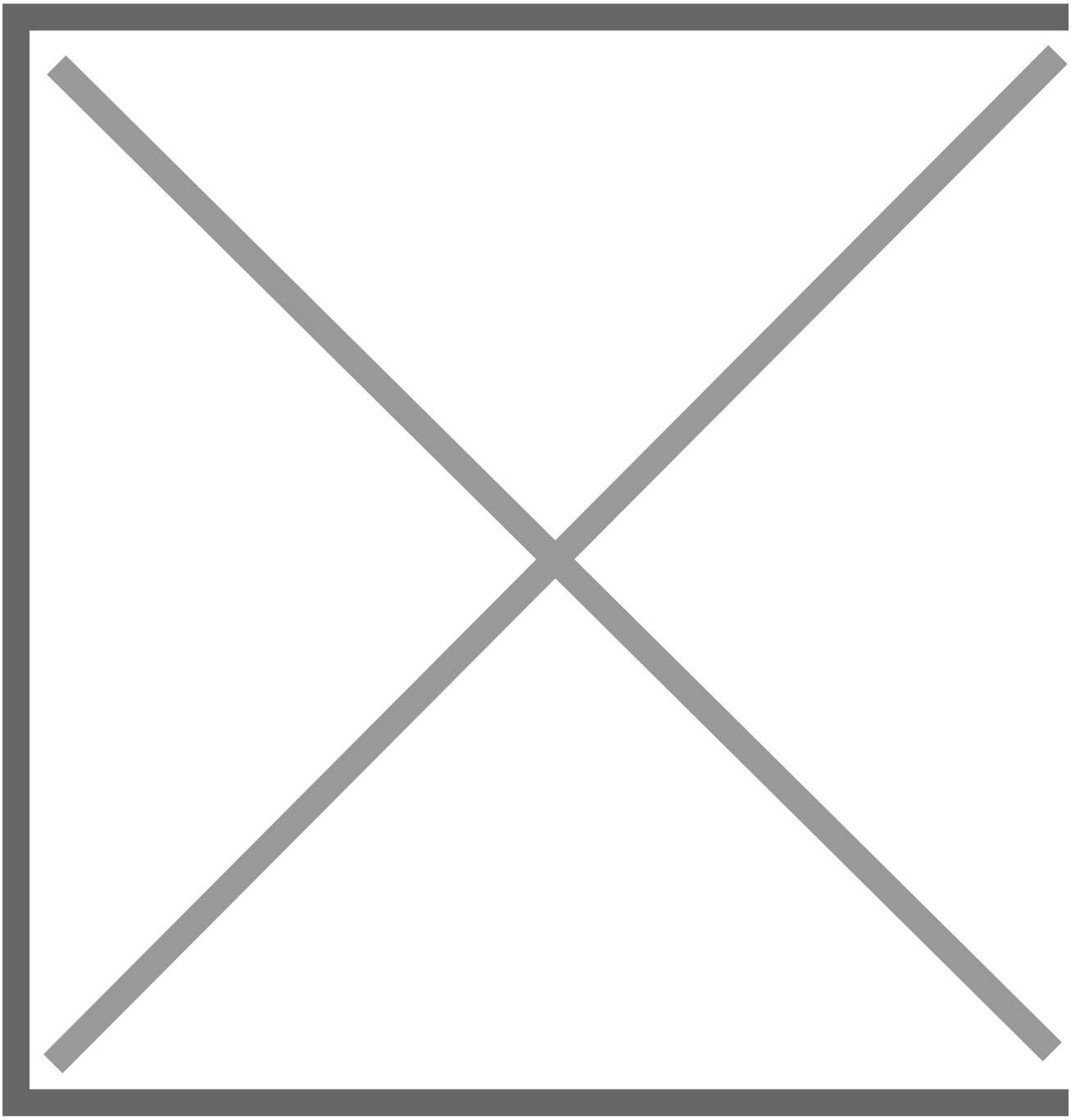
Schaltzentrale für den Server



Weboberfläche für Ansible mit Semaphore

Pakete aktualisieren, Docker installieren, Server neu starten: Das praktische Kommandozeilenwerkzeug Ansible hilft Admins dabei, nervige Aufgaben zu automatisieren und versetzt Serverflotten in einen reproduzierbaren Zustand. Semaphore erweitert Ansible um eine praktische Weboberfläche.

Von Niklas Dierking
c't 3/2024 Seite 154

c't kompakt

- Ansible kümmert sich um Konfigurationsaufgaben und Automatisierung, hat aber eine steile Lernkurve.
- Semaphore stellt Ansible eine Weboberfläche zur Seite, die den Einstieg in Ansible erleichtert und dabei hilft, die eigene Serverflotte im Griff zu behalten.
- Wir zeigen, wie Sie Semaphore installieren, Tasks für Ansible erstellen und zu festgelegten Zeitpunkten ausführen.

Ansible kümmert sich zuverlässig um Konfigurationsaufgaben, indem es To-do-Listen abarbeitet, die im Ansible-Jargon Playbooks heißen. Auf einem frischen Server installieren Admins so in Windeseile wichtige Pakete und versetzen das System in einen klar definierten Zustand. In [1] und [2] lesen Sie eine Einführung in Ansible und wie Sie mit unserem Ansible-Projekt telerec’t eine Reihe von containerisierten Diensten auf einem Heim- oder Mietserver einrichten, die sich bewährt haben.

Üblicherweise läuft Ansible auf dem lokalen System des Administrators (Control-Host). Es kann es sich aber lohnen, Ansible auf ein dediziertes System auszulagern. Beispielsweise, wenn man möchte, dass der Server (Ziel-Host) jede Nacht prüft, ob es Updates gibt und diese einspielt. Als dedizierter Control-Host reicht eine schmale VM oder ein älterer Raspberry Pi in Ihrem Heimnetzwerk.

Das Open-Source-Projekt Semaphore erweitert Ansible um eine Weboberfläche, die als Schaltzentrale für Ihre Serverflotte dient. Für Nutzer, die mit der Kommandozeile weniger vertraut sind, erleichtert Semaphore den Einstieg in die Automatisierung mit Ansible. In diesem Artikel erfahren Sie, wie Sie Semaphore in Betrieb nehmen und damit Aufgaben auf entfernten Systemen ausführen. Ein Grundverständnis für Ansible und SSH ist dafür hilfreich.

Installation

Am einfachsten installieren Sie Semaphore auf einem Ubuntu-Host als Snap-Paket:

```
sudo snap install semaphore
```

Wir haben für unsere Testläufe Ubuntu Server 22.04 LTS genutzt, das wir von einem anderen Rechner im Netz via SSH bedienen. Snap hat den Vorteil, dass Ansible, die Datenbank BoltDB und weitere Abhängigkeiten mit im Snap-Container stecken und nicht zusätzlich installiert werden müssen. Wenn Sie Snap meiden oder eine andere Linux-Distribution als Ubuntu vorziehen, beschreibt die Semaphore-Dokumentation (ct.de/y4au) weitere Installationswege, beispielsweise mittels Docker, oder Sie laden ein Debian- oder RPM-Paket herunter, installieren Semaphore und eine kompatible Datenbank manuell.

Nach der Installation müssen Sie Semaphore stoppen und ein Benutzerkonto für den Administrator anlegen:

```
sudo snap stop semaphore

sudo semaphore user add --admin \
--login cttest \
--name=Testuser \
--email=cttest@example.com \
--password=geheim
```

Ersetzen Sie `cttest` durch einen eigenen Benutzernamen und `cttest@example.com` durch Ihre E-Mail-Adresse. Setzen Sie außerdem ein sicheres Passwort. Damit Ansible seine Arbeit verrichten kann, müssen Sie Zugangsdaten für die Ziel-Hosts in Semaphore hinterlegen. Eine ungeschützte Semaphore-Instanz dient Hackern als Generalschlüssel. Wir raten deswegen dazu, Semaphore nur im lokalen Netzwerk zu nutzen und nicht in das Internet zu hängen.

Starten Sie Semaphore mit `sudo snap start semaphore`, rufen in Ihrem Browser http://semaphore-host:3000 auf und melden sich dann mit den zuvor konfigurierten Zugangsdaten an. Ersetzen Sie semaphore-host durch den Hostnamen oder die IP-Adresse des Servers, der Semaphore ausführt.

Semaphore anfüttern

Als ersten Schritt legen Sie ein Projekt an und benennen es. Projekte dienen in Semaphore dazu, Automatisierungsaufgaben logisch zu trennen, beispielsweise eine Produktions- von einer Testumgebung. Sie können später beliebig viele weitere Projekte hinzufügen. Jetzt begrüßt Sie die Semaphore-Weboberfläche, die mit einigen Informationen gefüttert werden will, bevor Sie Ihr erstes Playbook ausführen.

Statten Sie zuerst dem Key Store einen Besuch ab, den Sie über die Seitenleiste auf der linken Seite des Fensters erreichen. Er verwaltet Zugangsdaten für Ziel-Hosts und Git-Repositories. Ansible führt Aufgaben auf Ziel-Hosts mittels SSH aus.

Dabei sollten Sie die Authentifizierung mittels SSH-Schlüsseln stets Passwörtern vorziehen. Damit das klappt, müssen Sie private SSH-Schlüssel in Semaphore hochladen, die als Gegenstück zu den öffentlichen Schlüsseln auf den Ziel-Hosts dienen. Schützen Sie Ihre Semaphore-Instanz gut. Eine kompromittierte Instanz ist eine Art Generalschlüssel für Angreifer.

Legen Sie einen Eintrag vom Typ „SSH Key“ an, fügen den privaten Schlüssel ein und geben ihm einen Namen. Wie Sie SSH-Schlüsselpaare erstellen und verwalten, lesen Sie in [3]. Die Anmeldung erfolgt standardmäßig mit dem Benutzer root. Wenn Ansible sich als ein anderer Benutzer anmelden soll, müssen Sie bei der Erstellung des Eintrags in Semaphore den korrekten Benutzer angeben.

Für Ziel-Hosts, auf die Sie via Passwort zugreifen, legen Sie einen Eintrag vom Typ „Login with password“ an und hinterlegen dann Nutzername und Passwort. Diese Zugangsdaten können auch als Passwort für sudo dienen, wenn Ansible sich als unprivilegierter Benutzer anmeldet und Aufgaben ausführen soll, die Systemverwalterrechte benötigen, beispielsweise bei der Installation von Paketen.

Legen Sie zuletzt noch einen Eintrag vom Typ „None“ an, dem Sie einen beliebigen Namen geben können. Semaphore verlangt, dass Sie jedem Git-Repository, aus dem es die Playbooks herunterlädt, ein Eintrag im Key Store zuordnen. Das gilt auch für öffentliche Repositories, wie unser Beispiel-Repository, das keine Zugangsdaten benötigt. Dafür brauchen Sie später den „None“-Schlüssel.

> Statt wie bei Ansible üblich auf der Kommandozeile, verwalten Sie Ihre Ziel-Hosts und die zuge
> Statt wie bei Ansible üblich auf der Kommandozeile, verwalten Sie Ihre Ziel-Hosts und die zugehörigen Zugangsdaten mit Semaphore über eine Weboberfläche.

Wechseln Sie anschließend zum Menü namens Inventory. Hier erstellen Sie eine Liste der Ziel-Hosts. Die Liste entspricht der Inventory-Datei, die Ansible gewöhnlich in /etc/ansible/hosts sucht oder deren Pfad Sie auf der Kommandozeile mit dem Parameter `-i` übergeben. Eine Inventory-Datei vom Typ „Static“ im Ini-Stil sieht beispielsweise so aus:

```
[hosts]
192.168.1.101
192.168.1.102
192.168.1.103
```

Zusätzlich müssen Sie das Inventory benennen und eine der zuvor konfigurierten Authentifizierungsmethoden angeben.

Wechseln Sie jetzt in das Environment-Menü. Environments enthalten Variablen, mit denen Sie Playbooks weiter anpassen können. Für jeden Task, den Sie mit Semaphore ausführen, müssen Sie ein Environment angeben, auch wenn Sie keine Variablen definieren wollen. Also richten Sie ähnlich wie beim „None“-Schlüssel ein leeres Environment ein. Tragen Sie dafür beim Erstellen einfach `{}` in den Feldern „Extra variables“ und „Environment variables“ ein und vergeben einen Namen.

Als letzte Zutat braucht Semaphore noch eine Quelle an Playbooks. Um die zu versionieren und gemeinsam zu bearbeiten ist es üblich, sie in GitHub-Repositories abzulegen. Damit Sie sich mit Semaphore vertraut machen können, haben wir ein öffentliches GitHub-Repository mit einer Reihe simpler Beispiel-Playbooks erstellt. Das können Sie natürlich auch forken, um die Playbooks anzupassen.

Legen Sie in Semaphore im Menü namens Repositories einen neuen Eintrag an. Vergeben Sie einen Namen, beispielsweise ansible-examples, fügen Sie die URL des Repository (https://github.com/ndi-ct/ansible-examples) oder Ihres Forks ein und weisen Semaphore an, den Branch namens „main“ zu nutzen. Weil es ein öffentliches Repository ist, reicht der zuvor konfigurierte Access Key „None“. Wenn Sie später mit Semaphore private Git-Repositories anzapfen wollen, müssen Sie den entsprechenden SSH-Schlüssel im Key Store hinterlegen.

To-do-Liste abhaken

Semaphore hat jetzt alle nötigen Informationen, damit Ansible loslegen kann. Erstellen Sie im Menü namens Task Templates ein neues Template, indem Sie auf „Create Template“ klicken. Für einen Testlauf bietet sich das integrierte Ping-Modul `ansible.builtin.ping` von Ansible an. Ping prüft, ob Ansible eine SSH-Verbindung zu den Zielservern aufbauen kann und ob Python installiert ist. Wenn nicht, gibt Ping eine Fehlermeldung aus.

Das Playbook mit dem Namen ping.yml umfasst nur wenige Zeilen YAML-Code und steckt mit im Beispiel-Repository:

```
---
- hosts: all
  tasks:
    - ansible.builtin.ping:
```

Semaphore unterteilt Task Templates mit drei verschiedenen Labels. „Task“ bietet sich für Administrations- und Konfigurationsaufgaben an. „Build“ ist für die Integration von Semaphore in CI/CD-Pipelines und „Deploy“ eignet sich für komplexere Softwareinstallationen, beispielsweise für einen Verbund von Docker-Containern, wie im telerec’t-Projekt.

Der Ping-Testlauf passt am besten zu „Task“. Geben Sie dem Task Template einen Namen, beispielsweise „Testlauf“. Außerdem müssen Sie den Namen des Playbooks (ping.yml), sowie das Inventory, das Repository und das Environment so wie im Screenshot auf Seite 155 definieren. Die restlichen Angaben sind optional, beispielsweise Kommandozeilenparameter (CLI Args) wie `--become`, wenn die Tasks Systemverwalterrechte benötigen.

> Task Templates stehen im Mittelpunkt von Semaphore und enthalten ein Playbook, ein Reposi
> Task Templates stehen im Mittelpunkt von Semaphore und enthalten ein Playbook, ein Repository, Variablen (Environment) und Ziel-Hosts (Inventory).

Klicken Sie in der Liste der Task Templates jetzt auf den Namen Ihres Templates und anschließend auf die Schaltfläche „Run“. Die zusätzlichen Optionen „Debug“, „Dry Run“ und „Diff“ können Sie erst mal ignorieren. Sie helfen bei Testläufen und der Fehlersuche, sollte ein Playbook mal nicht funktionieren.

Jetzt können Sie sich zurücklehnen und Ansible bei der Arbeit über die Schulter schauen. Zunächst fischt Ansible das Playbook ping.yml aus dem konfigurierten GitHub-Repository. Danach führt es den Task `ansible.builtin.ping` aus. Wurde der Task auf dem Zielserver erfolgreich abgeschlossen,

quittiert Ansible das in der Ausgabe mit `ok`:

```
TASK [ansible.builtin.ping] ***
ok: [192.168.1.101]
ok: [192.168.1.102]
ok: [192.168.1.103]
```

Wiederkehrende Aufgaben

Eine Semaphore-Instanz eignet sich besonders gut, um wiederkehrende Aufgaben auf Ziel-Hosts zu automatisieren. Um das zu zeigen, nutzen wir ein simples Playbook namens updates.yml, das mit dem Paketmanager apt prüft, ob neue Updates vorliegen, diese installiert und nicht mehr benötigte Pakete entfernt:

```
---
- name: Update packages via apt
  hosts: all
  gather_facts: true

  tasks:
    - name: Update package cache
      apt:
        update_cache: yes

    - name: Upgrade packages
      apt:
        upgrade: dist
        autoclean: yes
```

Erstellen Sie ein weiteres Task Template nach dem Vorbild des Ping-Beispiels, aber tragen Sie diesmal bei „Playbook Filename“ den Namen updates.yml ein. Um die Aufgabe zu terminieren, müssen Sie einen Zeitpunkt bei „Cron“ eintragen, beispielsweise `03***`, um die Aufgabe jeden Tag um 3 Uhr auszuführen. Wenn Sie Schwierigkeiten haben, Ihren Wunschzeitpunkt in eine Cron-Expression zu übersetzen, hilft das Onlinetool crontab guru, das wir unter ct.de/y4au verlinkt haben. Semaphore führt das Playbook ab jetzt stets zum konfigurierten Zeitpunkt aus.

> Wenn Sie einen Task in Semaphore anschieben, kann man Ansible in einem Ausgabefenster b
> Wenn Sie einen Task in Semaphore anschieben, kann man Ansible in einem Ausgabefenster bei der Arbeit zusehen.

Um Administratoren über den Status von Aufgaben zu informieren, kann Semaphore Nachrichten an Kanäle im Messenger Telegram verschicken. Das hat bei unserem Testlauf mit der Snap-Variante aber nicht funktioniert. Wenn Sie trotzdem auf dem Laufenden bleiben wollen, können Sie stattdessen eines der integrierten Benachrichtigungsmodule von Ansible nutzen (siehe ct.de/y4au).

Das Playbook mit dem Namen update-notification.yml in unserem Beispiel-Respository enthält eine Vorlage, um mittels Webhook eine Nachricht an einen Discord-Server zu schicken:

```
    - name: Discord-Notification
      community.general.discord:
        webhook_id: "id"
        webhook_token: "token"
        content: "Software update complete on {{ ansible_hostname }}."
```

Sie müssen lediglich die Platzhalter `id` und `token` durch die ID und das Token Ihres Discord-Webhook ersetzen. Die URL, die beide Werte enthält, zeigt Discord an, wenn Sie in den Kanaleinstellungen im Menü „Integration“ einen neuen Webhook erstellen. Die ID und das Token folgen auf `webhooks/` und werden durch `/` getrennt. Die Variable `{{ ansible_hostname }}` befüllt Ansible automatisch mit dem Hostnamen.

Fazit

Semaphore erweitert Ansible um eine grafische Benutzeroberfläche, mit der Sie sich im Handumdrehen eine Serverschaltzentrale einrichten. Das erleichtert den Einstieg in die Automatisierung mit Ansible und hilft dabei, die eigene Serverflotte zuverlässig in den gewünschten Zustand zu bringen. Wer eine Ansible-GUI möchte und wem Funktionen von Semaphore nicht mehr ausreichen, sollte einen Blick auf AWX werfen, das als Grundlage für die Red Hat Ansible Automation Platform dient, aber vorrangig für den Betrieb in einem Kubernetes-Cluster gedacht ist. *(ndi@ct.de)*

1. Literatur

2. Klaus Gref und Pina Merkert, Telerec’t, Server im Rechenzentrum oder daheim mit Ansible automatisieren, c’t 1/2024, S. 150
3. Klaus Gref und Pina Merkert, Ordnung im Königreich, Die Grundausstattung für öffentliche Server – automatisiert mit Ansible, c’t 02/2024, S. 154
4. Niklas Dierking, Schlüsselmeister, Sicher und komfortabel arbeiten mit SSH, c’t 21/2022, S. 172

GitHub-Repository ansible-examples, Semaphore-Dokumentation: ct.de/y4au

---

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