# Ansible, openSUSE Ansible packaging, collections and ansible-lint

openSUSE Conference 2023

May 26, 2023


Johannes Kastl
Linux Trainer & Consultant
B1 Systems GmbH
kastl@b1-systems.de

# Agenda

- introduction
- Ansible
- roles, collections and ansible-core
- packaging Ansible for openSUSE
- linting Ansible code with `ansible-lint`
- Ansible tools

# About me

- Johannes Kastl
- he/him
- Linux Trainer and Consultant at B1 Systems
- OBS: `ojkastl_buildservice`
- Mastodon: `@johanneskastl@digitalcourage.social`
- GitHub/GitLab/Codeberg/...: `johanneskastl`

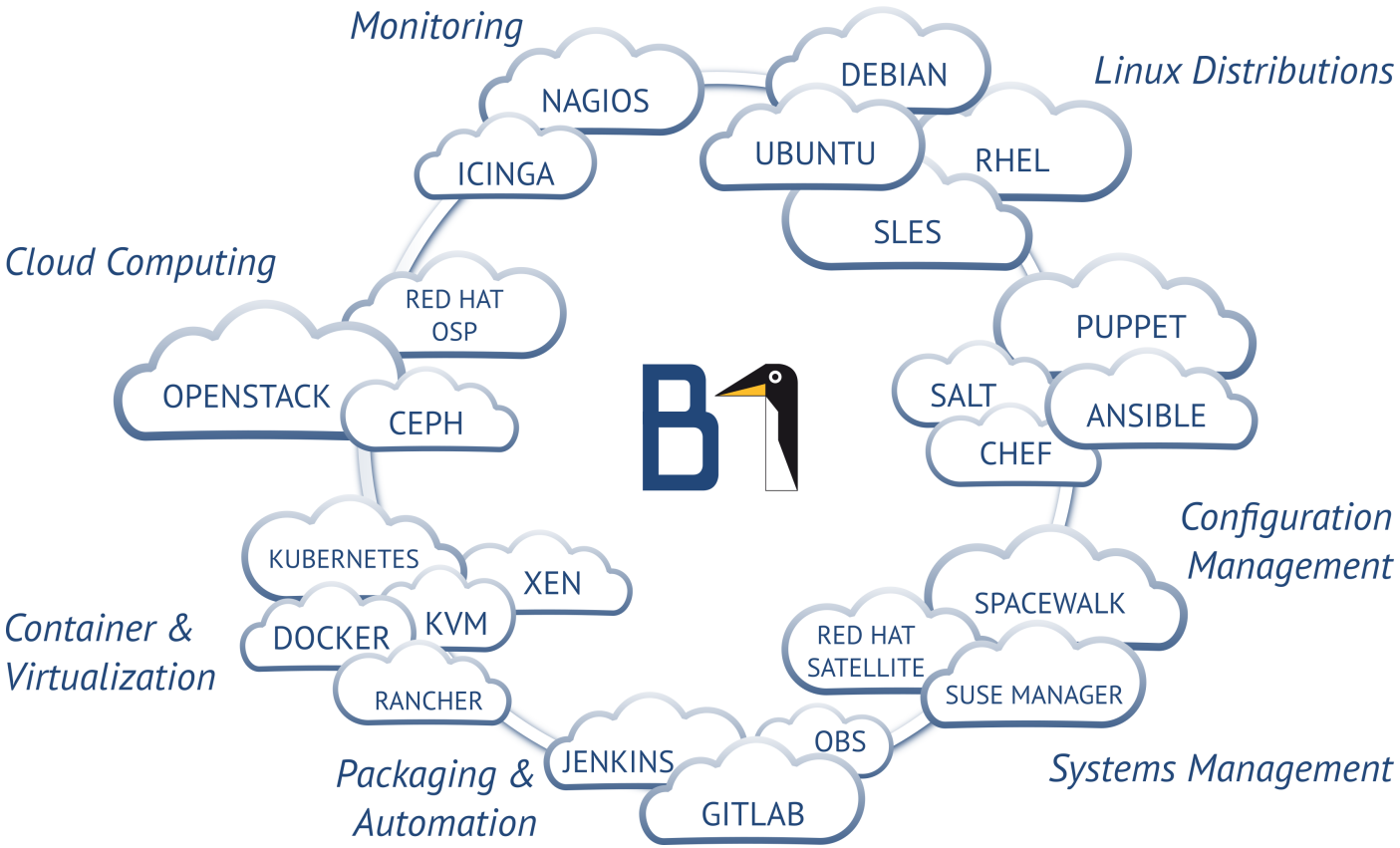
# What I do

- trainer, consultant, systems administrator, architect, spielkind
- configuration management (Ansible, Puppet, Chef, SaltStack)
- Infrastructure as Code (Terraform, . . . )
- Kubernetes and Containers (Podman, Docker, LXC, . . . )
- CI/CD (Jenkins, GitLab CI, GitHub Actions)
- agile and lean (Scrum, Kanban)
- maintainer for Ansible and some cloud-native tools

# Introducing B1 Systems

- founded in 2004
- operating both nationally & internationally
- more than 140 employees
- vendor-independent (hardware & software)
- focus:
  - consulting
  - support
  - training
  - managed service & operations
  - solutions & development
- branch offices in Rockolding, Berlin, Cologne, Dresden & Jena

# Areas of Expertise



# Ansible

- configuration management system (like Puppet, Chef, SaltStack)
- no central server (unlike Puppet, Chef, SaltStack)
- no agent required
  - SSH access needed to the target
  - Python3 needed on the target
- can manage servers, appliances and applications
  - PostgreSQL, MySQL, Grafana, NetBox, ...
  - AWS, Azure, GCP, OpenStack, Kubernetes, Podman, ...
  - Arista, Cisco, Juniper, F5, ...

# Ansible usage

- define your hosts and groups in an inventory file ("`inventory`")
- create a playbook ("`playbook.yml`")
- run the playbook

```
$ ansible-playbook -i inventory playbook.yml
```

# Ansible inventory example

## Inventory

```
localhost
```

# Ansible inventory example with groups

## Inventory

```
[webservers]
webserver-01
webserver-02
webserver-03

[databases]
database.example.com
```

# Ansible playbook example

## Playbook

```
---
- name: 'Say hello to osc23'
  hosts: localhost
  gather_facts: false

  tasks:
    - name: 'Output something'
      ansible.builtin.debug:
        msg: 'Hello openSUSE conference 2023!'
```

# Running Ansible

```
$ ansible-playbook -i inventory playbook.yml
PLAY [Say hello to osc23] ******************

TASK [Output something] ********************
ok: [localhost] => {
    "msg": "Hello openSUSE conference 2023!"
}

PLAY RECAP *********************************
localhost  : ok=1 changed=0 unreachable=0 failed=0
             skipped=0 rescued=0 ignored=0
```

# Ansible playbooks and DRY

- playbooks contain tasks
- multiple playbooks might contain the same code
- playbooks can contain multiple plays
- . . . but then they get reaaaally long

=> solution: roles

# Ansible roles

- reusable fragments of code
- do one thing and do it well
- can set variables needed for the role to work
- can be controlled by setting variables in the playbook
- distributed in the Ansible Galaxy
- mostly not used to distribute Ansible modules

# Ansible playbook using roles

## Playbook

```
- name: 'Configure the local user osc23'
  hosts: localhost

  roles:
    - role: 'johanneskastl.minimal_bashrc'
      bashrc_for_root: false
      additional_users:
        - osc23
    - role: 'johanneskastl.create_a_vimrc'
      vimrc_for_root: false
      additional_users:
        - osc23
```

# Ansible packaging up to Ansible 2.9

- all of the Ansible core functionality
  - Ansible executables
  - modules like template, service, debug, package, ...
- long list of selected modules
- packaged together as the “ansible” Python module

# Issues with the old way of packaging

- development of the core functionality
- development of modules
- too many modules
- releasing required getting everything in shape
- long release cycles
- fixes in modules took too long

# Introduction of ansible-core and collections

- split Ansible core functionality and modules
- enter: collections
- collections can contain
  - roles
  - playbooks
  - modules
  - plugins
- independent release cycles
- more modularity

# What's in the "ansible-core" package?

- Ansible core functionality
- Ansible core executables
  - `ansible`
  - `ansible-playbook`
  - `ansible-vault`
  - `ansible-inventory`
  - ...
- Ansible builtin modules
  - template
  - debug
  - package
  - service
  - ...
- built using the default Python3 version

# What's in the "ansible" package?

- curated list of community collections
- `/usr/lib/python3.10/site-packages/ansible_collections/`
- currently 51 "namespaces"
- `ansible-community` executable

# What to use?

- you always need `ansible-core`
- for collections (included in the `ansible` package):
  - install the `ansible` package
  - install the collections manually
- install other collections manually

```
$ ansible-galaxy collection install \
    my_namespace.my_collection
```

# Ansible for SLES15?

- Ansible needs Python3.9 or higher
- SLES15 has Python3.6
- SLES15 has a Python3 module with python3.10
  ...but only a small number of modules
- since May: new macro for building with python3.11
- home:ojkastl_buildservice:Branch_systemsmanagement_ansible

# Linting Ansible code with `ansible-lint`

- code written by humans has errors
- simple syntax check by running:

```
$ ansible-playbook -i inventory \
    --syntax-check playbook.yml
```

- better: `ansible-lint`
  - checks Ansible syntax
  - checks YAML syntax
  - checks community guidelines
  - warns of “no-gos”
  - available in openSUSE and as e. g. a GitHub Action

# ansible-lint usage and output

```
$ ansible-lint --profile=production playbook.yml
Passed with production profile:
0 failure(s), 0 warning(s) on 1 files.
$
```

# ansible-lint usage and output

```
$ ansible-lint --profile=production playbook.yml
WARNING  Listing 1 violation(s) that are fatal
fqcn[action-core]: Use FQCN for builtin module
actions (debug).
playbook.yml:7 Use `ansible.builtin.debug` or
`ansible.legacy.debug` instead.

Rule Violation Summary
count tag                  profile    rule associated tags
    1 fqcn[action-core] production formatting

Failed after shared profile, 4/5 star rating:
1 failure(s), 0 warning(s) on 1 files.
```

# ansible-builder

- tasks or roles might have collections as dependencies
- collections might have Python modules as dependencies
- you might want to pin versions of those dependencies
- involves multiple tools (RPM, pip, ansible-galaxy, . . . )
- Python virtual environments not easily distributable
- ⇒ run Ansible inside a container
- build the container image using `ansible-builder`

# ansible-runner

- `ansible-runner` lets you run Ansible inside a container easily
- automate execution of Ansible and consume the results
- big part of AWX/AAP automation (previously: Python virtual envs)

# `ansible-sign`

- using Ansible roles and collections is easy
- versioning of roles and collections is easy
- until now no cryptographic verification
- $\Rightarrow$ `ansible-sign` can sign Ansible content using GPG keys
- signed content usable in AWX/AAP

# AWX

- web-based user interface, REST API, and task engine
- central location for running Ansible (think: auditing, logs)
- RBAC
- runs in Kubernetes/OpenShift

# molecule

- tool for testing Ansible code
- can use podman
- testing roles for
  - multiple operating systems
  - multiple OS versions
  - virtualization providers
  - ...
- verification using
  - Ansible (run twice and look for changes)
  - Ansible (with `assert`)
  - the `testinfra` framework

Thank you for your attention!
Questions?

For more information, refer to info@b1-systems.de
or +49 (0)8457 - 931096

# Thank You!