Server mit Ansible verwalten

Andreas Krause, Jens Kubieziel, Andreas Scherbaum

11. März 2018

Organisatorisches

- Geplante Dauer: 3 Stunden
- Nach einer kurzen Einführung gibt es Übungen.
- Wir stellen euch AWS-Instanzen zur Verfügung.

Organisatorisches

- Geplante Dauer: 3 Stunden
- Nach einer kurzen Einführung gibt es Übungen.
- Wir stellen euch AWS-Instanzen zur Verfügung.

Ziel: Betrieb einer kleinen PHP-Anwendung mit Web- und Datenbankserver verwaltet über Ansible

Wir
Kurze Vorstellung

- Andreas Krause
- Jens Kubieziel
- Andreas Scherbaum

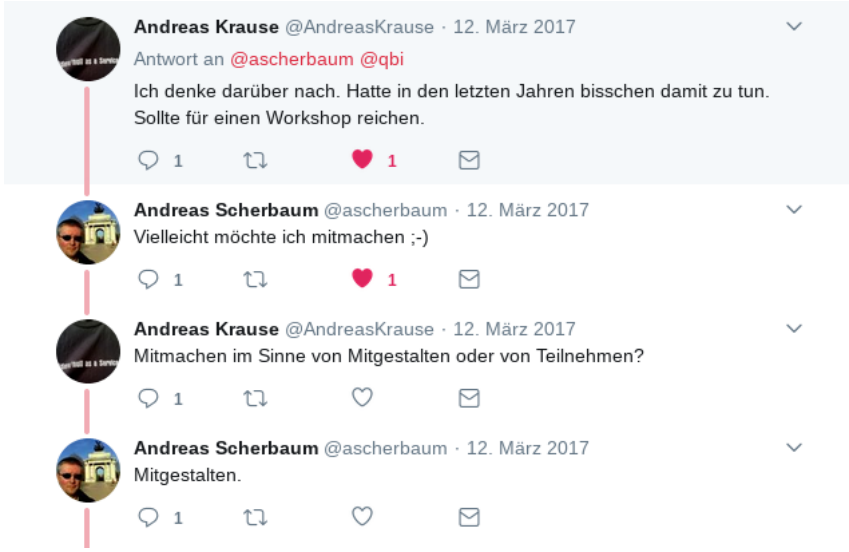
Was ist Ansible?

1. Ein Kommunikationsgerät, mit dem ihr mit Überlichtgeschwindigkeit kommunizieren könnt (Ursula K. Le Guin)

Was ist Ansible?

1. Ein Kommunikationsgerät, mit dem ihr mit Überlichtgeschwindigkeit kommunizieren könnt (Ursula K. Le Guin)
2. Eine Software, die bei der Verwaltung und der Konfiguration von Servern hilft.

Warum Ansible?



Was ist Ansible?
Details

- Werkzeug zur Automatisierung von Administrationsaufgaben

Was ist Ansible?
Details

- Werkzeug zur Automatisierung von Administrationsaufgaben
- Freie Software

Was ist Ansible?
Details

- Werkzeug zur Automatisierung von Administrationsaufgaben
- Freie Software
- in Python entwickelt

Wie funktioniert Ansible?

Ihr beschreibt den gewünschten Zustand der Zielsysteme. Ansible loggt sich per SSH ein und führt ggf. gewünschte Aktionen aus.

Wie funktioniert Ansible?
Voraussetzungen

- SSH
- Python (Version 2.6 bzw. Python 3)

Wie installiere ich Ansible?

- Über die Paketverwaltung deines GNU/Linux-Systems:
  - `apt install ansible`
  - Unter Ubuntu gibt es ein PPA: `apt-add-repository ppa:ansible/ansible`
  - `yum install ansible` (ggf. das EPEL-Repository aktivieren)
  - `emerge -av app-admin/ansible`

Wie installiere ich Ansible?

- Über die Paketverwaltung deines GNU/Linux-Systems:
  - `apt install ansible`
  - Unter Ubuntu gibt es ein PPA: `apt-add-repository ppa:ansible/ansible`
  - `yum install ansible` (ggf. das EPEL-Repository aktivieren)
  - `emerge -av app-admin/ansible`
- Aus den Quellen:
  - `pip install ansible`
  - `tar.gz` von `https://github.com/ansible/ansible/releases`

Übung 1

Bevor wir mit Ansible loslegen, wollen wir wissen, ob ihr euch auf den Maschinen einloggen könnt.

Führt die Übung 1 im Verzeichnis uebungen/`01-ssh` aus.

Übung 2

Mit dem ersten Ansible-Kommando wollen wir die Maschinen anpingen. Woher weiß Ansible, mit welchen Maschinen es reden soll?

Inventory

Das Inventory sammelt die diversen Systeme und besteht aus einer oder mehreren Dateien:

hosts

```
192.168.23.42
clt.18.example.org
[web]
192.168.17.189
clt.18.example.org
```

Inventory

Format

Das Inventory kann im INI-Format vorliegen

Inventory

Format

Das Inventory kann im INI-Format vorliegen

hosts

```
192.168.23.42
clt.18.example.org
[web]
192.168.17.189
clt.18.example.org
```

Inventory

Format

Das Inventory kann im INI-Format vorliegen oder als YAML-Datei:

Ad-Hoc-Modus

Mit dem Aufruf von Ansible auf der Kommandozeile lassen sich verschiedene Befehle mitgeben. Diese landen eventuell in der Shellhistory, sind aber ansonsten weg. Diese Art von Aufruf wird als *Ad-Hoc-Modus* bezeichnet.

Ad-Hoc-Modus

Mit dem Aufruf von Ansible auf der Kommandozeile lassen sich verschiedene Befehle mitgeben. Diese landen eventuell in der Shellhistory, sind aber ansonsten weg. Diese Art von Aufruf wird als *Ad-Hoc-Modus* bezeichnet.
Der Aufruf enthält die betreffenden Hosts sowie Optionen:

- `-i` bezeichnet den Ort des Inventorys
- `-m` Modul, welches ausgeführt werden soll
- `-a` Argumente zum obigen Modul bzw. Shell-Kommando (`command`-Modul)
- `-u` Benutzername (Standard: aktueller Benutzername)

Ad-Hoc-Modus

Mit dem Aufruf von Ansible auf der Kommandozeile lassen sich verschiedene Befehle mitgeben. Diese landen eventuell in der Shellhistory, sind aber ansonsten weg. Diese Art von Aufruf wird als *Ad-Hoc-Modus* bezeichnet.
Der Aufruf enthält die betreffenden Hosts sowie Optionen:

- `-i` bezeichnet den Ort des Inventorys
- `-m` Modul, welches ausgeführt werden soll
- `-a` Argumente zum obigen Modul bzw. Shell-Kommando (command-Modul)
- `-u` Benutzername (Standard: aktueller Benutzername)

Hello World

```
ansible all -i hosts -a '/bin/echo Hello World'
```

Übung 2

Für Ansible gibt es das Modul ping, welches einen oder mehrere Hosts kontaktiert und das Ergebnis zurückmeldet.

Führt die Übung 2 im Verzeichnis uebungen/`02-ping` aus.

Module

Module abstrahieren Konfigurations- bzw. Sysadmin-Aufgaben. Diese machen die eigentliche Arbeit. Eine Liste aller verfügbaren Module ist auf `http://docs.ansible.com/ansible/latest/modules_by_category.html`.

Module
Idempotenz

Ein wichtiges Konzept von Ansible, wie auch ähnlichen Managementprogrammen, ist die *Idempotenz*. Das heißt, ein mehrfacher Aufruf hat denselben Effekt wie einmalige Ausführung. Module sollten die Idempotenz garantieren, d. h. wenn diese feststellen, dass der gewünschte Stand erreicht ist, nehmen diese keine Änderung am System vor.

Module
Dokumentation

Die Dokumentation der Module kann über die Kommandozeile mittels `ansible-doc modulname` oder über die Webseite abgerufen werden.

Shellmodul

`ansible-doc shell` oder
`http://docs.ansible.com/ansible/latest/shell_module.html`

Übung 3

Führt weitere Ad-Hoc-Befehle aus, siehe `uebungen/03-addoc`.

Playbooks

Playbooks sind das zentrale Werkzeug von Ansible. Darüber wird Konfiguration, Deployment und Orchestration gesteuert.
Ein Playbook besteht aus einem oder mehreren kleinen Anleitungen, die angeben, was auf welchen Hostrechnern zu tun ist.

Playbooks

Begriffe

Task ist der Aufruf eines Moduls mit diversen Parametern.

Play ist die Abfolge mehrerer Tasks auf Rechnern aus dem Inventory.

Playbook ist eine Zusammenstellung eines oder mehrerer Plays.

Playbooks

Aufruf

Die Playbooks werden über einen speziellen Befehl aufgerufen:

Aufruf der Playbooks

```
ansible-playbook foo.yml
```

Playbooks

Aufruf

Die Playbooks werden über einen speziellen Befehl aufgerufen:

Aufruf der Playbooks

```
ansible-playbook foo.yml
```

Was passiert, wenn ein Playbook mehrfach nacheinander aufgerufen wird?

YAML

Ansible nutzt die Beschreibungssprache YAML für Playbooks und anderes. Im folgenden findet ihr einen kurzen Überblick über die Syntax.

YAML

Ansible nutzt die Beschreibungssprache YAML für Playbooks und anderes. Im folgenden findet ihr einen kurzen Überblick über die Syntax.
Der Start eines Dokuments kann durch drei Striche (`---`) und das Ende durch drei Punkte (`...`) markiert werden.

Das übliche Hallo-Welt-Beispiel

```
---
Hallo: Welt
...
```

YAML

Syntax

- Listen starten mit einem Anstrich (-) und Leerraum.

```
- Hallo
- Welt
```

YAML

Syntax

- Listen starten mit einem Anstrich (-) und Leerraum.

```
- Hallo
- Welt
```

- Dictionarys bestehen aus Schlüsselwert, :, Leerzeichen und Wert.

```
Hallo: Welt
Linux: Debian
```

Beides lässt sich kombinieren und verschachteln.

YAML

Beispiel

Stellt euch vor, ihr wollt sicherstellen, dass auf einem neuen Host ein bestimmter Nutzer angelegt ist, dieser eure Lieblingsshell als Login-Shell hat und diese natürlich installiert ist. Welche Schritte würdet ihr machen, um dies zu realisieren?

YAML

Beispiel

Stellt euch vor, ihr wollt sicherstellen, dass auf einem neuen Host ein bestimmter Nutzer angelegt ist, dieser eure Lieblingsshell als Login-Shell hat und diese natürlich installiert ist. Welche Schritte würdet ihr machen, um dies zu realisieren?

Nutzer anlegen

```
- hosts: all
  tasks:
  - name: Shell installieren
    apt: name=fish state=installed
  - name: Nutzer anlegen
    user: name=clt18 shell=/usr/bin/fish
```

YAML
Variablen

Später benötigen wir Variablen:

```
name: "{{ Variable }}"
```

Übung 4

Entwerft nun ein Playbook, welches den Hostnamen setzt, einen NTP-Server installiert und sicherstellt, dass der Daemon auch läuft.

Rollen

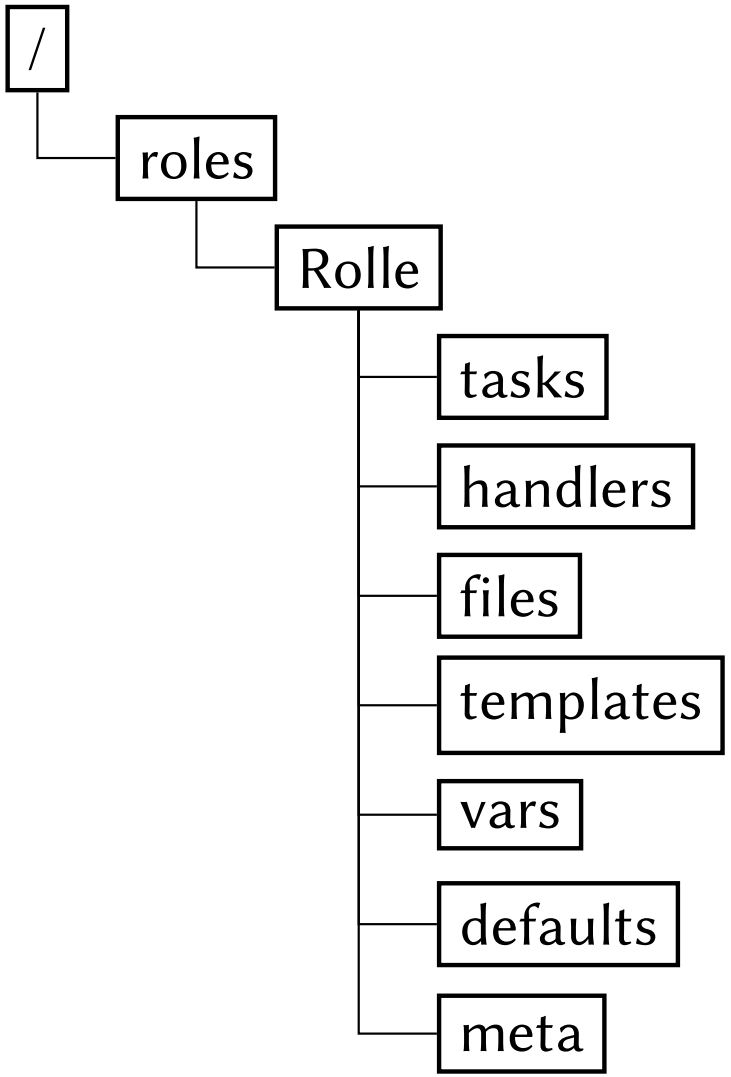
Einführung

Mit den Methoden können wir nun ein großes Playbook schreiben. Aber irgendwann kommt der Punkt, wo die Arbeit besser organisiert werden soll. Denn in der Regel sollen viele kleine Aufgaben ausgeführt werden statt einer großen.
Seit Ansible 2.4 gibt es die Möglichkeit, Inhalte anderer Dateien einzubinden (`import` und `include`).
Rollen sind ein älteres Mittel. Diese greifen auf eine vordefinierte Verzeichnisstruktur zurück und können Tasks ausführen, auf Variablen zugreifen etc.

Rollen

Verzeichnisstruktur

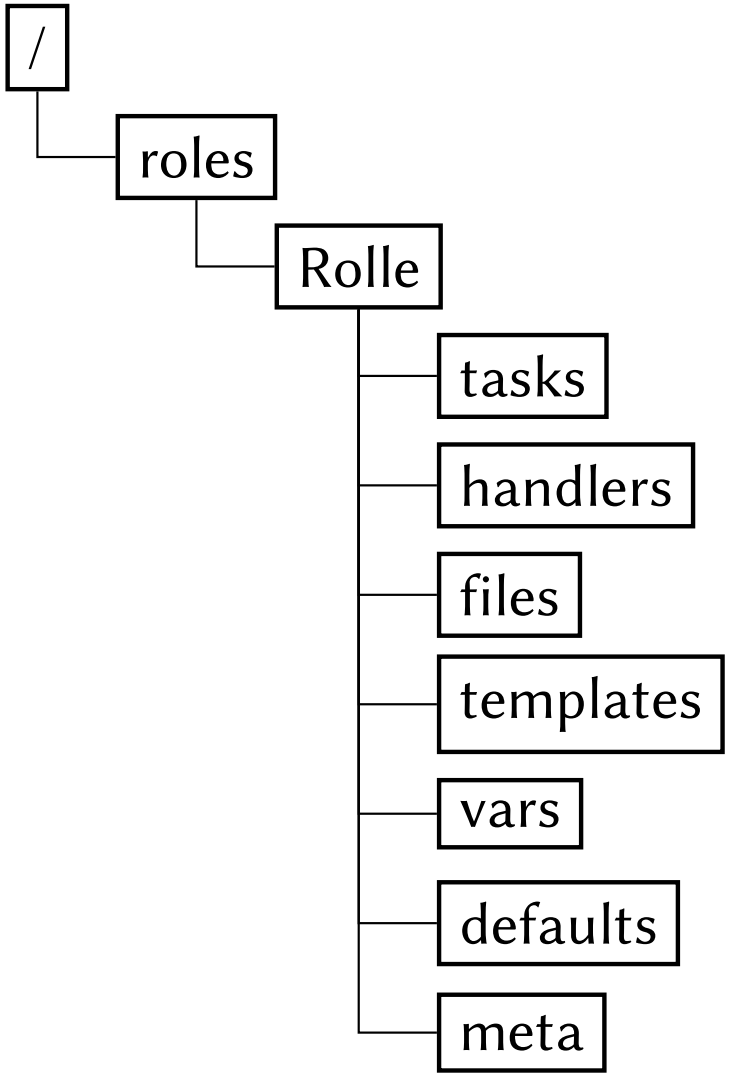
- Mindestens eines der Verzeichnisse muss existieren



Rollen

Verzeichnisstruktur

- Mindestens eines der Verzeichnisse muss existieren
- Die existierenden Verzeichnisse müssen eine Datei mit dem Namen `main.yml` enthalten.



Rollen

Inhalt der Verzeichnisse

tasks	enthalten die Liste an Tasks, die durch die Rolle ausgeführt wird
handlers	Handler, die durch die Rolle benutzt werden
files	Dateien, die von dieser Rolle benutzt werden
templates	Templates, die dann deployt werden
vars	Variablen für die Rolle
defaults	Standardwerte für Variablen
meta	Abhängigkeiten der Rolle

Übung 5

In der vorigen Übung haben wir den Hostnamen gesetzt und einen NTP-Server installiert. Baut das nun um, dass diese Aufgaben als Rolle ausgeführt werden.

Übung 6

Jetzt gehen wir einen Schritt weiter und bauen uns eine „Serverlandschaft“ auf. Wir haben Server, die als Webserver und solche, die als Datenbankserver fungieren. Bastelt jetzt Rollen, die in Abhängigkeit vom Host die entsprechenden Pakete installieren.

Webserver

- apache2
- apache2-utils
- libapache2-mod-php
- php
- php-dev
- php-pgsql
- php-pear
- php-gettext

PostgreSQL

- postgresql
- postgresql-client
- postgresql-contrib
- python-psycopg2
- postgresql-client-common
- postgresql-client