Next Level Ansible

Open Rhein Ruhr 2018

Manuel Bonk

3. November 2018

Manuel Bonk

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

Ansible – a brief overview

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

tricks – overview

- ansible-pull with cron module
- JUnit Callback Plugin
- omit
- tags behavior during import_tasks vs. include_tasks
- zip_longest
- merge dicts with default values
- dict rotation

tricks – ansible-pull

scenario:

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

tricks – ansible-pull

```
- 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"
...
```

tricks – JUnit callback

scenario:

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

tricks – JUnit callback

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

tricks – JUnit callback

ansible.cfg

```
[defaults]
callback_whitelist = junit
```

tricks – JUnit callback

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>
```

tricks – omit

scenario:

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

tricks – omit

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"
```

tricks – import_tasks vs. include_tasks

scenario:

- tags are used to execute a subset of tasks
- multiple tags in tasks file

option 1: run a subset of included tasks
option 2: run all tasks and ignoring set tags

- tags are used to execute a subset of tasks
- import_tasks is rendered statically →tags of imported tasks are ignored
- include_tasks is rendered dynamically →tags of included tasks are checked

tricks – import_tasks vs. include_tasks

imported.yaml

```
---
- debug:
    msg: i am an imported tag1
  tags:
   - tag1

- debug:
    msg: i am an imported tag2
  tags:
   - tag2
...
```

included.yaml

```
---
- debug:
    msg: i am an included tag1
  tags:
   - tag1

- debug:
    msg: i am an included tag2
  tags:
   - tag2
...
```

tricks – import_tasks vs. include_tasks

```yaml
---
- hosts: localhost
  connection: local
  gather_facts: False
  tasks:
    - include_tasks: included.yaml
      tags:
       - tag1
    - import_tasks: imported.yaml
      tags:
       - tag1
...
```

tricks – zip_longest

scenario:

- compiling a string for ‘command‘ module
- build cli parameter strings

zip_longest:

- zipping lists together
- use ‘fillvalue‘ for lists of unequal lengths

tricks – zip_longest

```
tasks:
  - debug:
      msg: "{{ list1
               | zip_longest( list2, fillvalue='fill')
               | map('join')
               |list  }}"
  - debug:
      msg: "{{ list1
               | zip_longest(list2, list3, fillvalue='fill')
               | map('join')
               |join (' ') }}"
  - debug:
      msg: "{{ []
               | zip_longest(provider,
                   fillvalue='--enable-foreman-compute-')
               | map('join')
               |join (' ') }}"
...
```

tricks – merge dicts with defaults

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 vaules have to be written down

tricks – merge dicts with defaults

```
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)}}"
```

tricks – dict rotation

from this…

```
product_oder_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
```

tricks – dict rotation

```
  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
```