

Ansible_

Rollen



Recap

Konfigurationsmanagement

Konfiguration von Systemen - Infrastructure as Code

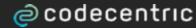
Verteilung und Orchestrierung von Software

Orchestrierung von Systemen

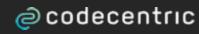
Zero-Downtime Updates

Ad-hoc Kommandos

Operation am offenen Herzen









Grundidee für Rollen sind Includes - Plays einbinden aus externen Playbooks

```
    include: Basissetup.yml Play Include
    name: Installiere Anwendung
    hosts: all
    tasks:

            include: Erstelle_User.yml
                 Task Include
                  include: Installiere_Anwendung.yml
```





Wiederverwendung

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



Loops

```
- include: foo.yml param={{item}}
  with_items:
  - 1
  - 2
  - 3
```

S



Includes - Static vs. Dynamic

Wann werden die includes verarbeitet?

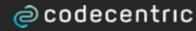
Dynamic (Standard): Verarbeitung bei der Ausführung

- Kein Trigger Handler
- Kein —start-at-task + Aber Variablen
- Keine tags

Static: Durch Pre-Prozessor

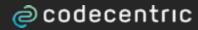
- include: "stuff.yml"
 static: no

when: verto is defined





Rollen





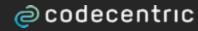
Rollen - Warum?

Kapselung

Nicht: "Wende diese Tasks auf diese(n) Host(s) an"

Sondern: "Diese(r) Host(s) ist ein Webserver"

Sehe das große Ganze, nicht direkt die Details





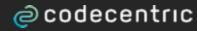
Rollen

Automatisches Laden von

Dateien, Variablen, Tasks ...

basierend auf fixer Ordnerstruktur.

Rollen sind Automatisierung von Includes



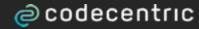


Rollen - Dateistruktur

```
site.yml
webservers.yml
fooservers.yml
roles/
   common/
     files/
     templates/
     tasks/
     handlers/
     vars/
     defaults/
     meta/
   webservers/
     files/
      •••
```

Aufruf im Playbook

```
---
- hosts: webservers
roles:
- common
- webservers
```





Rollen - Dateistruktur

files Statische Dateien

templates Dynamische Dateien

tasks Tasks der Rolle

handlers Handler der Rolle

vars Variablen der Rolle

defaults Default Variablen

meta Tasks, die den internen State beeinflussen



Rollen - Ausführung

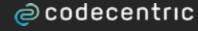
Dateien zum Play hinzufügen, wenn /roles/x/\$1/main.yml existiert

\$1: tasks, handlers, vars, defaults, meta

Innerhalb einer Rolle direkte Referenzen auf files, templates und tasks

Nicht alle Ordner müssen vorhanden sein

Weiterhin "lose" Tasks in Playbooks möglich - Rollen haben aber Priorität



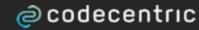


Rollen - Ausführung

```
- hosts: webservers
 pre tasks:
    - shell: echo 'hello'
 tasks:
    - shell: echo 'still busy'
 roles:
    - some role
 post_tasks:
    - shell: echo 'goodbye'
```

Rollen - Parametrisierung

```
---
- hosts: webservers
  roles:
    - common
    - { role: foo_app_instance, dir: '/opt/a', app_port: 5000 }
    - { role: foo_app_instance, dir: '/opt/b', app_port: 5001 }
```





Rollen - Abhängigkeiten

Automatisch Inhalt weiterer Rollen hinzufügen

Hinterlegt in meta/main.yml

```
dependencies:
   - { role: common, some_parameter: 3 }
   - { role: apache, apache_port: 80 }

---
dependencies:
   - { role: '/path/to/common/roles/foo', x: 1 }
```



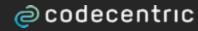


Rollen - Abhängigkeiten

Werden immer vor der eigentlichen Rolle ausgeführt

Rekursive Einbindung

Generell Ausführung nur 1x - außer: allow_duplicates: yes





Rollen - Abhängigkeiten

/roles/car/main/dependencies.yml

```
dependencies:
- { role: wheel, n: 1 }
- { role: wheel, n: 2 }
- { role: wheel, n: 3 }
- { role: wheel, n: 4 }
```



/roles/wheel/main/dependencies.yml

```
allow_duplicates: yes
dependencies:
  - { role: tire }
S - { role: brake }
```

```
tire(n=1)
brake(n=1)
wheel (n=1)
tire(n=2)
brake(n=2)
wheel (n=2)
car
```



Hands-On: Apache2 Webserver

Ziel: Apache2 Webserver installieren (2x)

Notwendige Schritte

- Apache2 Webserver installieren
- Index-Datei hinzufügen

