Telerec't

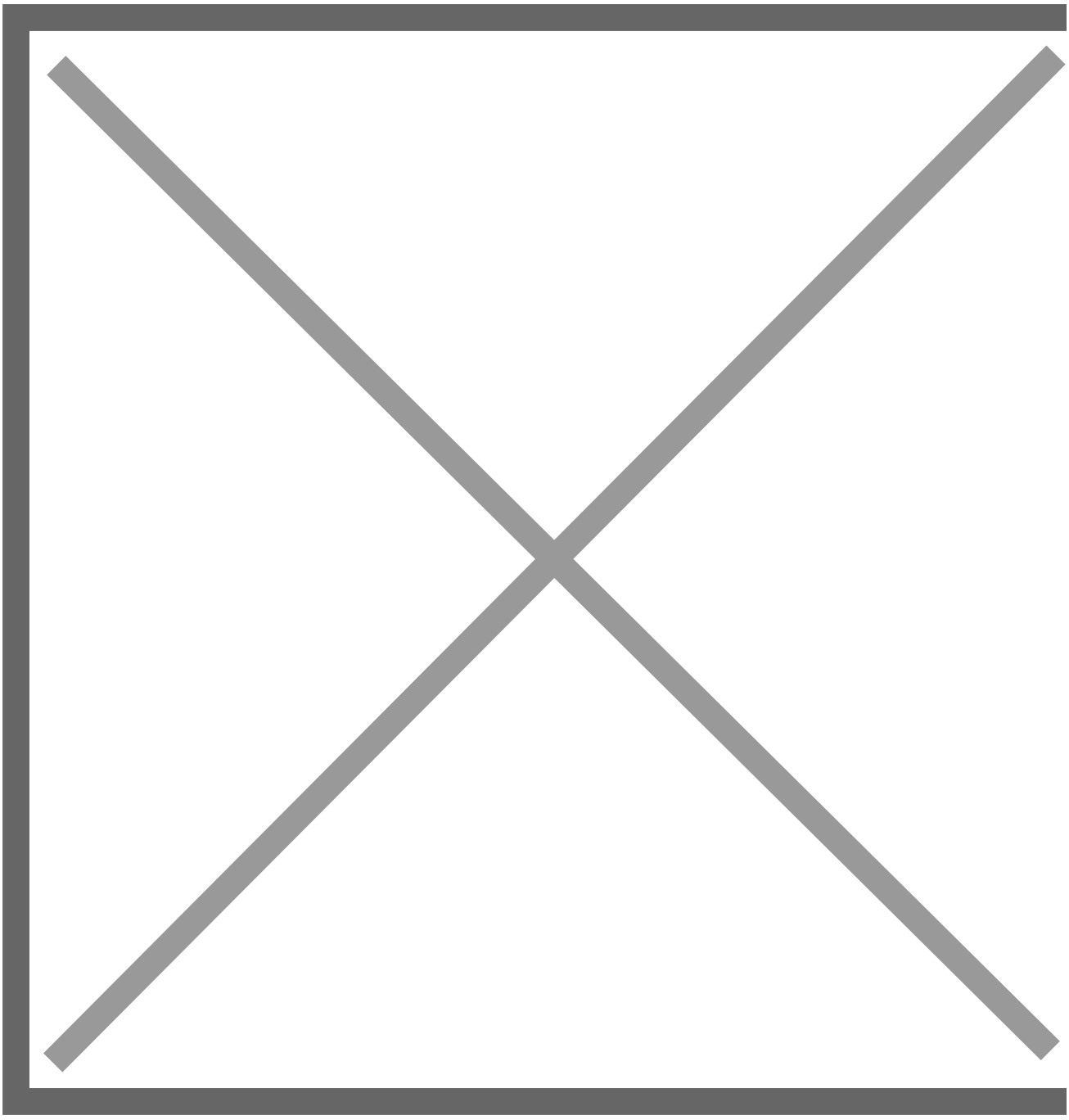


Bild: Thorsten Hübner

Server im Rechenzentrum oder daheim mit Ansible automatisieren

Ansible ist die Automatisierung der Admin-Tätigkeit: Installieren, Kopieren, Patchen – Ansible setzt Ihre Befehle reproduzierbar um, auf so vielen Servern, wie Sie wollen. Gleichzeitig sind die Anweisungen eine vollständige Dokumentation. Als Beispiel bauen wir einen Server-Baukasten namens „Telerec't".

Von Klaus Greff und Pina Merkert

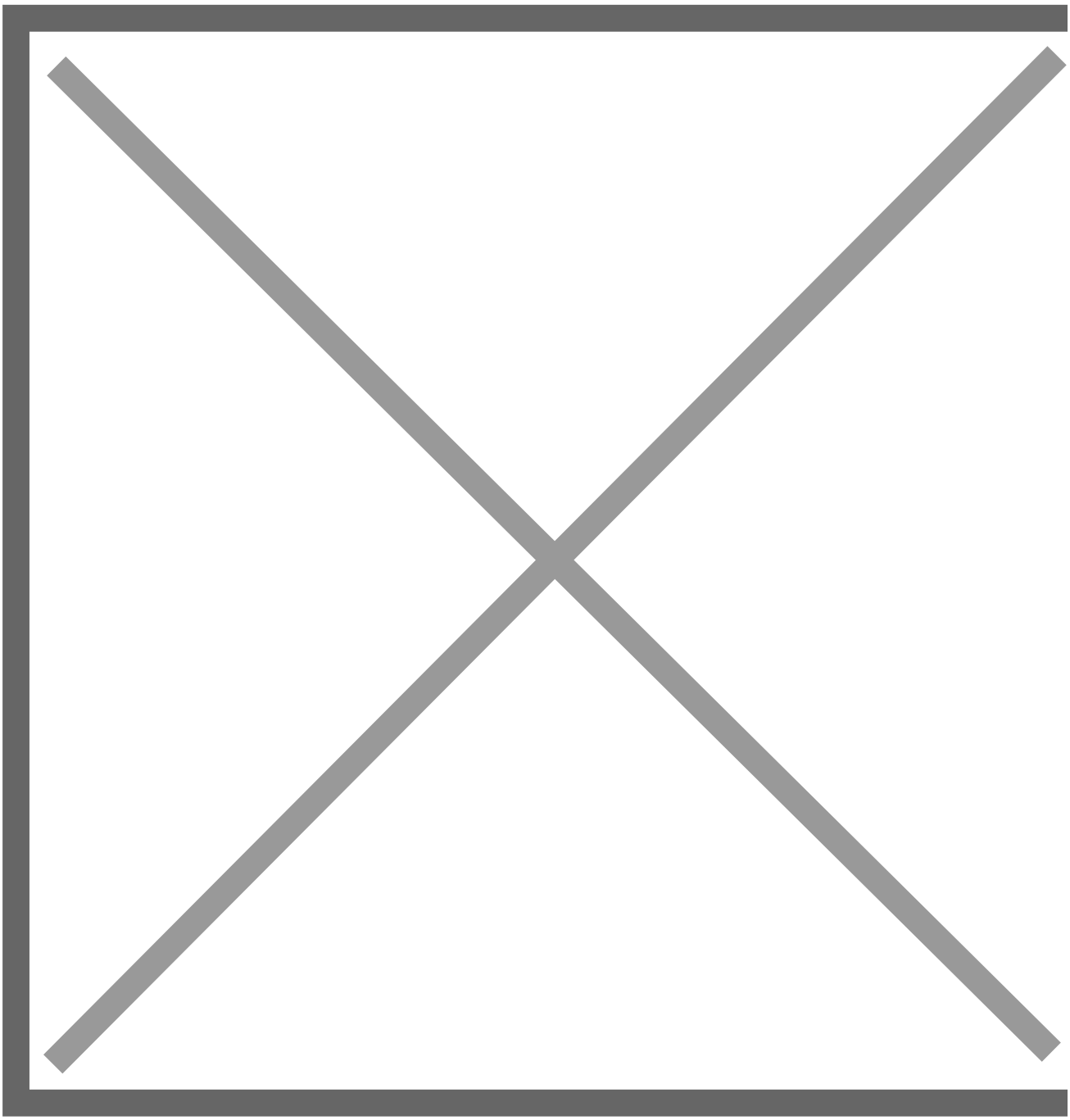
kompakt

- Ansible richtet vollautomatisch Server ein. Was das Programm tun soll, steht in sogenannten Roles und Playbooks.
- Infrastructure-as-Code: Die Roles und Playbooks dokumentieren alle Arbeitsschritte lückenlos und Sie können Programmiererwerkzeuge wie Ihre Lieblings-IDE und Git für die Administration nutzen.
- Als Beispiel haben wir „Telerec't", ein Server-Setup im Baukastensystem, zusammengestellt. Wir werden es in folgenden Artikeln erweitern.

Login per SSH, Docker installieren, immer die gleichen Container hochfahren. Die Arbeit eines Admins kann eintönig sein. Nur nicht vertippen! Wenn Sie da keine Lust drauf haben, sind Sie nicht allein.

Da man ohnehin Computer benutzt, um Computer zu administrieren, lässt sich das alles automatisieren. Eine beliebte Software dafür ist Ansible. Stellen Sie sich das Programm wie einen etwas unselbstständigen Admin-Kollegen im Homeoffice vor. Weil der ein Roboter ist, können sie nicht normal mit ihm reden, sondern geben ihm stattdessen sogenannte „Playbooks". Das sind To-do-Listen, die er stets fehlerfrei und in Rekordzeit abarbeitet.

Im Grunde verschiebt Ansible Know-how vom Notizzettel des Admins in ausführbare Playbooks. Der Fachbegriff dafür lautet „Infrastructure-as-Code". Playbooks lassen sich im Unterschied zu einer Zettelwirtschaft mit Git verwalten und mit einem vertrauten Texteditor oder einer verlässlichen Entwicklungsumgebung für Programmierer editieren.



Playbooks enthalten eine geordnete Liste an Teilaufgaben („Tasks“), die Robo-Kollege Ansible abzuarbeiten hat. Für eine bessere Übersicht können Sie zusammengehörende Aufgaben in „Roles“ organisieren. Eine Menge Fehler beim Administrieren von Servern entstehen durch Copy & Paste und danach vergessene Anpassungen. Damit Sie gar nichts anpassen müssen, unterstützt Ansible Variablen und Templates.

Letztendlich passiert aber nichts anderes als ein Log-in auf dem Server mit der Secure-Shell SSH [1]. Auf dem müssen Sie also keine Client-Software installieren. Ansible führt auf dem Server nur stinknormale Shell-Skripte aus, die Sie nicht mal selbst schreiben müssen. Bei all der Admin-Magie, die wir Ihnen im Folgenden vorführen, sollten Sie nicht vergessen, dass Sie alles auch per Hand ausführen könnten. Wenn Sie und kein anderer Admin eine Idee haben, wie es per Hand geht, werden Sie es Ansible auch nicht beibringen können.

Als Beispiel automatisieren wir die Administration eines eigenen Heim- und eines Root-Servers mit Debian oder Ubuntu und unserem Baukasten „Telerec't“ (von „tele erect“ – „aus der Ferne aufgerichtet“). Der Heimserver soll später mal Smart-Home-Dienste wie Mosquitto (MQTT-Broker) und NodeRed (No-Code Automatisierungsregeln) ausführen, der angemietete Root-Server im Rechenzentrum Wordpress (Blog-Software) und Nextcloud (Dateisynchronisation) beherbergen. SSH-Schlüssel hinterlegen sowie Basis-Pakete und Docker installieren ist für beide Server gleich. Im nächsten Artikel zu Ansible stellen wir vier Dienste vor, die in unseren Augen auf jedem Server laufen sollten. Erst danach folgen in weiteren c't-Ausgaben die eigentlichen Bausteine in Form zusätzlicher Ansible-Roles als eigene Artikel, sodass Sie ganz individuell auswählen können, was Sie auf Ihrem Server installieren wollen.

Die Bausteine setzen wir als Git-Submodules um. Submodules sind ein eher selten genutztes Feature der Versionsverwaltung Git [2, 3]. Jedes Submodule ist ein simpler Unterordner, aber gleichzeitig auch ein eigenständiges Git-Repository. Das binden Sie so in ein Basis-Repository ein, dass dieses protokolliert, auf welchem Stand das Gesamtkonstrukt war. Zum Baukasten wird das, weil Sie die Submodules unverändert in Ihr Basis-Repository einbinden können. Sie pflegen nur dieses eine Repository, denn das bildet Ihr Setup vollständig ab. Updates in den Submodules spielen Sie aber trotzdem mit nur einem Befehl ein.

Infrastructure-as-Code

In diesem Artikel zeigen wir, wie Sie Ansible installieren und eigene Playbooks schreiben. Im ersten Playbook dieses Artikels übertragen wir kryptografische Schlüssel, damit Sie und Ansible sich danach mit der Secure Shell SSH [1] ohne Passwort am Server anmelden können. Außerdem werden wir Docker und `docker-compose` installieren. Docker ist meist die erste Wahl, um containerisierte Anwendungen auszuführen. In Container verpackte Anwendungen kommen sich nicht gegenseitig in die Quere. Fertige Images lädt Docker aus dem Docker Hub herunter. Falls Ihnen eine große Auswahl fertiger Images nicht wichtig ist, können sie Telerec't stattdessen auch mit der Docker-Alternative Podman nachbauen. Wir werden für Telerec't aber Docker benutzen.

Ein Wort der Warnung: Telerec't ist entstanden, weil wir ein Setup für unsere eigenen Server gebraucht haben. Wir veröffentlichen das Basis-Repository und mehrere Submodules, weil diese ein hervorragendes Beispiel sind, wie Sie Ansible für Ihren Server nutzen können. Denken Sie dabei aber bitte mit, prüfen Sie, was wir machen, und tragen Sie gern Verbesserungsvorschläge als Pull-Requests an uns heran. Telerec't ist nämlich mit all seinen Bausteinen Open Source und unsere Konfiguration haben wir unter der Lizenz GPLv3 veröffentlicht. Wir leisten keinen professionellen Support und geben keine Garantie, die Repositories in Zukunft zu pflegen. Sehen Sie das Projekt als Freie Software ohne Maintainer. Forken Sie die Repositories, falls Sie Maintainer werden wollen.

Hardwareanforderungen

Mit Telerec't haben wir sowohl einen Heimserver als auch einen virtualisierten Server im Rechenzentrum aufgesetzt. Wir empfehlen mindestens 8 GByte RAM und zumindest zwei CPU-

Kerne. Mit einem Raspi Zero würden Sie sich keinen Gefallen tun, der große Raspi 5 (8 GByte) mit einer NVMe-SSD käme als kleiner Heimserver aber infrage. Zwei offizielle M.2-Adapter für den Raspi 5 sind für Frühjahr 2024 angekündigt. Momentan würden wir einen Heimserver mit kleinem Mainboard und einem sparsamen x68-Prozessor empfehlen (beispielsweise ein Billig-Barebone aus [5]).

In Rechenzentren ist die kleine Hardware ein virtualisierter Anteil an einem größeren Server. Angebote ab 15 Euro pro Monat sind meist stark genug für ein Dutzend Webserver-Container, sofern nicht viele Anfragen kommen. Hier steigen die Kosten, falls Sie viel Speicherplatz für eine Nextcloud brauchen. Wir mieten beispielsweise einen Server mit 16 virtuellen Kernen, 24 GByte RAM und 500 GByte Speicherplatz für knapp 18 Euro im Monat. Der reicht für einen Mailserver parallel mit vier Wordpress-Containern, einer Nextcloud, Wekan, Vaultwarden und drei eigenen Django-Projekten.

Ansible installieren

Für einen stressfreien Start bestellen Sie einen Root-Server mit einem vorinstallierten Debian oder Ubuntu. Falls Sie einen Heimserver aufsetzen, installieren Sie eines der Systeme wie gewohnt und aktivieren Sie das Log-in per SSH. Bei Raspis empfehlen wir ganz langweilig das Debian-basierte Raspberry Pi OS. Telerec’t funktioniert auf allen Linux-Distributionen mit dem Paketmanager `apt`. Der händische Teil der Installation endet, sobald Sie `ssh`-Zugang haben. Den sollten Sie testen, was Ihren Rechner nebenbei auf dem Server auch in ~/.ssh/known_hosts einträgt (Ansible trägt sich dort nicht automatisch ein).

Danach wechseln Sie zu Ihrem Admin-PC oder -Notebook mit Linux, macOS oder Windows mit dem WSL (Windows Subsystem for Linux). Dort installieren Sie Ansible – es läuft auch nur dort. Der Server bekommt nämlich gar nicht mit, dass Sie sich von Ansible helfen lassen. Sie können auch von mehreren verschiedenen Rechnern aus den Server administrieren. Dafür klonen Sie das Repository mit dem Infrastruktur-Code und geben jedem der Rechner Zugriff auf den Server.

Ansible ist in Python geschrieben, weshalb Sie es auf allen Desktopbetriebssystemen mit Python-Paketmanagern installieren können. Wir empfehlen diese Art der Installation zusammen mit `pipenv`, weil das automatisch ein virtuelles Environment mitverwaltet, sodass unterschiedliche Python-Projekte einander nicht in die Quere kommen können.

Alternativ: In den meisten Linux-Distributionen sind die Ansible-Pakete aus den Paketquellen aktuell genug. Wenn Sie Ansible auf diesem Weg installieren, läuft es ohne virtuelle Umgebung und Sie können `pipenv run` zu Beginn der folgenden Befehle weglassen. Zusätzlich brauchen Sie dann noch `python-passlib`, sparen sich später aber `pipenv install`. Wir haben den Weg über `pipenv` gewählt, weil Sie dann auf allen Betriebssystemen und Distributionen die aktuelle Version installieren.

Am einfachsten administrieren Sie einen Linux-Server von einem Linux-Rechner aus. Installieren Sie die Pakete für `git`, `sshpass` (fürs Einloggen mit Passwort) und `pipenv` beziehungsweise Ansible (falls Sie die Paketverwaltung bevorzugen). Unter Ubuntu geht das beispielsweise mit dieser Zeile:

```
sudo apt install git sshpass pipenv
```

Unter Windows raten wir zum WSL, das Sie ruckzuck installieren, indem Sie in ein cmd- oder PowerShell-Fenster mit Administratorrechten `wsl --install -d ubuntu` eintippen. Danach öffnen Sie Ubuntu über das Startmenü, was die Ersteinrichtung anstößt, die einige Sekunden dauert und während der Sie einen Benutzeraccount anlegen. In der Ubuntu-Konsole haben Sie danach den Ubuntu-Paketmanager `apt` zur Verfügung, um die restlichen Abhängigkeiten zu installieren:

```
sudo apt install python3 git pipenv sshpass
```

Mac-Nutzern empfehlen wir Homebrew, das Sie mit dem folgenden Befehl installieren:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Danach installieren Sie `git` und `pipenv` mit folgendem Befehl:

```
brew install git pipenv
```

Alternativ können Sie Ansible auf dem Mac ebenfalls mit `brew` installieren und `pipenv run` dann ebenfalls weglassen.

Die nächsten Schritte funktionieren auf allen Systemen gleich.

Privates Repository anlegen

Unser Beispiel-Setup Telerec’t fußt auf der Idee, dass Sie sich in einem privaten Git-Repository im Baukastensystem eine Konfiguration für Ihren eigenen Server zusammenbauen. Dafür brauchen Sie zunächst ein leeres privates Repository. Privat sollte das sein, damit Sie dort die Passwörter hinterlegen können. Auch bei kostenlosen GitHub-Accounts können Sie private Repositories erstellen. Falls Sie GitHub nicht vertrauen, können Sie auch mit `git init` ein lokales Repository anlegen und erst später mit `git add remote` ein Repository zum Synchronisieren hinzufügen, beispielsweise auf einem selbst gehosteten Gitea.

Um Ihnen die Arbeit an Ihrem individuellen Setup zu erleichtern, haben wir ein Beispiel-Repository erstellt, bei dem Sie sich bedienen können (siehe ct.de/yu9e). Statt mit Copy & Paste befüllen Sie Ihr Repository daraus schneller mit `git`:

```
git remote add ct-example https://github.com/ct-Open-Source/telerec-t-base
git pull ct-example main
git remote remove ct-example
```

Danach haben Sie eine Basiskonfiguration für Ansible bestehend aus den Dateien ansible.cfg und hosts sowie den Verzeichnissen group_vars/ und roles/. In letzterem befinden sich bereits Git-Submodule, die aber noch nicht geklont wurden. Das holen Sie mit folgendem Befehl nach:

```
git submodule update --init \
  --recursive
```

Weil Sie den länglichen Befehl für Updates der Submodule immer mal wieder brauchen, lohnt es sich, ein Alias für die Kurzform `git supdate` anzulegen:

```
git config alias.supdate 'submodule update --remote --merge'
```

Nun haben Sie zwar `pipenv` und die nötige Verzeichnisstruktur, aber Ansible und Passlib sind noch nicht installiert. Die stehen nämlich nur als Voraussetzung in der Datei Pipfile im Repository. Die Voraussetzungen installieren Sie aus dem Basis-Ordner des Repositorys mit:

```
pipenv install
```

Setup anpassen

Das neu angelegte und mit Beispieldaten befüllte Repository müssen Sie nun an den eigenen Server anpassen. Tragen Sie dafür zunächst in der zweiten Zeile der Datei `hosts` die statische IP-Adresse Ihres Servers oder bei einem Heimserver optional auch dessen Hostname ein. Passend zu diesem Rechner können Sie nun Variablen in host_vars/<ip_oder_hostname>.yml anlegen. Hat Ihr Server beispielsweise die IP-Adresse 192.168.7.140, lautet der Dateiname also host_vars/192.168.7.140.yml.

> Der Screenshot aus unserer IDE zeigt die Struktur des Setups: Die Submodule sind in den Ordr
> Der Screenshot aus unserer IDE zeigt die Struktur des Setups: Die Submodule sind in den Ordner roles/ eingebunden, Playbooks liegen im Wurzelverzeichnis. Die README.md enthält Befehle zum Herauskopieren.

Wie die meisten Konfigurationsdateien für Ansible ist auch diese Datei im YAML-Format, mit dem Sie Variablen als hierarchische Strukturen definieren können. Das folgende Beispiel müssen Sie für Ihren Server passend anpassen:

```
ansible_user: mariamuster
admin:
  name: mariamuster
  key: "{{ lookup('file', '~/.ssh/id_ed25519.pub') }}"
  email: "maria.muster@gmail.com"
locale: de_DE.UTF-8
timezone: Europe/Berlin
docker_dir: "/docker"
```

In der ersten Zeile steht der lokale Benutzername, der Block darunter definiert einen Account mit Systemverwalterrechten auf dem Server. Die `lookup`-Funktion hinter dem Schlüssel `key` sucht lokal, also auf dem Admin-System, im angegebenen Verzeichnis ~/.ssh/ nach dem öffentlichen Schlüssel für passwortloses Login mit `ssh`. Den Pfad und Dateinamen müssen Sie wahrscheinlich an die Dateinamen und Pfade an Ihrem Rechner anpassen. Moderne Schlüssel heißen oft

id_ed25519.pub, ältere oft id_rsa.pub [4].

Mit der E-Mail-Adresse darunter bauen Sie schon für die Dienste vor, die wir im nächsten Artikel installieren. Wichtig ist hier zunächst nur, dass es sich um eine externe Adresse handelt, die nicht vom eigenen Mailserver auf demselben Server bereitgestellt wird.

In der letzten Zeile können Sie das Verzeichnis umstellen, in dem auf dem Server letztlich die Daten aller Dienste landen.

Struktur erklärt

Ein Ansible-Projekt besteht aus „Tasks“, die zu „Roles“ zusammengefasst sein können. Aus diesen Elementen kann man dann Programme schreiben die im Ansible-Jargon „Playbooks“ heißen. Tasks, die immer gemeinsam ausgeführt werden müssen, sollten zusammen in einer Role stehen. Die Roles sind simple Unterverzeichnisse im roles-Ordner. Playbooks können alternativ auch direkt Tasks enthalten. Solche haben wir für Telerec’t aber nicht gebraucht, weil sie in Roles stets besser aufgehoben waren.

> Im roles/-Ordner gibt es Unterverzeichnisse für jeden Dienst. Was in files/ liegt, kann der copy-
> Im roles/-Ordner gibt es Unterverzeichnisse für jeden Dienst. Was in files/ liegt, kann der copy-Task übertragen, in templates/ liegen Jinja2-Templates für den template-Task bereit. Öffentliche Variablen dürfen in einem Order defaults/ stehen. In tasks/main.yml stehen die Arbeitsanweisungen für Ansible.

Das Herzstück jeder Role ist der tasks-Ordner, in dem Ansible nach einer Datei main.yml sucht. Die kann mit `include_tasks:` auch weitere YAML-Dateien einbinden, automatisch ausgeführt werden die aber nicht. In defaults/main.yml kann man jeweils Role-spezifische Variablen definieren. Der Ordner heißt „defaults“, weil im Playbook definierte Variablen die Variablen aus main.yml überschreiben. Ansible bevorzugt stets spezifisch definierte Variablen gegenüber allgemeinen. In files/ stellt man Dateien bereit, die Tasks unverändert auf den Server kopieren (`copy`-Task), in templates/ landen Dateien in der Template-Sprache Jinja2, die Ansible mit dem `template`-Task ausfüllt und überträgt. Neben Variablen können die Templates auch `if`-Abfragen und Schleifen enthalten.

Mit `ansible-galaxy` lassen sich Roles über eine Art Paketverwaltung teilen. Das haben wir im Playbook initial-setup.yml genutzt und installieren Docker mit einer Role von Raspi-YouTuber Jeff Geerling:

```
pipenv run ansible-galaxy role install geerlingguy.docker
```

Eine so geteilte Role ist im Prinzip nichts anderes als ein Verzeichnis unterhalb von roles/. Für Telerec't haben wir einfach Git-Repositories als Git-Submodules ins roles-Verzeichnis geklont. Wir hätten sie genauso auf Ansible-Galaxy veröffentlichen können. Mit diesen beiden Methoden bauen Sie auch später weitere Bausteine in Ihr Ansible-Setup ein: Sie suchen sich beispielsweise ein passendes Repository bei github.com/ct-Open-Source (siehe ct.de/yu9e) wie 2fauth [5] aus und

klonen es mit dem folgenden Befehl, den Sie im Wurzelverzeichnis des Projekts ausführen:

```
git submodule add https://github.com/ct-Open-Source/telerec-t-2fauth roles/twofauth
```

Beim Basis-Setup sind sechs Roles dabei, die in unseren Augen jeder Server braucht. Das nach roles/system geklonte telerec-t-debian überträgt den SSH-Schlüssel für passwortloses Login, fügt den Benutzer der `sudo`-Gruppe hinzu, installiert mit `apt` Pakete wie `vim`, `wget`, `curl` und `rsync` (welche genau, steht in roles/system/vars/main.yml) und legt das Verzeichnis für Docker-Volumes an. Wir haben dafür /docker/ voreingestellt, Sie können es aber über die Variable in host_vars/ auch ändern. In diesem Verzeichnis landen alle Daten, die Sie regelmäßig sichern sollten.

> Ansible schreibt bunt in die Konsole, welche Tasks geklappt haben. Bricht ein Task mit einem F
> Ansible schreibt bunt in die Konsole, welche Tasks geklappt haben. Bricht ein Task mit einem Fehler ab, werden zuvor ausgeführte Änderungen nicht mehr rückgängig gemacht. Deswegen sollten Tasks einen definierten Zustand herstellen, egal was vorher geklappt hat oder nicht.

Docker findet sich nicht in den voreingestellten Paketquellen. Die Role `geerlingguy.docker` richtet zuerst die Paketquelle mitsamt GPG-Schlüsseln ein und installiert dann Docker und das Compose-Plug-in. Da die Role von Ansible-Galaxy kommt, gibt es für sie keinen Unterordner in roles/. Beide Roles stehen aber im Playbook initial-setup.yml:

```
- hosts: server
  become: true
  roles:
    - system
    - geerlingguy.docker
```

Hinter `hosts:` steht der Name des Servers, den Sie in der Datei „hosts“ angegeben haben. `become: true` verschafft dem Benutzer auf dem Server Root-Rechte und hinter `roles:` folgt die Liste der Verzeichnisse in roles/, die Ansible im Zuge des Playbooks abarbeitet oder die über Ansible-Galaxy geladenen Roles ohne Verzeichnis.

Playbooks ausführen

Ansible will sich standardmäßig ohne Passwort anmelden, was noch nicht geht, wenn der öffentliche SSH-Schlüssel noch nicht hinterlegt ist. Das Playbook initial-setup.yml kopiert den Schlüssel auf den Server; um es auszuführen, müssen Sie Ansible aber anweisen, einmalig nach dem SSH-Passwort und dem `sudo`-Passwort zu fragen:

```
pipenv run ansible-playbook initial-setup.yml -i hosts --ask-pass --ask-become-pass
```

Ansible dokumentiert mit farbigen Ausgaben auf der Kommandozeile die Ausführung aller Tasks. Grün deutet an, dass es nichts tun musste, bei Orange war der Task erfolgreich. Fehlermeldungen markiert Ansible rot.

Der Befehl zeigt, wie mächtig Ansible in der Praxis sein kann. Mit obigem Befehl sind Dutzende Pakete installiert und Schlüssel hinterlegt. Und nichts hält Sie davon ab, zusätzlich auch beliebige Skripte auszuführen, Software zu kompilieren oder Daten umherzuschieben. Alles mit nur einem Befehl, und wenn Sie das wünschen, sogar auf Dutzenden Servern gleichzeitig.

Erfahrungsgemäß werden Sie sich schon nach kurzer Zeit Playbooks für alle Routineaufgaben bei der Server-Administration erstellen. Oft ist die Aufgabe dann mit einem einzelnen Befehl erledigt und Sie können sich wieder interessanteren Tätigkeiten zuwenden. Außerdem dokumentieren Roles und Playbooks exakt, was Sie gemacht haben. So protokollieren Sie Ihre Arbeit nicht nur für Kollegen, sondern auch für sich selbst, und stellen sicher, dass Sie auch Monate später noch nachvollziehen können, welche Schritte Sie genau benutzt haben.

Ähnlich wie das Schreiben von Dokumentation verlangt ein Ansible-Setup aber auch Disziplin. Ändern Sie nicht einfach in einer SSH-Session Konfigurationsdateien auf dem Server. Für Telerec’t müssen sie das eigentlich auch gar nicht. Wenn Sie aber trotzdem einen Hotfix brauchen, denken Sie daran, dass vermutlich eines Ihrer Playbooks den Hotfix zu einem ungünstigen Zeitpunkt überschreiben wird. Fügen Sie der betroffenen Ansible-Role deswegen auch gleich Tasks hinzu, die die gleiche Änderung auch automatisch umsetzen können. Das geht beispielsweise mit Tasks, die mit regulären Ausdrücken Textdateien für die geänderte Konfiguration editieren.

Zugegeben: Wir haben einige Stunden gebraucht, bis wir mit Ansible vertraut genug waren, um die gewohnten Wartungsarbeiten am Server vorzunehmen. Inzwischen hat sich dieser Aufwand aber ausgezahlt: Neue Dienste setzen wir mit weniger Tipparbeit auf und sparen bei jedem etwas Zeit. Durch die gute Dokumentation sind wir deutlich konsequenter beim Einrichten der Dienste, was für uns selbst und auch andere die Übersicht verbessert. Vor allem aber ist das Gefühl verschwunden, dass wir einen früheren Schritt vergessen haben könnten und der neueste Befehl alles kaputtmachen könnte. Unser Setup ist in der Versionsverwaltung gesichert und wir können eine funktionierende alte Konfiguration mit einem Checkout und einem Ansible-Durchlauf ganz schnell wieder rekonstruieren. Administrationsaufgaben reproduzierbar ausführen zu können, gibt ein Gefühl der Sicherheit, das die Einarbeitungszeit schon bald rechtfertigt. Zeitersparnisse, weil Sie Befehle nur genau einmal eintippen müssen, und die Möglichkeit, mit anderen Admins Roles auszutauschen, sind der Zuckerguss auf dem Kuchen.

Im nächsten Artikel zu Telerec’t werden wir uns der Aufgabe widmen, vier essenzielle Docker-Container aufzusetzen und zu starten. *(pmk@ct.de)*

1. Literatur

2. Peter Siering, FAQ: SSH - Secure Shell, c’t 03/2018, S. 158
3. Herbert Braun, Unvergessen, Erste Schritte mit dem Versionskontrollsystem Git und mit GitHub, c’t 5/2014, S. 176
4. Jan Mahn und Merlin Schumacher, Arbeiten mit GitHub, c’t 21/2018, S. 158

5. Niklas Dierking, Schlüsselmeister, Sicher und komfortabel arbeiten mit SSH, c’t 21/2022, S. 172
6. Markus Stubbig, Cloudtresor zum Generieren von Einmalpasswörtern, 2FA-Authentifikator selbst gemacht, c’t 27/2023, S. 146
7. Christof Windeck, Desktop-Duell, So schlägt sich der Raspberry Pi 5 als PC-Ersatz, c’t 29/23, S. 102

Repositories: ct.de/yu9e

Revision #1
Created 2026-09-26 13:20:54 UTC by ralf
Updated 2026-09-26 13:20:54 UTC by ralf