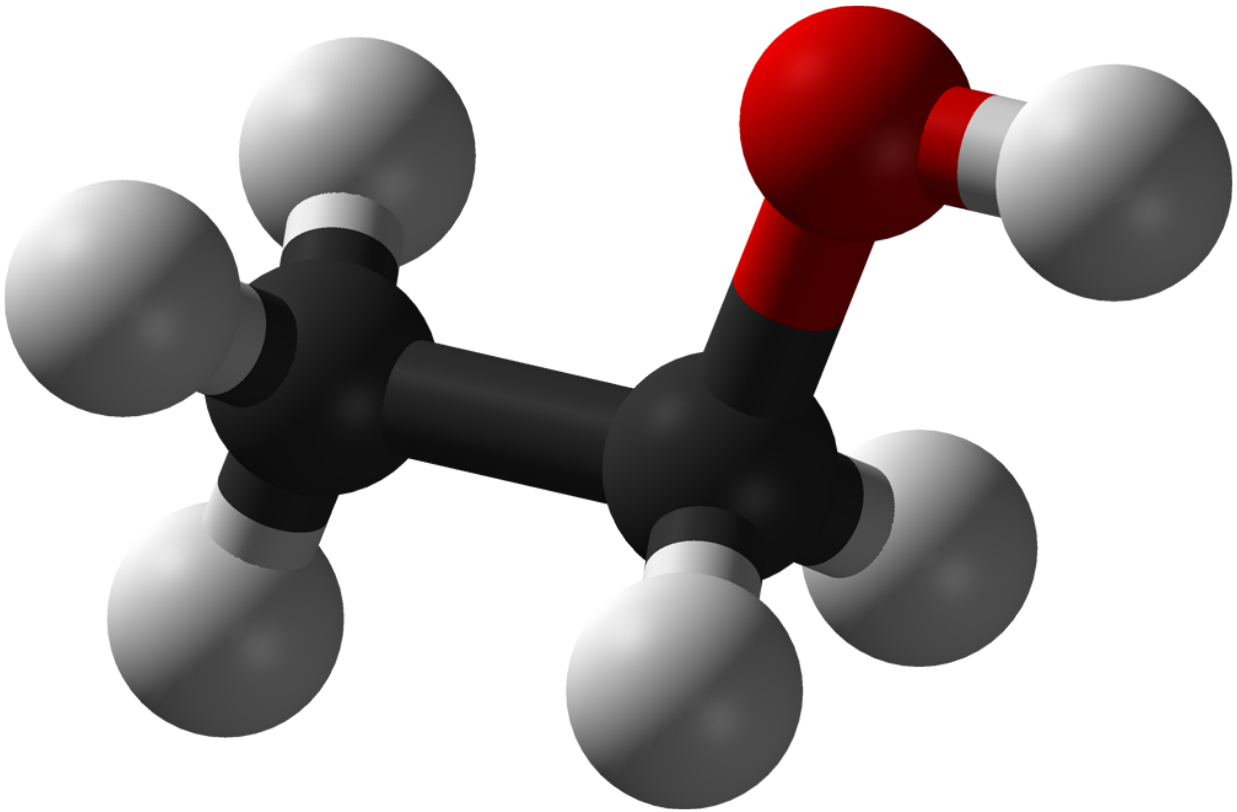
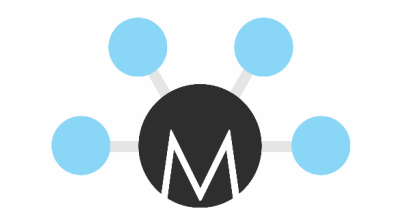


Probing Ansible Bonds with Molecule Tests



Matthias Dellweg & Bernhard Hopfenmüller

4. Februar 2020

whoarewe

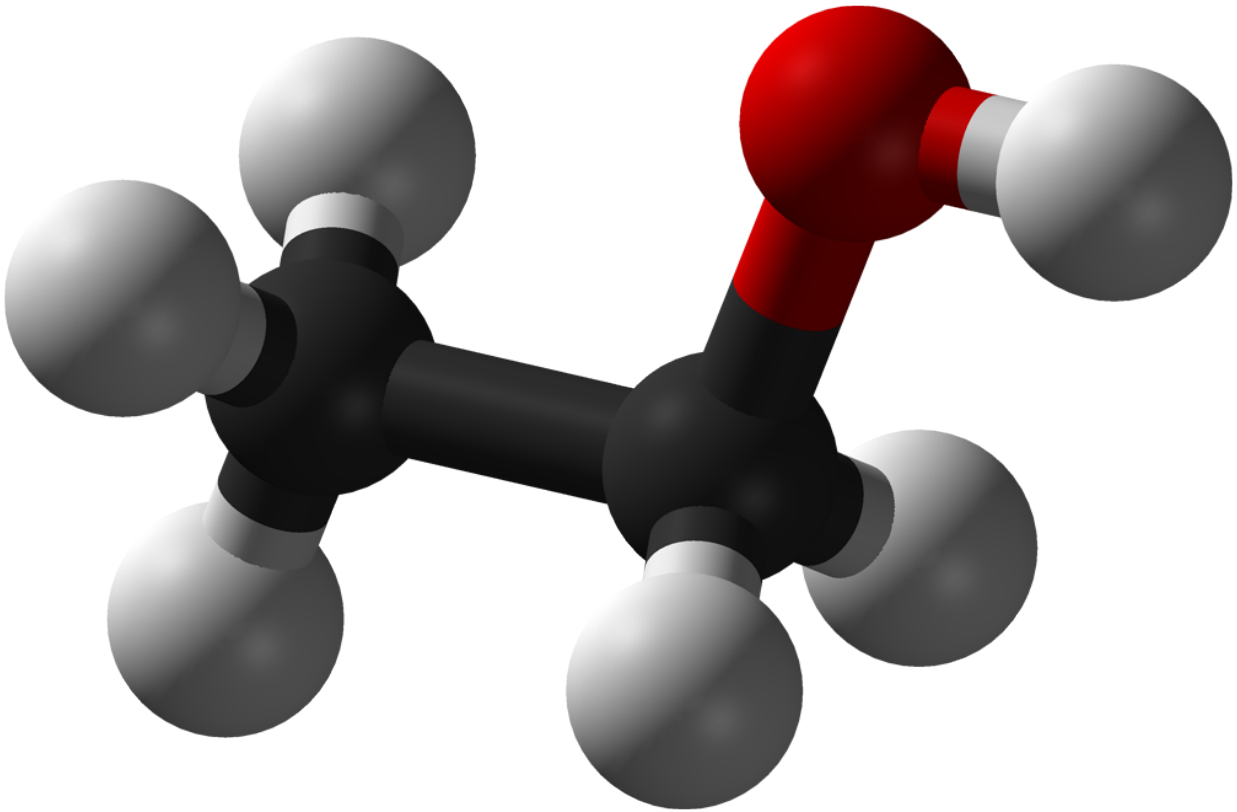
- Matthias Dellweg
- Physicist
- Senior Software Engineer
- Foreman, Pulp, Ansible (FAM), …
- Github: mdellweg, dellweg@atix.de, IRC: x9c4

- Bernhard Hopfenmüller
- Physicist
- Senior IT Consultant
- Ansible (FAM), Kafka, “DevOps”, …
- Github/Twitter/IRC: Fobhep, hopfenmueller@atix.de

whoarewe

- Matthias Dellweg
- Physicist
- Senior Software Engineer
- Foreman, Pulp, Ansible (FAM), …
- Github: mdellweg, dellweg@atix.de, IRC: x9c4

- Bernhard Hopfenmüller
- Physicist
- Senior IT Consultant
- Ansible (FAM), Kafka, “DevOps”, …
- Github/Twitter/IRC: Fobhep, hopfenmueller@atix.de



Test Everything

Chemical Substance

- Does my chemical plant work as specified?
- Can I optimize without breaking?
- Can I produce additional substances in the same plant?

Ansible role

- Does my system work as specified?
- Can I refactor without breaking?
- Can I develop further features with backwards compatibility?

Test Everything

Chemical Substance

- Does my chemical plant work as specified?
- Can I optimize without breaking?
- Can I produce additional substances in the same plant?

Ansible role

- Does my system work as specified?
- Can I refactor without breaking?
- Can I develop further features with backwards compatibility?

Test Everything

Chemical Substance

- Does my chemical plant work as specified?
- Can I optimize without breaking?
- Can I produce additional substances in the same plant?

Ansible role

- Does my system work as specified?
- Can I refactor without breaking?
- Can I develop further features with backwards compatibility?

Test Everything

Chemical Substance

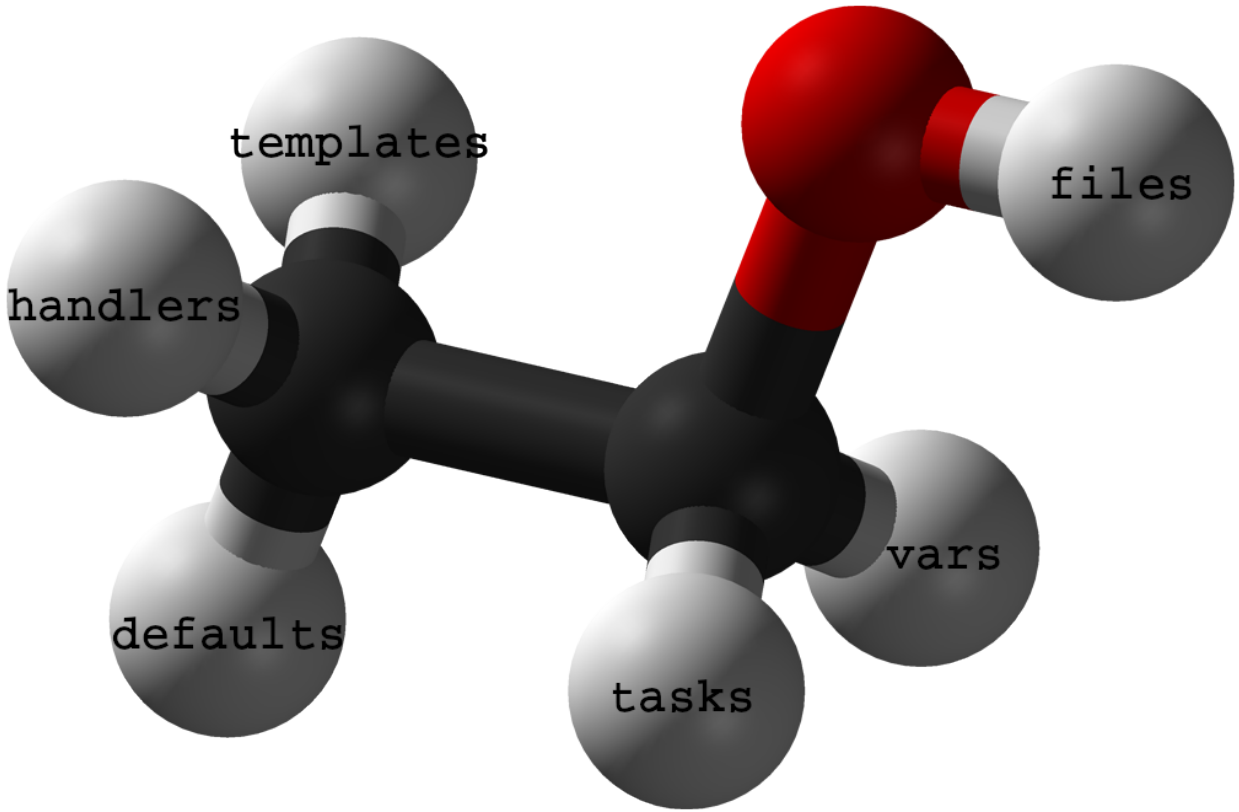
- Does my chemical plant work as specified?
- Can I optimize without breaking?
- Can I produce additional substances in the same plant?

Ansible role

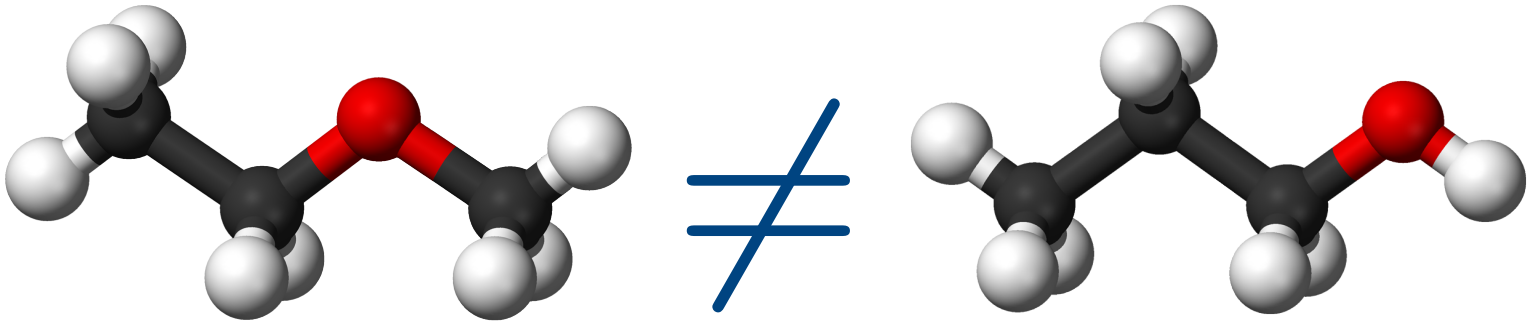
- Does my system work as specified?
- Can I refactor without breaking?
- Can I develop further features with backwards compatibility?

Building Blocks

Modules are already tested. Why should i test roles?

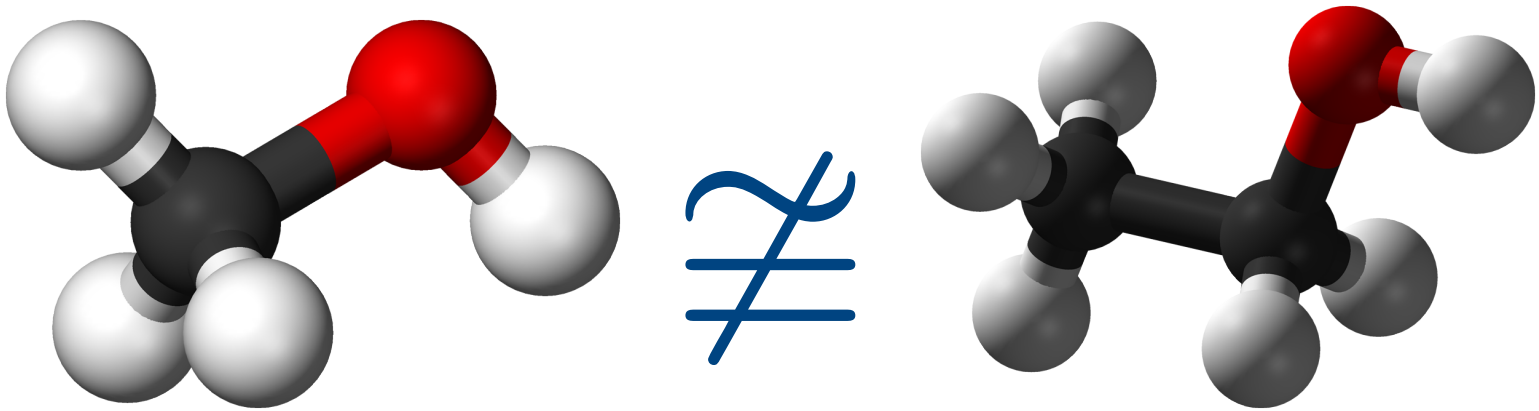


More than just a Bunch of Atoms



ordering matters

More than just a Bunch of Atoms



mind the additional side effects!

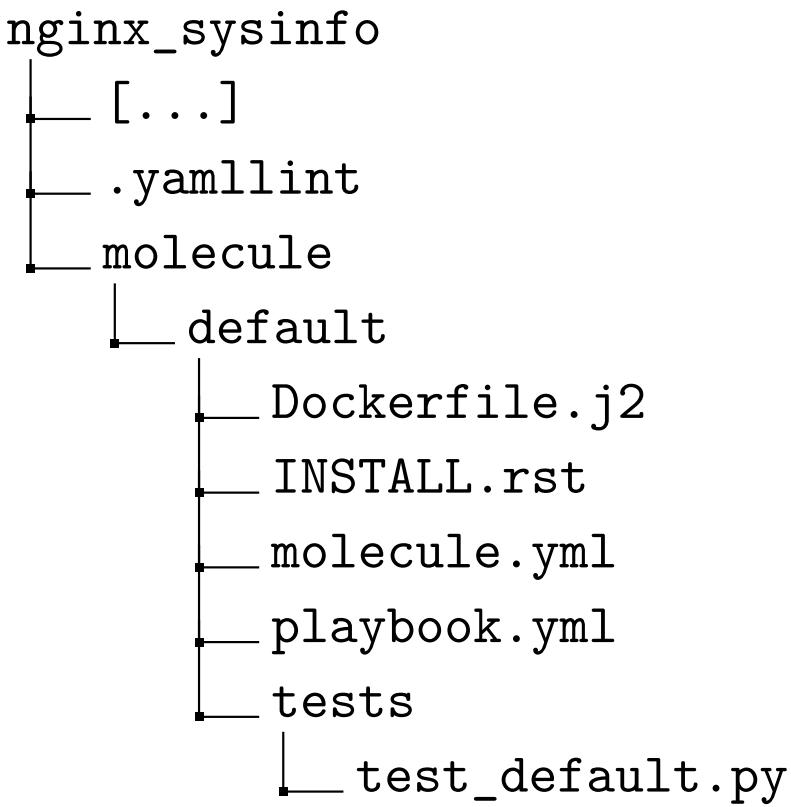
A Role

```
nginx_sysinfo
├── defaults
│   └── main.yaml
├── files
│   └── stylsheet.css
├── handlers
│   └── main.yaml
├── tasks
│   └── main.yaml
├── templates
│   └── index.html.j2
└── vars
    └── main.yaml
```

Bootstrap

```
$mkvirtualenv --python=python3.6 molecule
(molecule)$ pip install molecule
(molecule)$ molecule init --scenario
```

A Role


nginx_sysinfo
[...]
.yamllint
molecule
default
Dockerfile.j2
INSTALL.rst
molecule.yml
playbook.yml
tests
test_default.py

Molecule Config File

```
# config file for tests
# molecule/default/molecule.yml
driver:
  name: docker
platforms:
  - name: centos7
    image: centos:7
  - name: debian10
    image: debian:10
```

Test Matrix

```
(molecule)$ molecule matrix [-s default] test
default
   lint
   dependency
   destroy
   create
   prepare
   converge
   idempotence
   verify
   destroy
```

Linting

```
---
# molecule/default/molecule.yml
driver:
  name: docker
lint:
  name: yamllint
platforms:
  - name: centos7
    image: centos:7
  - name: debian10
    image: debian:10
```

yaml ≠ yaml

```yaml
# some yaml file of the role
# we want this to fail
foo: bar
foo2: bar
foo3:
- bar1
- bar2
foo4:
  - de
  - no
foo5: False
foo6: Yes
```

yaml ≠ yaml

```yaml
--- # this looks better
foo: bar
foo2: bar
foo3:
  - bar1
  - bar2
foo4:
  - de
  - no
foo5: false
foo6: true
```

Linting Taste

edited .yamllint in role dir (added my molecule)

```
rules:
  document-start: enable
  indentation: {spaces: 2, indent-sequences: consistent}
  truthy: enable
  ignore: |
    .gitlab-ci.yml
```

Test Matrix

```
(molecule)$ molecule matrix [-s default] test
default
   lint # ✓
   dependency
   destroy
   create
   prepare
   converge
   idempotence
   verify
   destroy
```

Does my Role have Dependencies?

```
---
# molecule/default/molecule.yml
dependency:
  name: galaxy
driver:
  name: docker
lint:
  name: yamllint
platforms:
  - name: centos7
    image: centos:7
  - name: debian10
    image: debian:10
```

Requirements File

```
---
# molecule/default/requirements.yml
- src: geerlingguy.*
```

Get Dependencies

```
(molecule)$ molecule dependency
--> Test matrix

 default
    dependency

--> Scenario: 'default'
--> Action: 'dependency'
   - changing role geerlingguy.ntp from 1.6.4 to unspecified
   - downloading role 'ntp', owned by geerlingguy
   - downloading role from https://github.com/geerlingguy/[...]
   - extracting geerlingguy.ntp [...]
   - geerlingguy.ntp (1.6.4) was installed successfully
Dependency completed successfully.
```

Alternatives to Galaxy?

- gilt - A GIT layering tool.

```
---
dependency:
  name: gilt
---
# gilt config file
- git: https://github.com/blueboxgroup/ursula.git
  version: master
  files:
    - src: roles/logging
      dst: roles/blueboxgroup.logging/
```

- shell - An Unix tool

```
---
dependency:
  name: shell
  command: curl | bash
```

Use Dependencies?

Patience please

Test Matrix

```
(molecule)$ matrix [-s default] test
default
   lint # ✓
   dependency # ✓
   destroy
   create
   prepare
   converge
   idempotence
   verify
   destroy
```

Create, Converge, Destroy, Prepare

The Ansible provisioner

```
---
# molecule/default/molecule.yml
[...]
provisioner:
  name: ansible
  lint:
    name: ansible-lint
  options:
    vvv: true
  playbooks:
    create: create.yml
    converge: playbook.yml
    destroy: destroy.yml
    prepare: prepare.yml
```

Create Matrix

Prepare infrastructure to run role

```
(molecule)$ molecule matrix [-s default] create
--> Test matrix

 default
    dependency
    create
    prepare
```

Steps for Creating

- Consider adjusting Dockerfile.j2
- Write prepare playbook

```
---
- hosts: all
  #roles:
  #  - geerlingguy.ntp
  tasks:
    - name: Install curl
      become: true
      package:
        name: curl
```

- Done

Docker/Vagrant: create, destroy, prepare are bundled

Converge Matrix

Prepare infrastructure to run role

```
(molecule)$ molecule matrix [-s default] converge

 default
    dependency
    create
    prepare
    converge
```

Converge Playbook

Prepare infrastructure to run role

```
---
- name: Converge
  hosts: all
  roles:
    - role: nginx_sysinfo
```

Converge Playbook

Since playbook names are defaulted

```
---
# molecule/default/molecule.yml
[...]
provisioner:
  name: ansible
  lint:
    name: ansible-lint
  #playbooks:
    #create: create.yml
    #prepare: prepare.yml
```

Test Matrix

```
(molecule)$ molecule matrix [-s default] test
default
   lint # ✓
   dependency # ✓
   destroy # ✓
   create # ✓
   prepare # ✓
   converge # ✓
   idempotence
   verify
   destroy # ✓
```

Idempotence

- Rerun converge
- all tasks unchanged? ✓

Careful: failure is not always a sign of wrong

Remove step from test sequence?

```
---
# molecule/default/molecule.yml
[...]
scenario:
 test_sequence:
   - lint
   - dependency
   - destroy
   - create
   - prepare
   - converge
   #- idempotence
   - verify
   - cleanup
   - destroy
```

Rescue Idempotence

sometimes changes are more equal than others

```
---
- name: apt-get update
  apt:
    update_cache: true
  changed_when: false
```

Test Matrix

```
(molecule)$ molecule matrix [-s default] test
default
   lint # ✓
   dependency # ✓
   destroy # ✓
   create # ✓
   prepare # ✓
   converge # ✓
   idempotence # ✓
   verify
   destroy # ✓
```

Verify

Test the results of your role. Default python testinfra + flake8 linter

```
---
#molecule/default/molecule.yml
[...]
verifier:
 name: testinfra
 lint:
  name: flake8
```

Verify with Testinfra

```
#molecule/default/test/test_default.py
import os
from testinfra.utils import ansible_runner

testinfra_hosts = ansible_runner.AnsibleRunner(
   os.environ['MOLECULE_INVENTORY_FILE']
).get_hosts('all')


def test_nginx_is_installed(host):
   nginx = host.package("nginx")
   assert nginx.is_installed
```

Verify with Ansible

```
---
#molecule/default/molecule.yml
[...]
verifier:
 name: ansible
 lint:
  name: ansible-lint
```

Create verify.yml in default dir

Test Matrix

```
(molecule)$ molecule matrix [-s default] test
default
   lint # ✓
   dependency # ✓
   destroy # ✓
   create # ✓
   prepare # ✓
   converge # ✓
   idempotence # ✓
   verify # ✓
   destroy # ✓
```

Run single steps of test sequence?

Run Molecule Steps

```
$ molecule [--scenario-name default] <sequence_step>
$ molecule lint # lint
$ molecule create # only create infra
$ molecule list # list created infra
$ molecule converge # run role
$ molecule login # connect with instance to debug
$ molecule verify # only run testinfra/ansible tests
$ molecule destroy # destroy infra
$ molecule test [--destroy=never]
# run all of the above (keep infra)
```

Role results

Network information

Hostname	localhost.localdomain
IPv4 addresses	192.168.121.25
IPv6 addresses	fe80::5054:ff:fea2:17c5
Default IPv4 interface -- macaddress	52:54:00:a2:17:c5
Default IPv4 interface -- network	192.168.121.0
Default IPv4 interface -- mtu	1500
Default IPv4 interface -- broadcast	192.168.121.255
Default IPv4 interface -- alias	eth0
Default IPv4 interface -- netmask	255.255.255.0
Default IPv4 interface -- address	192.168.121.25
Default IPv4 interface -- interface	eth0
Default IPv4 interface -- type	ether
Default IPv4 interface -- gateway	192.168.121.1
Hostname	localhost
FQDN	localhost.localdomain

Role results

- What about more info?

Role results

Network information

Hostname	localhost.localdomain
IPv4 addresses	192.168.121.25
IPv6 addresses	fe80::5054:ff:fea2:17c5
Default IPv4 interface -- macaddress	52:54:00:a2:17:c5
Default IPv4 interface -- network	192.168.121.0
Default IPv4 interface -- mtu	1500
Default IPv4 interface -- broadcast	192.168.121.255
Default IPv4 interface -- alias	eth0
Default IPv4 interface -- netmask	255.255.255.0
Default IPv4 interface -- address	192.168.121.25
Default IPv4 interface -- interface	eth0
Default IPv4 interface -- type	ether
Default IPv4 interface -- gateway	192.168.121.1
Hostname	localhost
FQDN	localhost.localdomain

OS Facts

This system is running on	CentOS
Version	7.6
OS Family	RedHat
Used package manager	yum
AppArmor	disabled
Selinux	enabled
Python Version	2.7.5

Environment variables

LANG	en_US.UTF-8
TERM	xterm-256color
SHELL	/bin/bash
XDG_RUNTIME_DIR	/run/user/1000
MAIL	/var/mail/vagrant
SHLVL	2
SSH_TTY	/dev/pts/0
SELINUX_LEVEL_REQUESTED	
SSH_CLIENT	192.168.121.1 55122 22
LESSOPEN	|/usr/bin/lesspipe.sh %s
PWD	/home/vagrant
SELINUX_ROLE_REQUESTED	
SELINUX_USE_CURRENT_RANGE	
LOGNAME	vagrant
USER	vagrant
PATH	/usr/local/bin:/usr/bin
HOME	/home/vagrant
LS_COLORS	rs=0:di=38;5;27:ln=38;5;51:mh=44;38;5;15:pi=40;38;5;11:so=38;5;13:do=38;5;5:bd=48;5;232;38;5;11:cd=48;5;232;38;5;3:or=48;5;232;38;5;9:mi=05;48;5;232;38;5;15:su=48;5;196;38;5;15:sg=48;5;11;38;5;16:ca=48;5;196;38;5;226:tw=48;5;10;38;5;16:ow=48;5;10;38;5;21:st=48;5;21;38;5;15:ex=38;5;34:*.tar=38;5;9:*.tgz=38;5;9:*.arc=38;5;9:*.arj=38;5;9:*.taz=38;5;9:*.lha=38;5;9:*.lz4=38;5;9:*.lzh=38;5;9:*.lzma=38;5;9:*.tlz=38;5;9:*.txz=38;5;9:*.tzo=38;5;9:*.t7z=38;5;9:*.zip=38;5;9:*.z=38;5;9:*.Z=38;5;9:*.dz=38;5;9:*.gz=38;5;9:*.lrz=38;5;9:*.lz=38;5;9:*.lzo=38;5;9:*.xz=38;5;9:*.bz2=38;5;9:*.bz=38;5;9:*.tbz=38;5;9:*.tbz2=38;5;9:*.tz=38;5;9:*.deb=38;5;9:*.rpm=38;5;9:*.jar=38;5;9:*.war=38;5;9:*.ear=38;5;9:*.sar=38;5;9:*.rar=38;5;9:*.alz=38;5;9:*.ace=38;5;9:*.zoo=38;5;9:*.cpio=38;5;9:*.7z=38;5;9:*.rz=38;5;9:*.cab=38;5;9:*.jpg=38;5;13:*.jpeg=38;5;13:*.gif=38;5;13:*.bmp=38;5;13:*.pbm=38;5;13:*.pgm=38;5;13:*.ppm=38;5;13:*.tga=38;5;13:*.xbm=38;5;13:*.xpm=38;5;13:*.tif=38;5;13:*.tiff=38;5;13:*.png=38;5;13:*.svg=38;5;13:*.svgz=38;5;13:*.mng=38;5;13:*.pcx=38;5;13:*.mov=38;5;13:*.mpg=38;5;13:*.mpeg=38;5;13:*.m2v=38;5;13:*.mkv=38;5;13:*.webm=38;5;13:*.ogm=38;5;13:*.mp4=38;5;13:*.m4v=38;5;13:*.mp4v=38;5;13:*.vob=38;5;13:*.qt=38;5;13:*.nuv=38;5;13:*.wmv=38;5;13:*.asf=38;5;13:*.rm=38;5;13:*.rmvb=38;5;13:*.flc=38;5;13:*.avi=38;5;13:*.fli=38;5;13:*.flv=38;5;13:*.gl=38;5;13:*.dl=38;5;13:*.xcf=38;5;13:*.xwd=38;5;13:*.yuv=38;5;13:*.cgm=38;5;13:*.emf=38;5;13:*.axv=38;5;13:*.anx=38;5;13:*.ogv=38;5;13:*.ogx=38;5;13:*.aac=38;5;45:*.au=38;5;45:*.flac=38;5;45:*.mid=38;5;45:*.midi=38;5;45:*.mka=38;5;45:*.mp3=38;5;45:*.mpc=38;5;45:*.ogg=38;5;45:*.ra=38;5;45:*.wav=38;5;45:*.axa=38;5;45:*.oga=38;5;45:*.spx=38;5;45:*.xspf=38;5;45:
XDG_SESSION_ID	6
_	/usr/bin/python
SSH_CONNECTION	192.168.121.1 55122 192.168.121.25 22

Roles are parameterizable

```
{% if full_info %}
<table>
  <tr> <th colspan='2'>OS Facts</th> </tr>
  <tr>
    <td>This system is running on</td>
    <td>{{ ansible_distribution }}</td>
  </tr>
  [...]
</table>
{% endif %}
```

Scenarios test different Situations

Our role supports an option to include more (sensitive) information.

```
(molecule)$ molecule init scenario -s full
```

```
full
├── Dockerfile.j2
├── molecule.yml
├── playbook.yml
└── tests
    └── test_default.py
```

Don’t repeat yourself

Usually only some aspects are different.

```
$ cd full
$ ln -sf ../default/molecule.yml
$ ln -sf ../default/Dockerfile.j2
$ ln -sf ../default/prepare.yml
$ cp ../default/playbook.yml .
```

Converge in Scenario “full”

```
# playbook.yml
---
- name: Converge
  hosts: all
  vars:
    full_info: true  # test with non default option
  roles:
    - role: nginx_sysinfo
```

Verify Scenario “full”

```
$ rm tests/*
$ ln -sf ../default/tests/test_common.py tests/
$ cp ../default/tests/test_content_default.py \
  tests/test_content_full.py
```

```
# tests/test_content_full.py
[...]
def test_nginx_serving_content(host):
    assert host.addr("localhost").port(80).is_reachable
    result = host.check_output("curl localhost:80")
    assert "IPv4 addresses" in result
    assert "AppArmor" in result
    assert "Environment variables" in result
```

Run Scenario Test

```
# Test non default scenario
(molecule)$ molecule test -s full


# Test all scenarios in parallel
(molecule)$ molecule test --all --parallel
```

Use different Driver

```
(molecule)$ pip install 'molecule[hetznercloud]'
(molecule)$ molecule scenario init hetzner_default \
 --driver-name hetznercloud
```

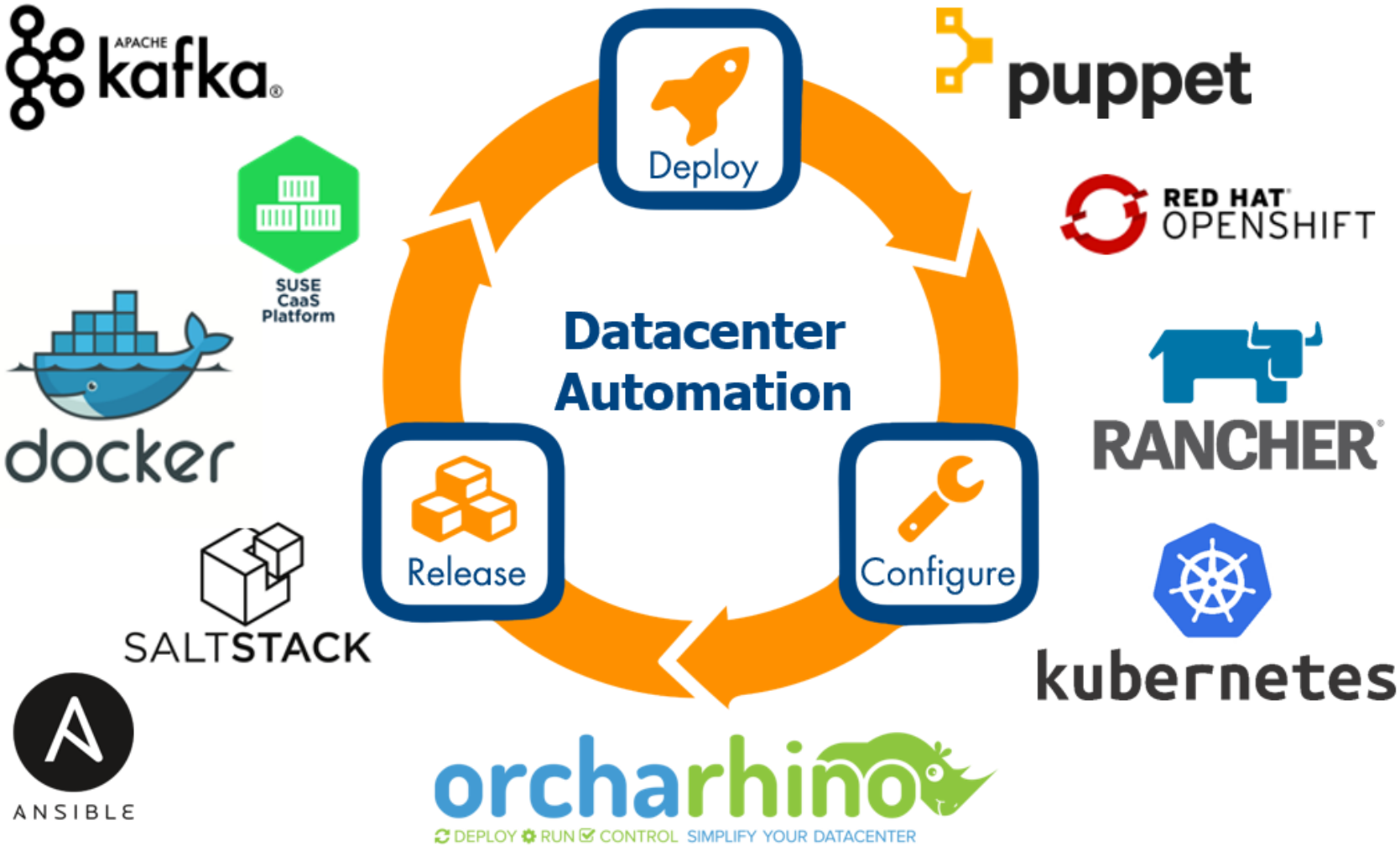
```
hetzner_default
├── create.yml
├── destroy.yml
├── molecule.yml
├── playbook.yml
├── prepare.yml
└── tests
    └── test_default.py
```

Use different Driver

```
# molecule.yml
---
[...]
driver:
  name: hetznercloud
.platform_base: &platform_base
  server_type: cx11
platforms:
  - <<: *platform_base
    name: centos7
    image: centos-7
  - <<: *platform_base
    name: debian10
    image: debian-10
```

There is more …

- even more driver: DigitalOcean, Podman, Vagrant, LXC, EC2,…
- Test-Driven-Development → make red green
- systemd in docker? → RTFM: examples
- use CI/CD → RTFM: examples


APACHE kafka
puppet
SUSE CaaS Platform
RED HAT OPENSHIFT
docker
Deploy
Datacenter Automation
RANCHER
Release
Configure
SALTSTACK
kubernetes
ANSIBLE
orcharhino
DEPLOY RUN CONTROL SIMPLIFY YOUR DATACENTER

Questions?

- Matthias Dellweg
- Physicist
- Senior Software Engineer
- Foreman, Pulp, Ansible (FAM), …
- Github: mdellweg, dellweg@atix.de, IRC: x9c4

- Bernhard Hopfenmüller
- Physicist
- Senior IT Consultant
- Ansible (FAM), Kafka, "DevOps", …
- Github/Twitter/IRC: Fobhep, hopfenmueller@atix.de

https://github.com/ATIX-AG/ansible_nginx_sysinfo