codecentric

Ansible_

Variablen

Recap

Konfigurationsmanagement

- Konfiguration von Systemen - Infrastructure as Code

Verteilung und Orchestrierung von Software

- Orchestrierung von Systemen
- Zero-Downtime Updates

Ad-hoc Kommandos

- Operation am offenen Herzen

Basics

Variablen - Motivation

Automation macht wiederholenden Einsatz einfach

- Nicht alle Systeme sind identisch
- Status/Variablen der remote Maschine beeinflussen die Konfiguration

Variablen - Namen

Buchstaben [A-Z,a-z], Zahlen [0-9] und Unterstrich [_]

```
foo_port
foo5
```

```
foo.port
foo port
foo-port
12
```

Variablen - Level

Variablen in Inventory

Variablen in Playbook

Variablen in Include und Rollen

Variablen - Inventory

Variablen in Inventory

Host

```
[atlanta]
host1 http_port=80
host2 http_port=303
```

Group

```
[atlanta]
host1
host2

[dacota]
host3
host4

[atlanta:vars]
http_port=303
```

Variablen - Level

Variablen in Playbook

```
- hosts: webservers
  vars:
    http_port: 80
```

Variablen - Level

Variablen in Include und Rollen

Include

```
tasks:

  - include: wordpress.yml
    vars:
        wp_user: timmy
        ssh_keys:
          - keys/one.txt
          - keys/two.txt
```

Rollen

```
---

- hosts: webservers
  roles:
    - common
    - { role: foo_app_instance,app_port:5000 }
```

/defaults

/vars

S

Variablen - Nutzung

jinja2 Templating Engine

```
My amp goes to {{ max_amp_value }}
```

```
template: src=foo.cfg.j2 dest={{ remote_install_path }}/foo.cfg
```

Gotcha

```
- hosts: app_servers
  vars:
      app_path: {{ base_path }}/22
```

→

```
- hosts: app_servers
  vars:
      app_path: “{{ base_path }}/22”
```

Facts

Variablen - System Facts

Daten des remote Systems auslesen

```
ansible hostname -m setup
```

Können in Tasks verwendet werden

```
{{ ansible_hostname }}
```

Deaktivieren

```
- hosts: whatever
  gather_facts: no
```

Variablen - Benutzerdefinierte Facts

Facts, die vom Benutzer festgelegt werden

```
/etc/ansible/facts.d
```

Abrufen

```
ansible all -i inventory -m setup -a “ansible_local”
```

Variablen - Fact Caching

Referenzen zu anderen Hosts nutzen

```
{{ hostvars['asdf.example.com']['ansible_os_family'] }}
```

Entweder im Play bereits vorher von `asdf.example.com` **gelesen**

Oder Caching in `ansible.cfg` **konfigurieren:**

```
[defaults]
gathering = smart
fact_caching = redis
fact_caching_timeout = 86400 # seconds
```

```
[defaults]
gathering = smart
fact_caching = jsonfile
fact_caching_connection = /path/to
fact_caching_timeout = 86400
```

S

Erweiterte Themen

Variablen - Registrierung

Ergebnisse eines Task-Runs wiederverwenden

```
- hosts: web_servers

  tasks:

     - shell: /usr/bin/foo
       register: foo_result

     - shell: /usr/bin/bar
       when: foo_result.rc == 5
```

Conditionals

Variablen - Conditionals

Ergebnisse eines Task-Runs wiederverwenden

```
---
- name: Installiere Webserver
  hosts: all
  tasks:
  - name: Installiere apache
    apt: name=apache2 state=present
    when: ansible_os_family=="Debian"

  - name: Installiere httpd
    yum: name=httpd state=present
    when: ansible_os_family=="Centos"
```

S

Variablen - Conditionals

Logische Verknüpfungen

```
when: (ansible_distribution == "CentOS") or
      (ansible_distribution == "Debian" and ansible_distribution_version == "7")


when:
      - ansible_distribution == "Debian"
      - ansible_distribution_version == "7"
```

`is defined, is undefined, true/false, filter` **möglich**

Variablen - Komplexe Datenstrukturen

```
> ansible all -m setup

"ansible_eth0": {
            "ipv4": {
                "address": "192.168.10.14",
                "broadcast": "192.168.10.15",
                "netmask": "255.255.255.240",
                "network": "192.168.10.0"
            },
            "ipv6": [
                {
                    "address": "fe80::1a:46ff:fe47:30c1",
        …
        },
```

```
{{ ansible_eth0["ipv4"]["address"] }}
{{ ansible_eth0.ipv4.address }}
```

Variablen - Komplexe Datenstrukturen

```
> ansible all -m setup

"ansible_interfaces": [
            "lo",
            "eth0"
        ],
```

```
{{ ansible_interfaces[0] }}
```

Variablen - Magie

Variablen, die keine Facts sind und vom User nicht definiert wurden:

hostvars

group_names

groups

`inventory_hostname` **manchmal besser nutzen als** `ansible_hostname`

Variablen - Externe Dateien

Externe Dateien mit Variablen einbinden:

```
- hosts: all
  remote_user: root
  vars:
    favcolor: blue
  vars_files:
    - /vars/external_vars.yml
```

/vars/external_vars.yml

```
---
somevar: somevalue
password: magic
```

Variablen - Bei der Ausführung

Generalisierte Playbooks - bei der Ausführung Variablen nutzen

```
ansible-playbook release.yml --extra-vars "version=1.23.45 env=stage"
```

Variablen - An welchen Ort?

Wir haben viele Möglichkeiten gesehen, variablen zu platzieren:

Variablen am richtigen Ort platzieren - nicht explizit Überschreibungen nutzen

Keep it simple!

Idee: Je expliziter die Angabe - desto höher die Priorität der Variable

Variablen - An welchen Ort?

role defaults < inventory group_vars/all < playbook group_vars/all <

inventory group_vars/* < playbook group_vars/* < inventory host_vars <

playbook host_vars < host facts < play vars < play vars_files <

roles vars < block vars < task vars < role params < include params <

set_facts < extra vars

Variablen - Templating

jinja2

```
{% if 'webserver' in group_names %}
   # some part of a configuration file that only applies to webservers
{% endif %}



{% for host in groups['app_servers'] %}
   # something that applies to all app servers.
{% endfor %}



{% for host in groups['app_servers'] %}
   {{ hostvars[host]['ansible_eth0']['ipv4']['address'] }}
{% endfor %}
```

Variablen - Loops

Mit Loop

```
- name: add several users
  user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  with_items:
     - testuser1
     - testuser2
```

Ohne Loop

```
- name: add user testuser1
  user:
    name: "testuser1"
    state: present
    groups: „wheel“

- name: add user testuser2
  user:
    name: "testuser2"
    state: present
    groups: "wheel"
```

Variablen - Loops

Simple

```
- name: add several users
  user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  with_items:
     - testuser1
     - testuser2
```

Nested

```
- name: give users access to multiple databases
  mysql_user:
    name: "{{ item[0] }}"
    priv: "{{ item[1] }}.*:ALL"
    append_privs: yes
    password: "foo"
  with_nested:
    - [ 'alice', 'bob' ]
    - [ 'clientdb', 'employeedb', 'providerdb' ]
```

Variablen - Loops

Weitere Loops:

with_files

```
with_file:
  - first_example_file
  - second_example_file
```

Variablen - Loops

Weitere Loops:

with_files

with_fileglobs

```
with_fileglob:
  - "/playbooks/files/fooapp/*"
```

Variablen - Loops

Weitere Loops:

with_files

with_fileglobs

with_together

```
---
alpha: [ 'a', 'b', 'c', 'd' ]
numbers:  [ 1, 2, 3, 4 ]


with_together:
  - "{{ alpha }}"
  - "{{ numbers }}"
```

Variablen - Loops

Weitere Loops:

with_files

with_fileglobs

with_together

with_subelements

```
with_subelements:
  - "{{ users }}"
  - "{{ mysql.hosts }}"
```

Variablen - Loops

Weitere Loops:

with_files

with_fileglobs

with_together

with_subelements

with_inventory_hostnames

```
with_inventory_hostnames:
  - all:!www
```

https://docs.ansible.com/ansible/playbooks_loops.html

Variablen - Blocks

```
tasks:
     - block:
         - yum: name={{ item }} state=installed
           with_items:
             - httpd
             - memcached

         - template: src=templates/src.j2 dest=/etc/foo.conf

         - service: name=bar state=started enabled=True

       when: ansible_distribution == 'CentOS'
       become: true
       become_user: root
```

Variablen - Block error handling

```
tasks:
 - block:
     - debug: msg='i execute normally'
     - command: /bin/false
     - debug: msg='i never execute, cause ERROR!'
   rescue:
     - debug: msg='I caught an error'
     - command: /bin/false
     - debug: msg='I also never execute :-('
   always:
     - debug: msg="this always executes"
```

Variablen - Block error handling

```
tasks:
 - block:
     - debug: msg='i execute normally'
       notify: run me even after an error
     - command: /bin/false
   rescue:
     - name: make sure all handlers run
       meta: flush_handlers
handlers:
  - name: run me even after an error
    debug: msg='this handler runs even on error'
```