Kapitel 1

Ablaufsteuerung

Flussablauf Ansible

- Bedingungen
- when
- Iterativ
- with_*
- Verwendung von Bedingungen mit den with_* Statement
- Verwendung von Variablen und facts um Subroutinen zu überspringen
- Selektiver Anwendung von Rollen
- Bedingungen in Jinja2-Vorlagen

1.1 Bedingungen

When-Satement:

- Ob eine Variable definiert ist
- Ob ein vorheriges Kommando erfolgreich abgelaufen ist
- Ob ein Task abgelaufen ist
- Ob die Zielplatform entsprechend der unterstützten Plattformen entspricht
- Ob eine bestimmte Datei vorhanden ist

```
- name: download wordpress
  register: wp_download
- name: extract wordpress
  when: wp_download.rc == 0
```

oder

```
- name: extract wordpress
  when: wp_download|success
- name: notify devops engineers
  when: wp_download|failed
```

Falls ein Kommando scheitern muss/darf verwendet man:

```
ignore_errors: True
```

Kapitel 2

Facts als Grundlage von Entscheidungen

Abhängig vom Zielsystem können Entscheidungen getroffen werden.

- Entscheidung ob ein task ausgeführt werden darf
- Entscheidung ob eine Datei inkludiert wird
- Entscheidung ob eine Datei importiert wird
- Entscheidung ob eine Rolle auf einem Zielsystem ausgeführt wird

Beispiel:

```
ansible_os_family
```

Kapitel 3

Refactoring mysql-Rolle

Um auf eine MySQL-DB zuzugreifen, wird der mysql-client benötigt. Diese sollen nicht nur auf die Gruppe `db` sondern auch auf die Gruppe `webserver` installiert werden.

Hier der überarbeite Hash (Dictionary)

```
mysql:
  config:
    user: mysql
    port: 3306
    datadir: /var/lib/mysql
    bind: 127.0.0.1
    pid: /var/run/mysqld/mysqld.pid
    socket: /var/run/mysqld/mysqld.sock
    cnfpath: /etc/mysql/my.cnf
  service: mysql
  pkg:
    server: mysql-server
    client: mysql-client
    python: python-mysqldb
  server: false
```

Zugriff:

```
#file: roles/mysql/tasks/main.yml
- include: configure.yml
  when: mysql.server
- include: service.yml
  when: mysql.server
```

oder

```
---
# filename: roles/mysql/tasks/install_Debian.yml
- name: install mysql client
```

```
  apt:
    name: "{{ mysql['pkg']['client'] }}"
    update_cache: yes
- name: install mysql server
  apt:
    name: "{{ mysql['pkg']['server'] }}"
    update_cache: yes
    when: mysql.server
```

3.1 Zusammenfügen von Hashes

In der Ansible-Konfigaration:

```
# ansible.cfg
# http://docs.ansible.com/ansible/intro_configuration.html#hash-
   behaviour
hash_behaviour=merge
```

Somit kann in `db.yaml` folgendes stehen:

```
vars:
  mysql:
    server: true
    config:
      bind: "{{ ansible_eth1.ipv4.address }}"
```

3.2 Bedingungen in Vorlagen

Beispiel:

```
{% if condition %}
do_some_thing
{% elif condition2 %}
do_another_thing
{% else %}
do_something_else
{% endif %}
```

Siehe Diff-Datei

```
user = {{ mysql['config']['user'] | default("mysql") }}
{% if mysql.config.pid is defined %}
  pid-file    = {{ mysql['config']['pid'] }}
{% endif %}
{% if mysql.config.pid is defined %}
```

```
  socket          = {{ mysql['config']['socket'] }}
{% endif %}
{% if mysql.config.pid is defined %}
  port        = {{ mysql['config']['port'] }}
{% endif %}
{% if mysql.config.pid is defined %}
  datadir         = {{ mysql['config']['datadir'] }}
{% endif %}
{% if mysql.config.pid is defined %}
  bind-address = {{ mysql['config']['bind'] }}
{% endif %}
```

3.3 Einen Task einmalig laufen lassen

Zum Beispiel einmalig die Datenbank einzurichten.

```
run_once: true
```