine PHP-Laufzeitumgebung) einfach auf der Maschine zu installieren wie jedes andere Programm auch. Dies allerdings unschön im Bezug auf Flexibilität und Wartbarkeit: Nach einiger Zeit ist vergessen, welche Komponenten in welcher Version installiert wurden. Auch das Upgrade auf neue Versionen ist mit einem gewissen Risiko verbunden, da nicht mit garantiert werden kann, ob das System nach dem Upgrade noch so läuft wie es soll.

Aufteilung der Komponenten in Container Die bessere Lösung wäre eine standardisierte Laufzeit- und Entwicklungsumgebung, die sowohl auf dem Testsystem als auch auf dem Produktivsystem identisch ist. Genau diese Funktionalität stellt Docker bereit: Eine Funktionalität wird in einem *Container* gekapselt. Ein Container ist vergleichbar mit einer virtuellen Maschine, die jedoch nativ im Kernel des Host-Systems läuft und dadurch wesentlich ressourcensparender ist als eine richtige VM. Alle benötigten Abhängigkeiten werden zur Laufzeit geladen und sind auch nur dann auf dem System vorhanden. Sobald der Container nicht mehr benötigt wird, kann dieser mit allen Abhängigkeiten mit einem Befehl vollständig entfernt werden.

Des Weiteren kann mit Docker risikofrei ein Versionsupgrade durchgeführt werden: Die neue Version einer Software wird lokal ausgiebig getestet und dann auf das Produktivsystem geladen. Durch einen Neustart der Container ist das Versionsupgrade vollzogen und es gibt nicht mal einen Moment Downtime.

Abgesehen von den Linux-Standard-Werkzeugen und einem Editor (der Autor nutzt GNU (x)Emacs) läuft sämtliche Software in Docker-Containern. Die Server-Software-Architektur ist in Kapitel 3.3 beschrieben.

3.3 Softwarearchitektur

Literatur

[1] *Beschreibung der automatischen Installation von theurbanpenguin.*
URL: `https://www.theurbanpenguin.com/auto-installing-ubuntu-16-04/` (besucht am 25. 10. 2019).

[2] *GRUB2 Konfiguration - ubuntuusers.de.*
URL: `https://wiki.ubuntuusers.de/GRUB_2/Konfiguration/` (besucht am 25. 10. 2019).

[3] *Ubuntu Manual - Installation mittels Preseeding.*
URL: `https://help.ubuntu.com/lts/installation-guide/amd64/apb.html` (besucht am 25. 10. 2019).