

Next Level Ansible

Grazer Linuxtage 2019



Manuel Bonk

27. April 2019

- ▶ consultant at ATIX AG
- ▶ mainly Ansible, Saltstack, Kafka, Foreman
- ▶ training Ansible
- ▶ contributing to the Foreman Project (foreman-ansible-modules, nailgun)

- ▶ configuration management and orchestration tool
- ▶ masterless and daemonless
- ▶ idempotence
- ▶ minimal node requirements: sshd and python
- ▶ YAML
- ▶ jinja2 templating engine

- ▶ ansible-pull with cron module
- ▶ JUnit Callback Plugin
- ▶ omit
- ▶ merge dicts with default values
- ▶ dict rotation

scenario:

- ▶ automatically reset a system every x minutes
- ▶ no execution node
- ▶ →puppet-like behaviour in an ansible environment

```
- name: deploy cronjob
  cron:
    name: "run latest playbook"
    minute: "*/30"
    job: >
      ansible-pull
      -U https://github.com/manuelbonk/ansible-pull.git
      -i 'localhost,'
      play.yaml"
```

...

scenario:

- ▶ ansible executed via jenkins pipeline
- ▶ by default no verbose feedback if ansible failed or not

scenario:

- ▶ ansible executed via jenkins pipeline
- ▶ by default no verbose feedback if ansible failed or not

solution:

- ▶ JUnit callback plugin →playbook output as a JUnit test case result file
- ▶ file can be visualized in Jenkins

ansible.cfg

```
[defaults]
```

```
callback_whitelist = junit
```

play-1541222639.388318.ml

```
<?xml version="1.0" ?>
<testsuites disabled="0" errors="0" failures="0" tests="1" time=
  <testsuite disabled="0" errors="0" failures="0" name="play" sk
    <testcase classname="/home/m/Documents/vorträge/next_level_a
      <system-out>{
        &quot;changed&quot;: false,
        &quot;msg&quot;: &quot;Hello world!&quot;;
      }</system-out>
    </testcase>
  </testsuite>
</testsuites>
```

scenario:

- ▶ developing a new module
- ▶ testing task with different parameters (using `include_tasks`)

tasks/organization.yaml

```
- foreman_organization:
  name: "{{ organization_name }}"
  label: "{{ organization_label | default(omit) }}"
  state: "{{ organization_state }}"
```

...

playbook.yaml:

```
- include: tasks/organization.yaml
  vars:
    organization_state: present
- include: tasks/organization.yaml
  vars:
    organization_state: present
    organization_label: "my_new_label"
```

scenario:

- ▶ lots of similar configuration files for microservices
- ▶ a lot of them only differ 'name' and 'image' value
- ▶ one dict with default settings
- ▶ only non-default values have to be written down

```
vars:
  default_server:
    kernel_memory: 64m
    cpu_count: 2
    state: started

  a_new_server:
    name: "a new server"
    kernel_mempory: 32m
tasks:
  - set_fact:
      server: "{{ default_server |
                combine(a_new_server, recursive=True) }}"
```

from this...

```
product_order_list:  
- name: product1  
  customers:  
    - customer_a  
    - customer_b  
- name: product2  
  customers:  
    - customer_b  
    - customer_c
```

...into this

```
customer_order_dict:  
  customer_a:  
    - product1  
  customer_b:  
    - product1  
    - product2  
  customer_c:  
    - product2
```

```
vars:
  product_order_list:
    - name: product1
      customers:
        - customer_a
        - customer_b
    - ...
- set_fact:
  customer_order_list: "{{ customer_order_list
    | default({})
    | combine( {item.1:
      customer_order_list[item.1]
    | default([])
    | union([item.0.name] ) }) }}"

with_subelements:
  - "{{ product_order_list }}"
  - customers
```