Server mit Ansible verwalten

Jens Kubieziel, Andreas Scherbaum, ~~Andreas Krause~~

16. März 2019

Agenda

1. Einleitung
2. Einführung in Ansible
3. Erste Schritte mit Ansible
   - Erreichbarkeit der Maschinen
   - Ad-Hoc-Befehle
4. Playbooks
   - Einführung zu Playbooks
   - Rollen
   - Konfiguration
   - Dateien kopieren

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

- Jens Kubieziel
- Andreas Scherbaum
- ~~Andreas Krause~~

Agenda

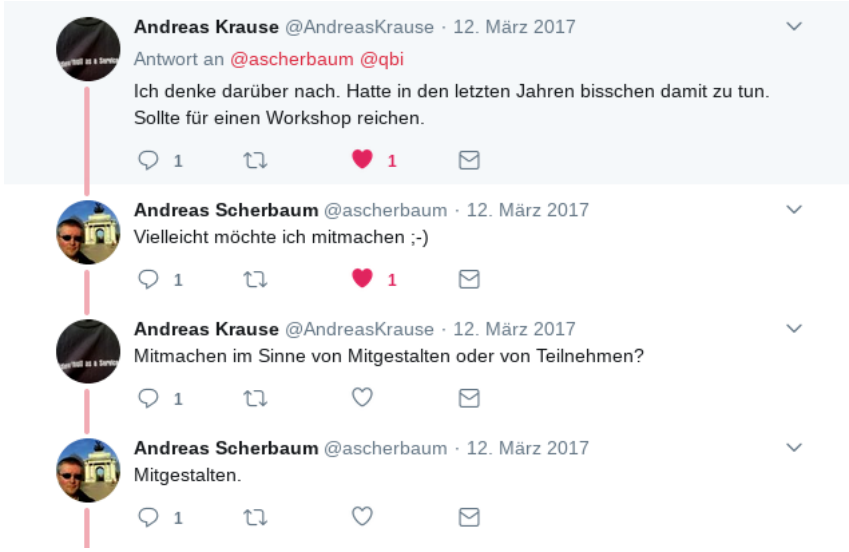
Was ist Ansible?

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

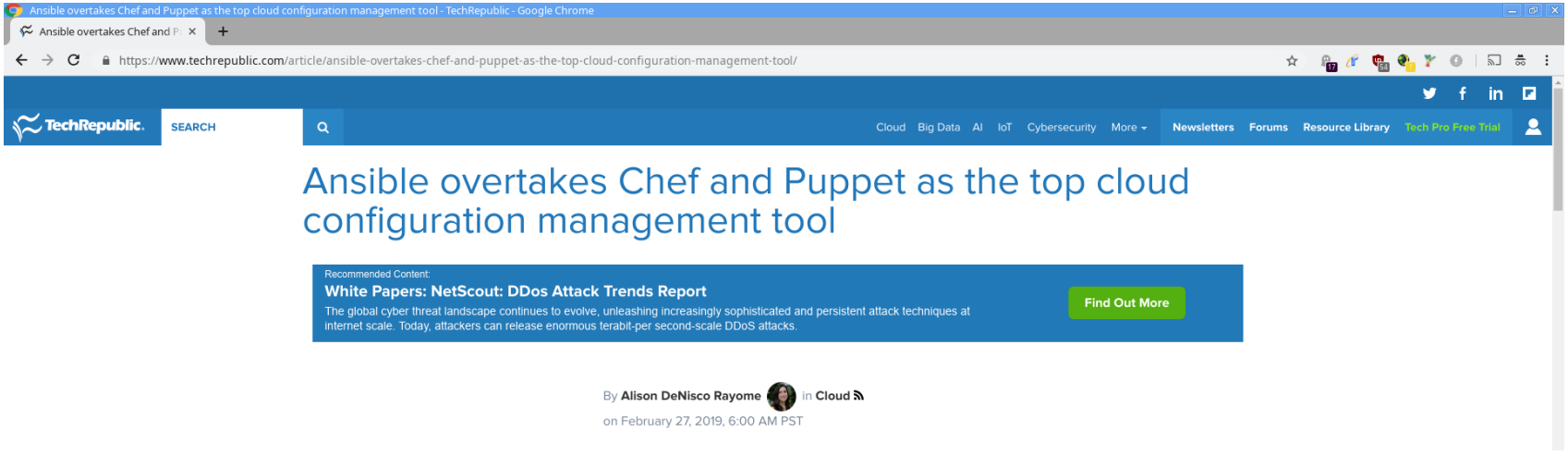
Was ist Ansible?

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?



Warum Ansible?



Quelle:

https://www.techrepublic.com/article/ansible-overtakes-chef-and-puppet-as-the-top-cloud-configuration-management-tool/

Was ist Ansible?
Details

- leichtgewichtiges Werkzeug zur Automatisierung von Administrationsaufgaben

Was ist Ansible?
Details

- leichtgewichtiges Werkzeug zur Automatisierung von Administrationsaufgaben
- Freie Software

Was ist Ansible?
Details

- leichtgewichtiges 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. notwendige 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`

Agenda

1. Einleitung
2. Einführung in Ansible
3. Erste Schritte mit Ansible
   - Erreichbarkeit der Maschinen
   - Ad-Hoc-Befehle
4. Playbooks
   - Einführung zu Playbooks
   - Rollen
   - Konfiguration
   - Dateien kopieren

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.19.example.org
[web]
192.168.17.189
clt.19.example.org
```

Inventory

Format

Das Inventory kann im INI-Format vorliegen

Inventory

Format

Das Inventory kann im INI-Format vorliegen

hosts als INI

```
192.168.23.42
clt.19.example.org
[web]
192.168.17.189
clt.19.example.org
```

Inventory

Format

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

hosts als YAML

```
all:
  hosts:
    192.168.23.42
    clt.19.example.org
  children:
    web:
      clt.19.example.org:
```

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 nirgendwo hinterlegt. 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 nirgendwo hinterlegt. 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 (z. B. shell)
- `-a` Argumente zum obigen Modul bzw. Shell-Kommando (command-Modul)
- `-u` Benutzername (Standard: aktueller Benutzername)
- `-b` Aktionen werden mit den Rechten des angegebenen Benutzers ausgeführt

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 nirgendwo hinterlegt. 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 (z. B. shell)
- `-a` Argumente zum obigen Modul bzw. Shell-Kommando (command-Modul)
- `-u` Benutzername (Standard: aktueller Benutzername)
- `-b` Aktionen werden mit den Rechten des angegebenen Benutzers ausgeführt

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 erledigen 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 die 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 aufgerufen 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`.

Agenda

1. Einleitung
2. Einführung in Ansible
3. Erste Schritte mit Ansible
   - Erreichbarkeit der Maschinen
   - Ad-Hoc-Befehle
4. Playbooks
   - Einführung zu Playbooks
   - Rollen
   - Konfiguration
   - Dateien kopieren

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, welcher Zustand auf welchen Zielrechnern erwünscht 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. Im folgenden findet ihr einen kurzen Überblick über die Syntax.

YAML

Ansible nutzt die Beschreibungssprache YAML für Playbooks. 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
  - name: Nutzer anlegen
    user: name=clt19 state=present shell=/usr/bin/fish
```

YAML
Variablen

Später benötigen wir Variablen:

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

YAML
Variablen

Nicht gesetzte Variablen führen zum Abbruch des Playbooks.

Übung 4

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

Übung 4

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

Vorüberlegungen

- Auf welchen Hosts soll die Aufgabe ausgeführt werden (`hosts`)?

Übung 4

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

Vorüberlegungen

- Auf welchen Hosts soll die Aufgabe ausgeführt werden (`hosts`)?
- Mit welchen Rechten wird die Aufgabe auf den Hosts aufgeführt (`become`, `become_user` etc.)?

Übung 4

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

Vorüberlegungen

- Auf welchen Hosts soll die Aufgabe ausgeführt werden (`hosts`)?
- Mit welchen Rechten wird die Aufgabe auf den Hosts aufgeführt (`become`, `become_user` etc.)?
- In welchen Schritten kann die Aufgabe ausgeführt werden?

Übung 4

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

Vorüberlegungen

- Auf welchen Hosts soll die Aufgabe ausgeführt werden (`hosts`)?
- Mit welchen Rechten wird die Aufgabe auf den Hosts aufgeführt (`become`, `become_user` etc.)?
- In welchen Schritten kann die Aufgabe ausgeführt werden?
- Welche Module werden benötigt? Welche Parameter sind sinnvoll?

Rollen

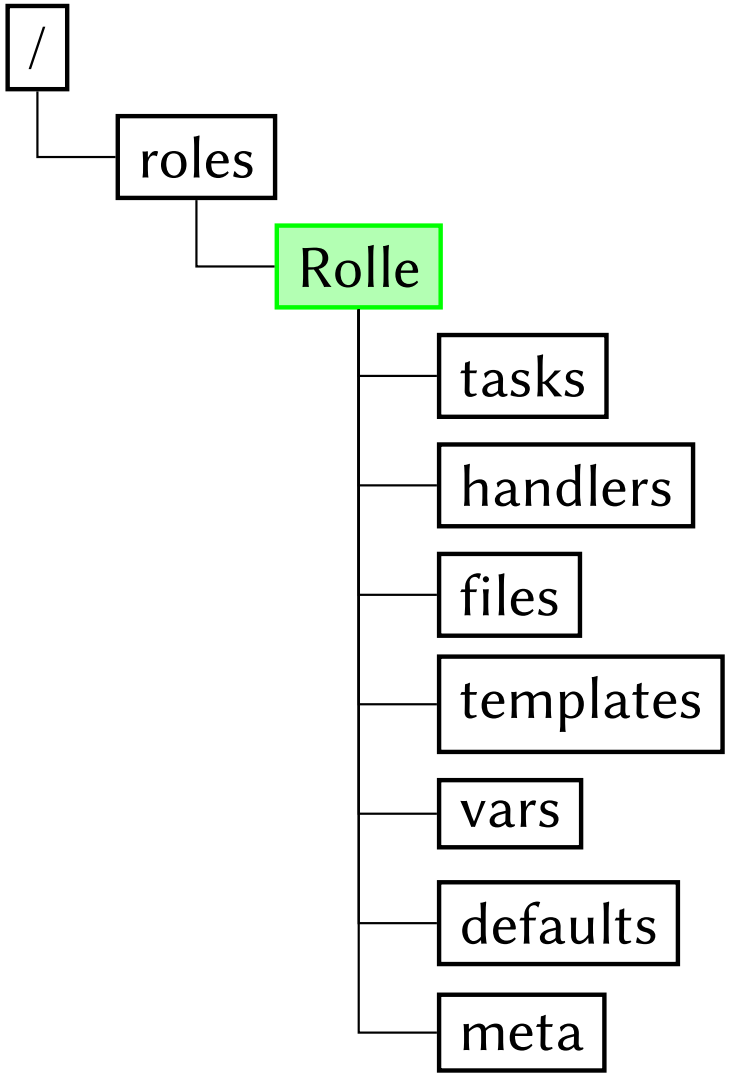
Einführung

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

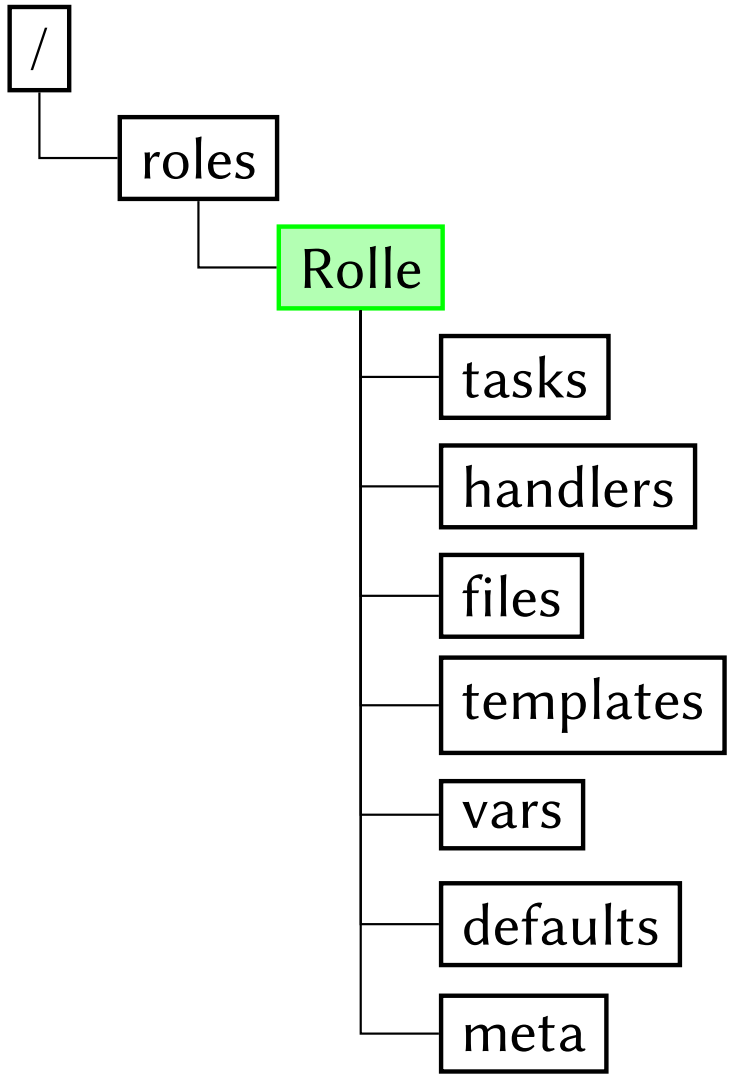
- Mindestens eines der Verzeichnisse muss existieren



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.

Rollen

Schleifen in Playbooks

Für die Übung 5 reichte es, einen Ordner für die Tasks anzulegen.
Nun wollen wir auch mit Variablen arbeiten und Schleifen nutzen.

Rollen

Schleifen in Playbooks

Über den Schlüssel `with_items` können wir eine Liste von Werten anlegen. Der Zugriff darauf erfolgt mittels `{{ item }}`.

Rollen

Schleifen in Playbooks

Über den Schlüssel `with_items` können wir eine Liste von Werten anlegen. Der Zugriff darauf erfolgt mittels `{{ item }}`.

Playbook mit Schleife

```
- name: Installiere Shell und git
  package: "{{ item }}"
  state: latest
  with_items:
  - fish
  - git
```

Rollen

Werte in Variablen speichern

Ansible erlaubt es, ausgegebene Werte in Variablen zu speichern und später wieder darauf zurückzugreifen. Der Schlüssel `register` legt den Namen der Variablen fest.

Variable setzen

```
tasks:
  - shell: /usr/bin/git branch
    register: branch_name
```

auf Variable zugreifen

```
    foo: {{ branch_name.stdout }}
```

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.
Für den Webserver benötigen wir die folgenden Pakete und möchten sicherstellen, der Apache nach erfolgreicher Installation auch gestartet ist:

Webserver

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

Übung 6

Als Datenbank nutzen wir PostgreSQL. Allerdings wollen wir dies gern von `https://www.postgresql.org/download/linux/debian/` installieren. Welche Schritte sind hierfür notwendig?

PostgreSQL

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

Konfiguration

Bisher installierten wir Dienste, starteten Dienste und sahen, wie neue Nutzer eingerichtet werden können. Doch wie lässt sich mittels Ansible die Konfiguration von Rechnern anpassen?

Konfiguration

Bisher installierten wir Dienste, starteten Dienste und sahen, wie neue Nutzer eingerichtet werden können. Doch wie lässt sich mittels Ansible die Konfiguration von Rechnern anpassen?
Wir nutzen dafür das Modul `lineinfile`. Dieses stellt sicher, dass eine bestimmte Zeile in einer Datei vorhanden bzw. nicht vorhanden ist.

Bestimmten Nameserver entfernen

```
lineinfile:
  dest: /etc/resolv.conf
  regexp: '^nameserver'
  line: 'nameserver 8.8.8.8'
  state: absent
```

Handler

Bei Änderungen auf einem Remote-System müssen manchmal Aktionen ausgeführt werden. Ein klassisches Beispiel sind Änderungen an der Konfiguration. Diese sind verbunden mit einem Reload oder Neustart des betreffenden Dienstes.
Diese Aktionen sollen nicht bei jedem Aufruf eines Playbooks passieren, sondern nur bei Bedarf.

Handler

Bei Änderungen auf einem Remote-System müssen manchmal Aktionen ausgeführt werden. Ein klassisches Beispiel sind Änderungen an der Konfiguration. Diese sind verbunden mit einem Reload oder Neustart des betreffenden Dienstes.
Diese Aktionen sollen nicht bei jedem Aufruf eines Playbooks passieren, sondern nur bei Bedarf. Ansible hat hierfür das Konzept der Handler eingebaut.

Handler

Am Ende eines Plays werden die Aktionen unter `notify` aufgeführt und die Liste an Aufgaben steht unter `handlers`:

Apache neustarten

```
- name: Foo
  ...
  notify:
    - restart apache
...
handlers:
  - name: restart apache
    service: name=apache state=restarted
```

Lookups

Mittels Lookups können wir innerhalb von Ansible auf andere Quellen zugreifen. Das heißt, im Dateisystem lesen wie auch externe Dienste kontaktieren.

Lookups

Mittels Lookups können wir innerhalb von Ansible auf andere Quellen zugreifen. Das heißt, im Dateisystem lesen wie auch externe Dienste kontaktieren.

```
- hosts: all
  vars:
     contents: "{{ lookup('file', '/etc/debian_version')
          }}"
  tasks:
     - debug: msg="Die Debian-Version ist {{ contents
         }}"
```

Lookups

Passwort erzeugen

Mittels password können wir ein Passwort erzeugen und dies in eine Datei schreiben. Sollte die Datei bereits existieren, wird der Inhalt ausgelesen. Standardmäßig erzeugt Ansible einen Mix aus Buchstaben, Zahlen und Sonderzeichen mit 20 Zeichen.

```
password: "{{ lookup('password', 'PIN chars=digits
    length=4) }}"
password: "{{ lookup('password', '/home/user/.geheim/
    passwort chars=ascii_letters,digits,hexdigits,
    punctuation') }}"
```

Übung 7

Wir stellen nun das installierte PHP sowie die Datenbank nach unseren Wünschen ein:

- Wir aktivieren `short_open_tag`, deaktivieren `allow_url_fopen`, setzen die `date.timezone` auf Europe/Berlin und die `upload_max_filesize` auf `64M`.

Übung 7

Wir stellen nun das installierte PHP sowie die Datenbank nach unseren Wünschen ein:

- Wir aktivieren `short_open_tag`, deaktivieren `allow_url_fopen`, setzen die `date.timezone` auf Europe/Berlin und die `upload_max_filesize` auf `64M`.
- In der `postgresql.conf` wird der Wert `listen_addresses` auf `*` gesetzt.

Übung 7

Wir stellen nun das installierte PHP sowie die Datenbank nach unseren Wünschen ein:

- Wir aktivieren `short_open_tag`, deaktivieren `allow_url_fopen`, setzen die `date.timezone` auf Europe/Berlin und die `upload_max_filesize` auf `64M`.
- In der `postgresql.conf` wird der Wert `listen_addresses` auf `*` gesetzt.
- In der `pg_hba.conf` wird die Zeile `host test test 0/0 md5` eingefügt.

Übung 7

Wir stellen nun das installierte PHP sowie die Datenbank nach unseren Wünschen ein:

- Wir aktivieren `short_open_tag`, deaktivieren `allow_url_fopen`, setzen die `date.timezone` auf Europe/Berlin und die `upload_max_filesize` auf `64M`.
- In der `postgresql.conf` wird der Wert `listen_addresses` auf `*` gesetzt.
- In der `pg_hba.conf` wird die Zeile `host test test 0/0 md5` eingefügt.
- Schließlich wird ein Nutzer `test` und eine Datenbank `test` angelegt.

Dateien kopieren

Jetzt haben wir eine kleine Anwendung entwickelt. Diese soll auf den Server kopiert werden.

Dateien kopieren

Jetzt haben wir eine kleine Anwendung entwickelt. Diese soll auf den Server kopiert werden.
Dazu kann das copy-Modul verwendet werden. Ähnlich wie bei cp, muss mindestens Quelle und Ziel der Operation angegeben werden.

index.html auf den Webserver kopieren

```
- name: Nameserver-Einstellungen kopieren
  copy:
   src: resolv.conf
   dest: /etc
```

Das debug-Modul

Das debug-Modul gibt Informationen bei der Ausführung von Playbooks aus. Wir nutzen dies, um über die Variable `ansible_host` die IP-Adresse des Hosts anzuzeigen.

```
- debug:
  msg: "Die IP ist: {{ ansible_host }}."
```

Übung 8

Kopiert die Dateien in `../uebungen/08-simpleapp` auf den Server und lasst euch die IP-Adresse ausgeben.

Templates

Neben dem Kopieren einfacher Dateien können wir auch Templates anfertigen. Ansible baut daraus die korrekte Datei und lädt diese ins Zielsystem. Die Basis für die Templates ist Jinja2.

Aufruf mit Template-Modul

```
- name: Konfiguration
  template: src=config.j2 dest=/etc/programm/tor.{{item.
      host}}.conf
  with_items: {{ ipadressen }}
```

Jinja2

Die Template-Sprache Jinja2 kommt aus dem Dunstkreis von Python und funktioniert mit aktuellen Version (2.6.x, 2.7.x, ab 3.3.x) der Sprache.

Jinja2
Variablen

Im Unterordner `vars` können Variablen in die Datei `main.yml` eingebaut werden. Auf diese greift die Template-Datei zu und fügt die Werte ein.

```
ipadressen:
  - host: Bridge1
    ip: 192.168.192.23
    port: 12345
  - host: Bridge2
    ip: 192.168.192.42
    port: 4521
```

Übung 9

Nutzt den Wert `ansible_host` aus dem Inventory und übergebt den Wert in die `database.php`. Die so entstandene Datei soll mit dem Template-Modul auf den Server kopiert werden. Mittels debug solltet ihr euch wieder zumindest die IP-Adresse des Rechners ausgeben lassen.