SUSECON digital 21

19 MAY 2021

Writing Salt Formulas for SUSE Manager in a test-driven nutshell

DEV-1025

whoami

Christian
Stankowic

Senior System Engineer

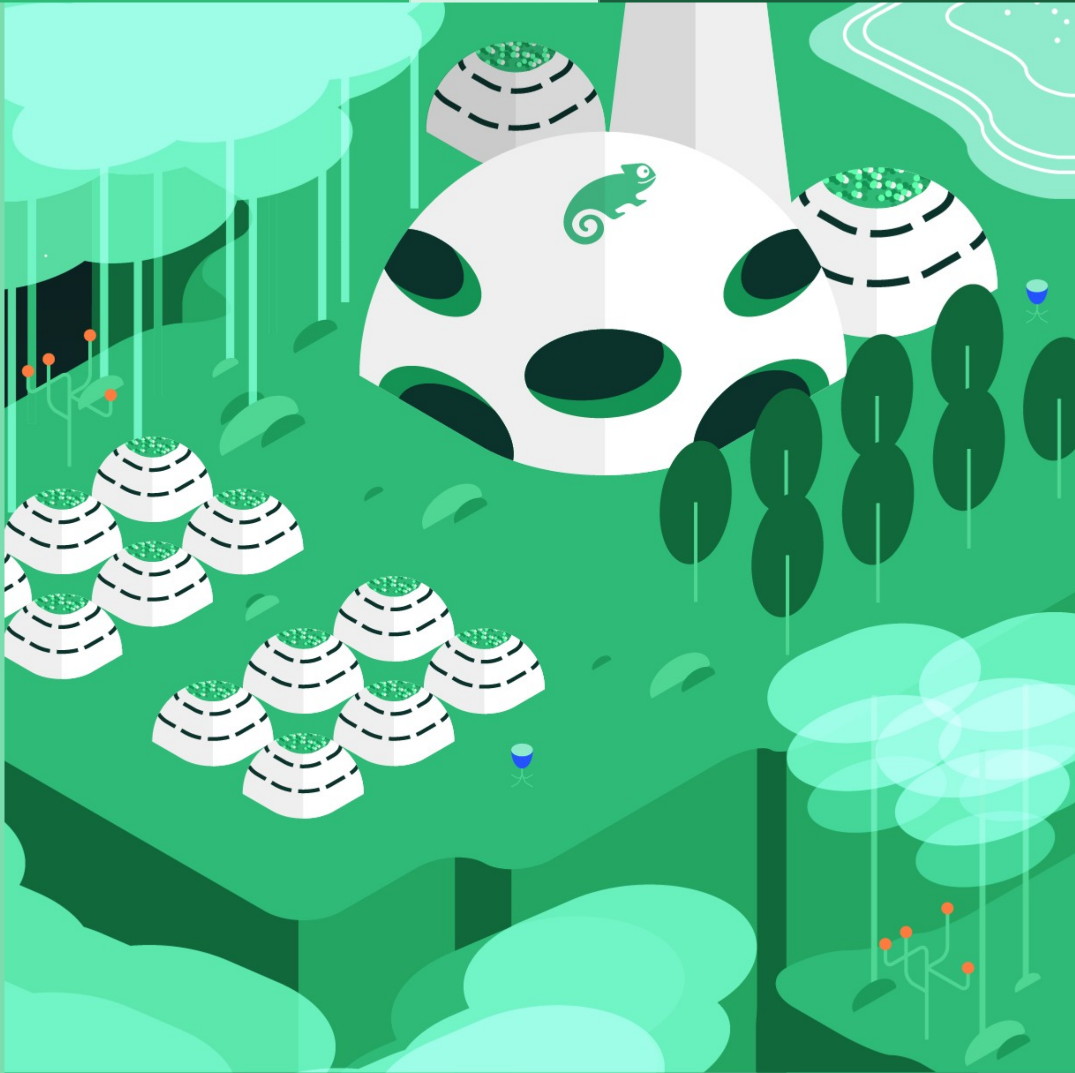
christian.stankowic@sva.de

SUSECON digital 21

Agenda

Recap: SUSE Manager and Salt

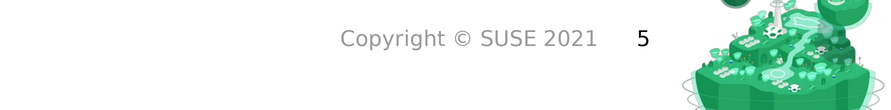
Usability meets automation



SUSECON digital 21

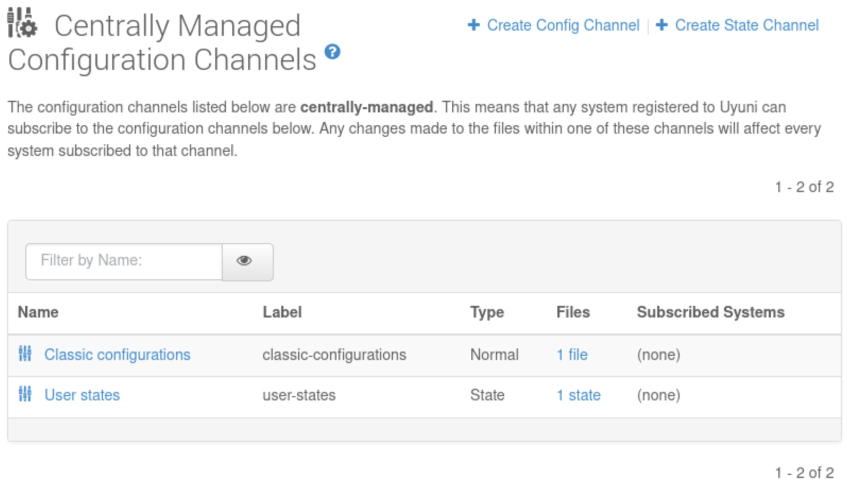
SUSE Manager + Salt =

- Back in 2016, SUMA 3.0 introduced Salt for the very first time
- Modern **declarative** configuration management
- Faster communication than `osad`
- Infrastructure deployment using `salt-cloud`



SUSE Manager + Salt =

- SUMA integrates a full Salt Master
- Configuration using WebUI
- Same workflow users know from legacy configuration management
- Salt States can be configured on many levels
  - System, group, organization



Centrally Managed Configuration Channels
+ Create Config Channel | + Create State Channel
Filter by Name:

The configuration channels listed below are **centrally-managed**. This means that any system registered to Uyuni can subscribe to the configuration channels below. Any changes made to the files within one of these channels will affect every system subscribed to that channel.

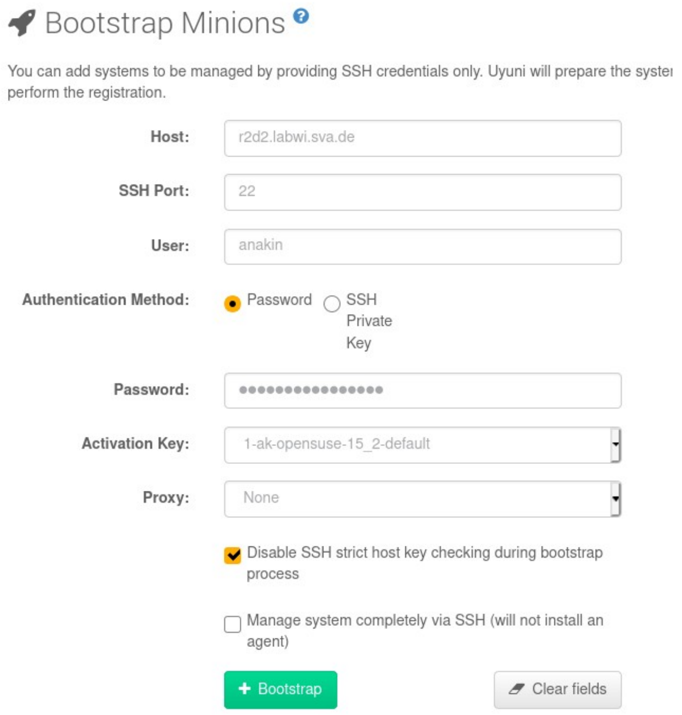
1 - 2 of 2

Name	Label	Type	Files	Subscribed Systems
Classic configurations	classic-configurations	Normal	1 file	(none)
User states	user-states	State	1 state	(none)

1 - 2 of 2

SUSE Manager + Salt =

- New systems can easily be bootstrapped
  - Enter hostname, password
  - Select Activation Key
- High State can be checked and set from the WebUI
  - Recurring States in 4.1



Bootstrap Minions

You can add systems to be managed by providing SSH credentials only. Uyuni will prepare the syste
perform the registration.

Host:	r2d2.labwi.sva.de
SSH Port:	22
User:	anakin
Authentication Method:	[✓ Password] SSH Private Key
Password:	●●●●●●●●●●●●●●●●
Activation Key:	1-ak-opensuse-15_2-default
Proxy:	None

[✓] Disable SSH strict host key checking during bootstrap process

[ ] Manage system completely via SSH (will not install an agent)

Salt Formula 101

SUSECON digital 21

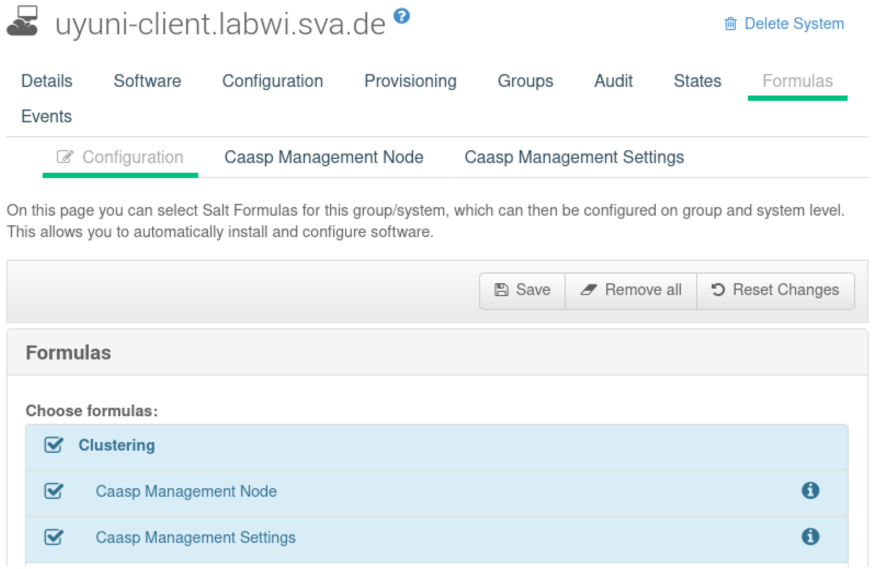
Salt Formula 101

- Reusable Salt code
- Easy to adjust via **Pillar**
- e.g. ntp-formula
  - Set servers (`ntp.servers`)
  - Set service (`ntp.service`)
- Apply state to configure

- Available from the Salt community
  - ~350 formulas
- SUSE also offers some formulas
  - `dhcpd`, `tftpd`, PXE, `bind`
  - Prometheus, Grafana
  - SAP HANA / NetWeaver

Salt Formula and SUSE Manager

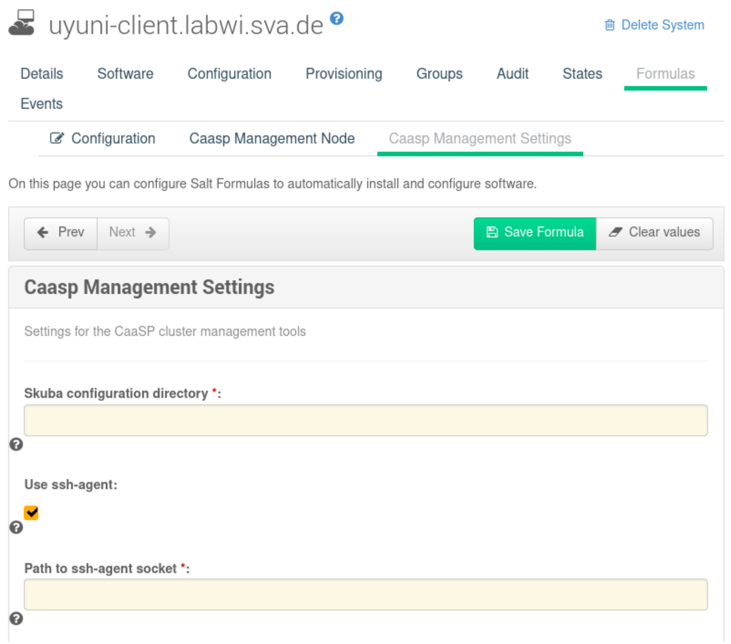
- Salt Formulas can be controlled using the SUMA WebUI
  - Stored in `/usr/share/susemanager`
- Formula Pillar is mapped to WebUI elements
- Simplifies service deployment drastically



On this page you can select Salt Formulas for this group/system, which can then be configured on group and system level. This allows you to automatically install and configure software.

SUSECON digital 21

Setting Pillar and applying High State


uyuni-client.labwi.sva.de

Delete System

Details | Software | Configuration | Provisioning | Groups | Audit | States | Formulas | Events

Configuration | Caasp Management Node | Caasp Management Settings

On this page you can configure Salt Formulas to automatically install and configure software.

Prev | Next | Save Formula | Clear values

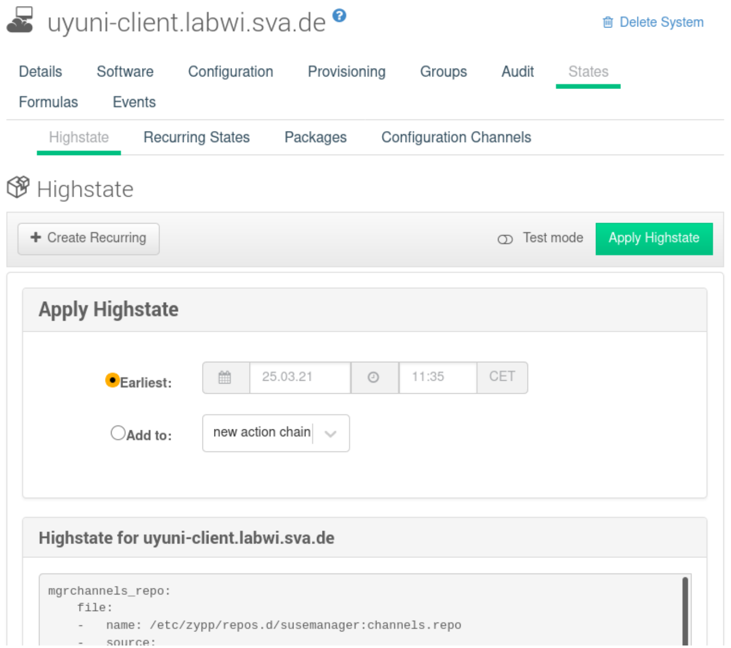
Caasp Management Settings

Settings for the CaaSP cluster management tools

Skuba configuration directory *:

Use ssh-agent: ✓

Path to ssh-agent socket *:


uyuni-client.labwi.sva.de

Delete System

Details | Software | Configuration | Provisioning | Groups | Audit | States | Formulas | Events

Highstate | Recurring States | Packages | Configuration Channels

Highstate

Create Recurring | Test mode | Apply Highstate

Apply Highstate

[✓ Earliest:] 25.03.21 11:35 CET

Add to: new action chain

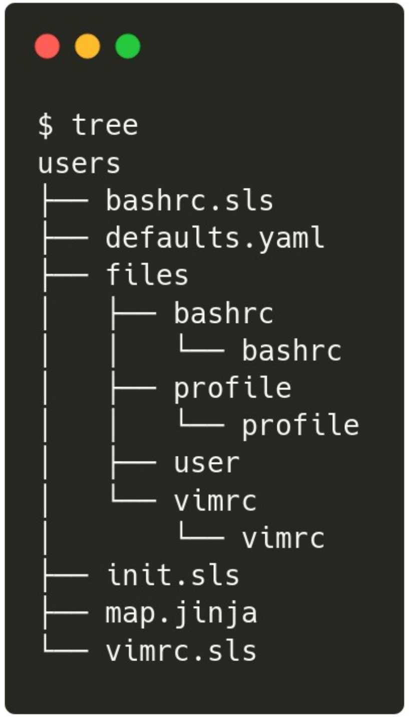
Highstate for uyuni-client.labwi.sva.de

```
mgrchannels_repo:
    file:
    -   name: /etc/zypp/repos.d/susemanager:channels.repo
    -   source:
```

Folder hierarchy

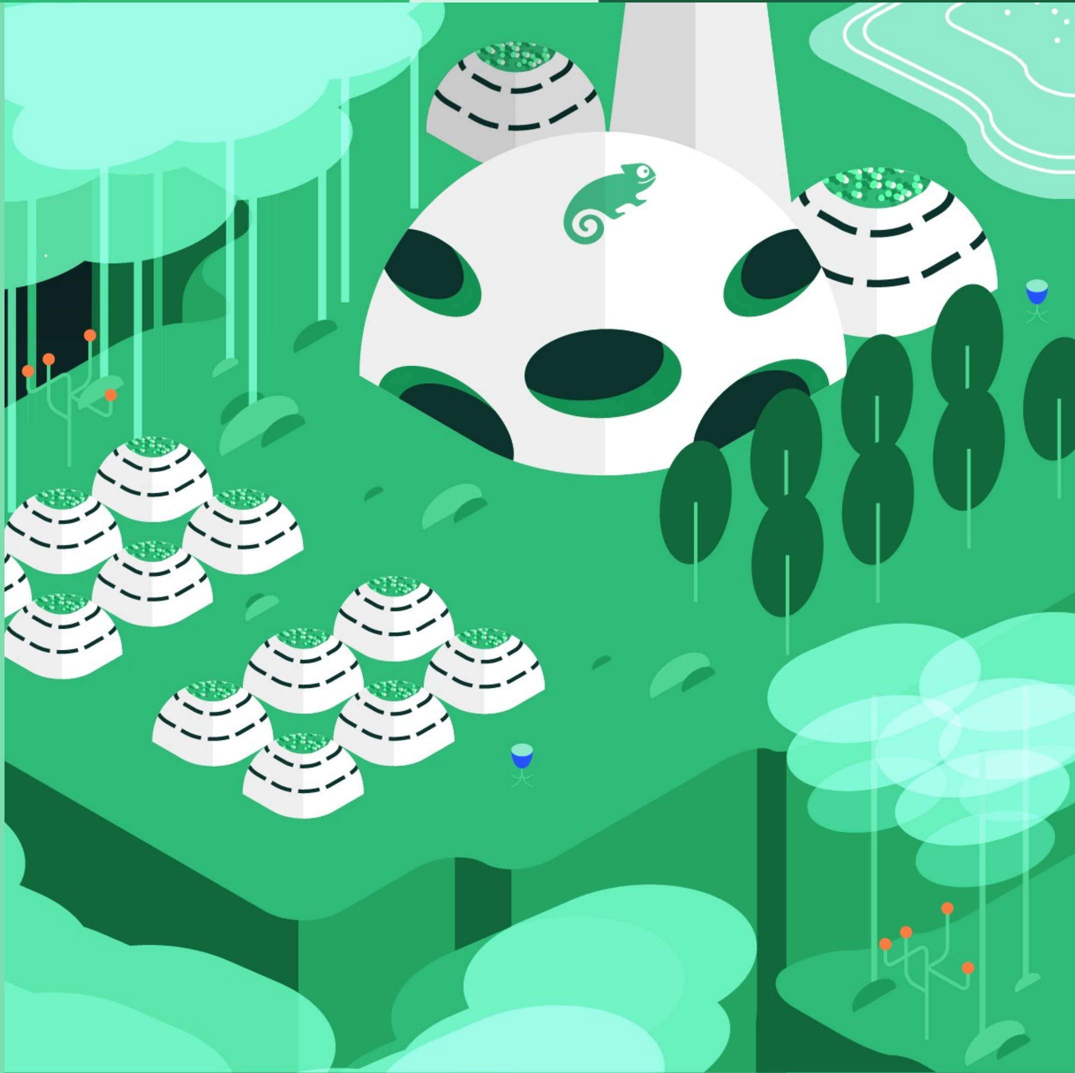
Formula folder can contain:

- `init.sls` – Salt state entry
- `map.jinja` – set of dictionaries per distribution, architecture, etc.
- `files` – files to copy to target system
- `templates` – Jinja2 template files



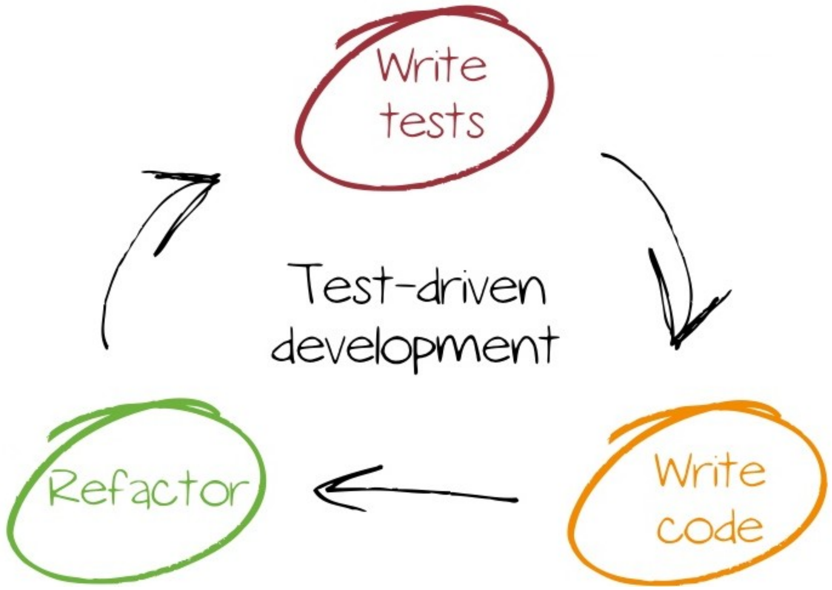
```
$ tree
users
├── bashrc.sls
├── defaults.yaml
├── files
│   ├── bashrc
│   │   └── bashrc
│   ├── profile
│   │   └── profile
│   ├── user
│   └── vimrc
│       └── vimrc
├── init.sls
├── map.jinja
└── vimrc.sls
```

Test-driven development with test-kitchen



Recap: Test-driven development

- Approach to write acceptance criteria **before** actual code
- Can increase code quality as planning is intensified
- Also slightly increases effort
- Assists **iterative** development



test-kitchen

- Integration tool for developing and testing infrastructure code
- Numerous drivers for virtualization and container technology
- Supports multiple testing frameworks
- Written in Ruby
- Part of Chef suite

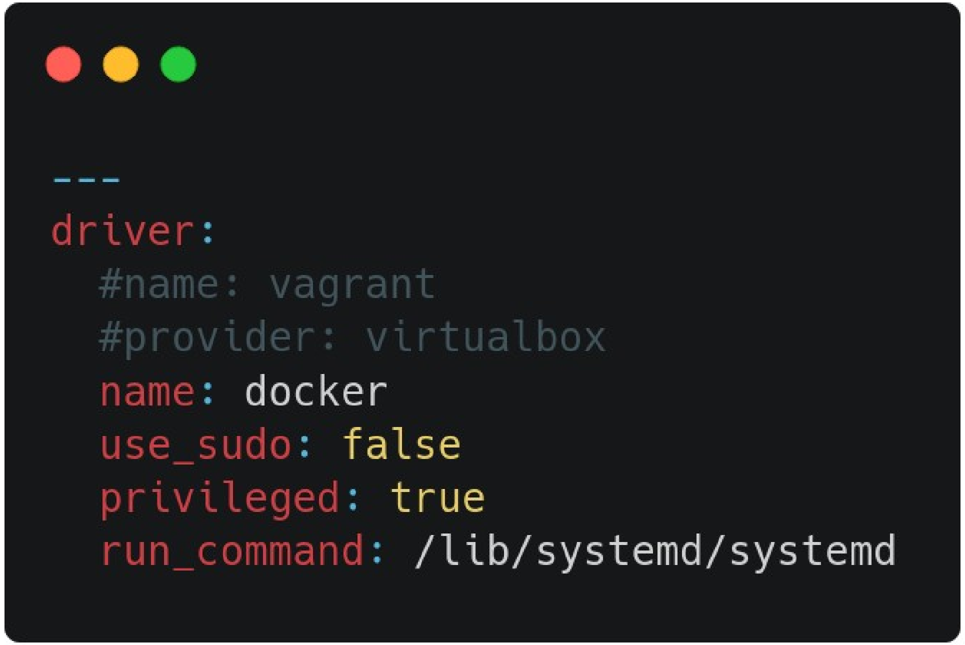
- Supports
  - Hashicorp Vagrant
  - Microsoft Azure
  - Docker
  - Ansible, Salt, Puppet, Chef
  - InSpec, ServerSpec

test-kitchen commands

`kitchen init`	Create empty configuration
`kitchen list`	List specified instances
`kitchen create`	Create instances
`kitchen converge`	Apply Salt states
`kitchen verify`	Run unit tests
`kitchen destroy`	Remove instances
`kitchen test`	Create, converge, verify and destroy

Example – driver definition

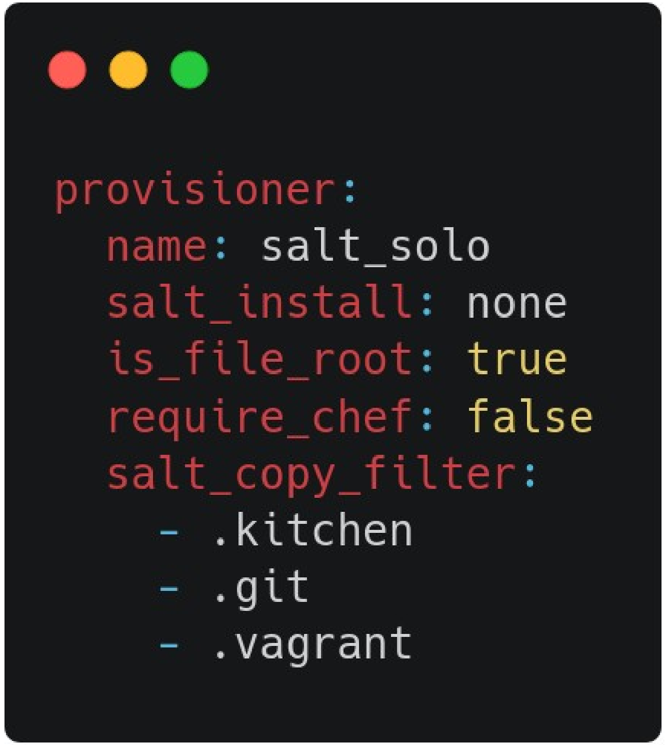
kitchen.yml



```
---
driver:
  #name: vagrant
  #provider: virtualbox
  name: docker
  use_sudo: false
  privileged: true
  run_command: /lib/systemd/systemd
```

Example – provisioner definition

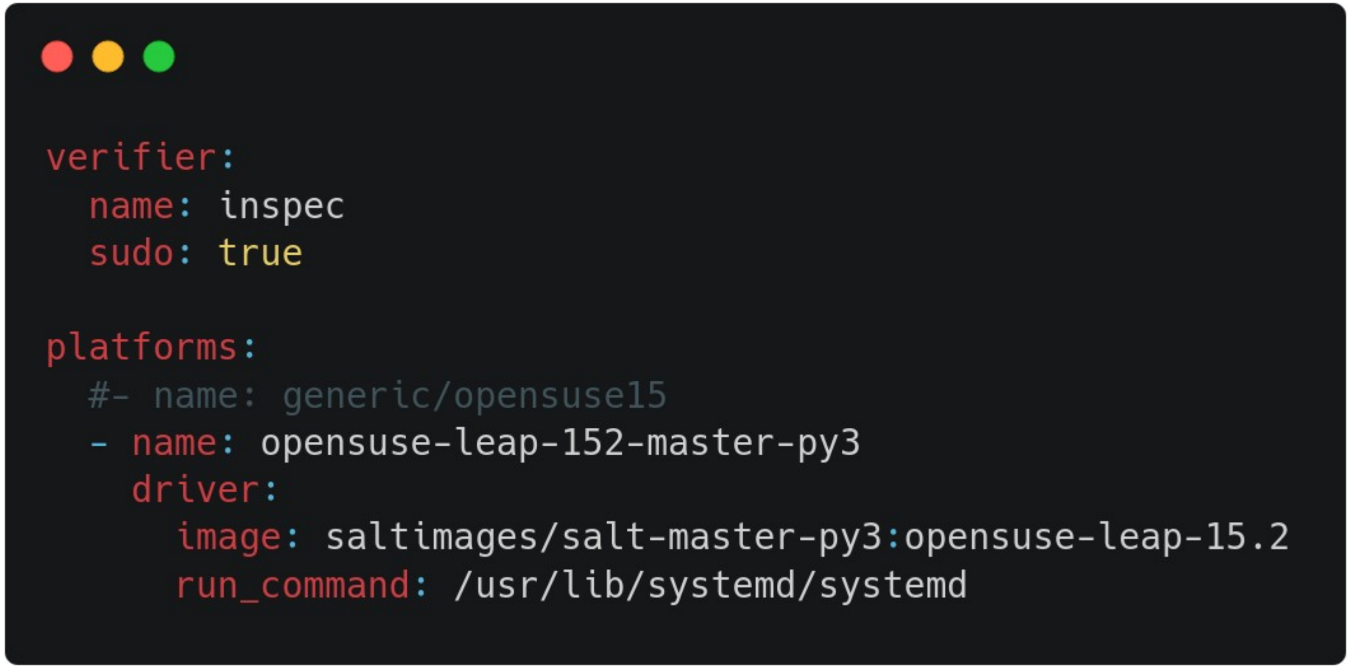
kitchen.yml



```
provisioner:
  name: salt_solo
  salt_install: none
  is_file_root: true
  require_chef: false
  salt_copy_filter:
    - .kitchen
    - .git
    - .vagrant
```

Example – verifier and platform definitions

kitchen.yml

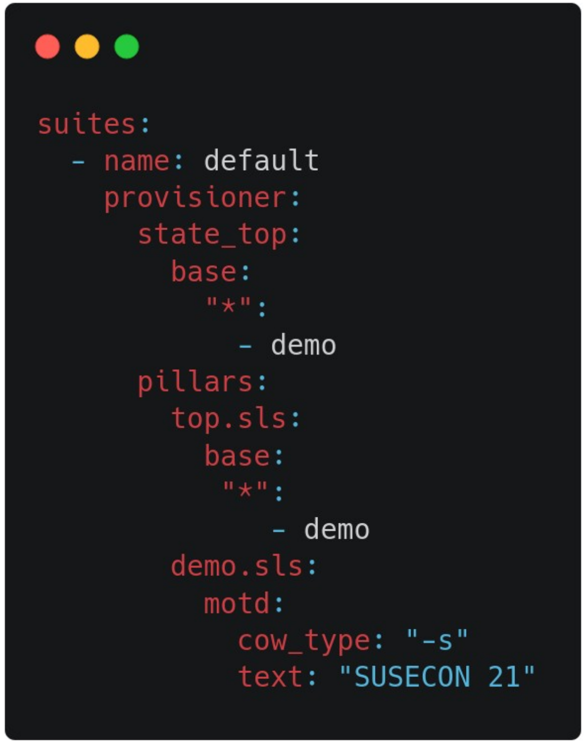


```
verifier:
  name: inspec
  sudo: true

platforms:
  #- name: generic/opensuse15
  - name: opensuse-leap-152-master-py3
    driver:
      image: saltimages/salt-master-py3:opensuse-leap-15.2
      run_command: /usr/lib/systemd/systemd
```

Example – suite definitions (1/2)

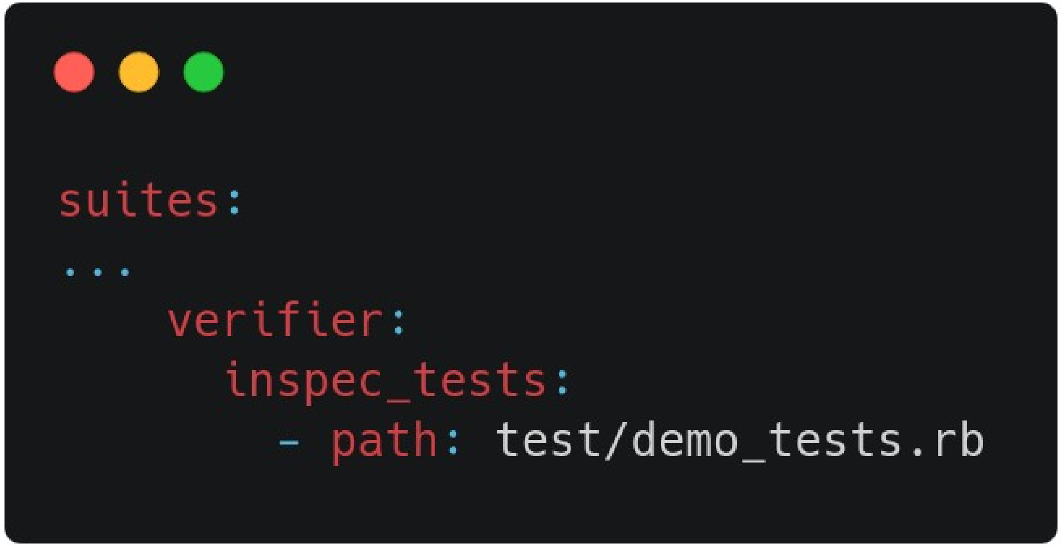
`kitchen.yml`



```
suites:
  - name: default
    provisioner:
      state_top:
        base:
          "*":
            - demo
      pillars:
        top.sls:
          base:
            "*":
              - demo
        demo.sls:
          motd:
            cow_type: "-s"
            text: "SUSECON 21"
```

Example – suite definitions (2/2)

`kitchen.yml`



```
suites:
...
    verifier:
      inspec_tests:
        - path: test/demo_tests.rb
```

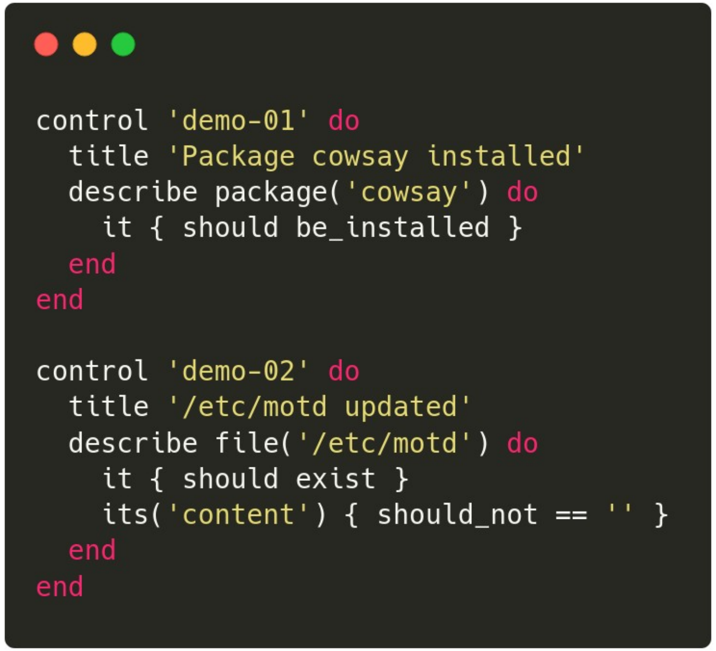
InSpec

- Auditing and testing framework
- Human- and machine-readable
- Written in Ruby
- Part of Chef suite

- Supports
  - Hashicorp Vagrant
  - Microsoft Azure
  - Docker
  - Ansible, Salt, Puppet, Chef
  - InSpec, ServerSpec

Example

demo_tests.rb

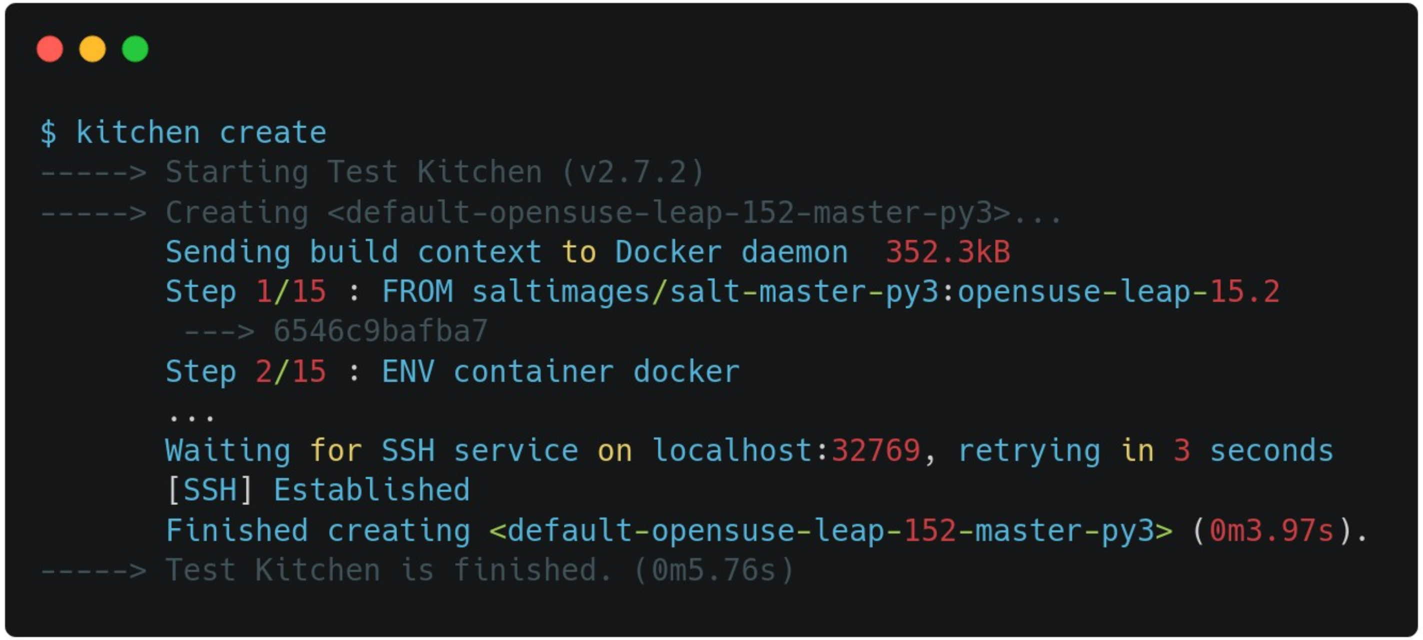


```ruby
control 'demo-01' do
  title 'Package cowsay installed'
  describe package('cowsay') do
    it { should be_installed }
  end
end

control 'demo-02' do
  title '/etc/motd updated'
  describe file('/etc/motd') do
    it { should exist }
    its('content') { should_not == '' }
  end
end
```

Putting it all together...

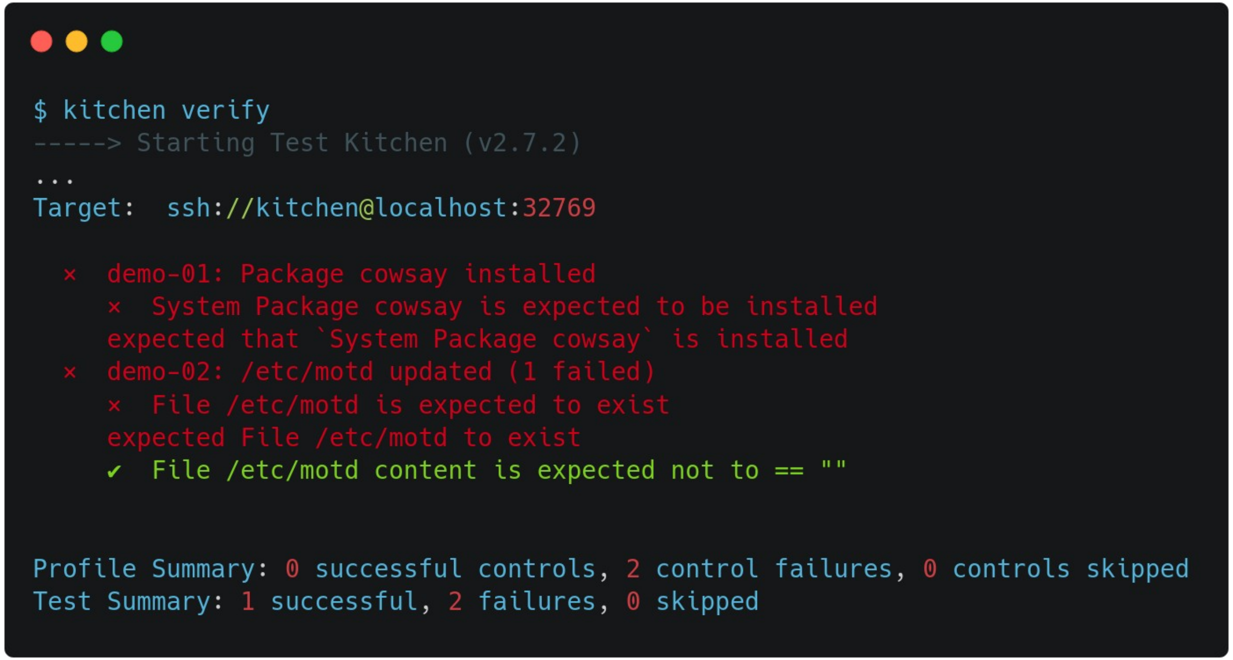
Create instances



```
$ kitchen create
-----> Starting Test Kitchen (v2.7.2)
-----> Creating <default-opensuse-leap-152-master-py3>...
       Sending build context to Docker daemon  352.3kB
       Step 1/15 : FROM saltimages/salt-master-py3:opensuse-leap-15.2
        ---> 6546c9bafba7
       Step 2/15 : ENV container docker
       ...
       Waiting for SSH service on localhost:32769, retrying in 3 seconds
       [SSH] Established
       Finished creating <default-opensuse-leap-152-master-py3> (0m3.97s).
-----> Test Kitchen is finished. (0m5.76s)
```

Putting it all together...

Unit tests will fail



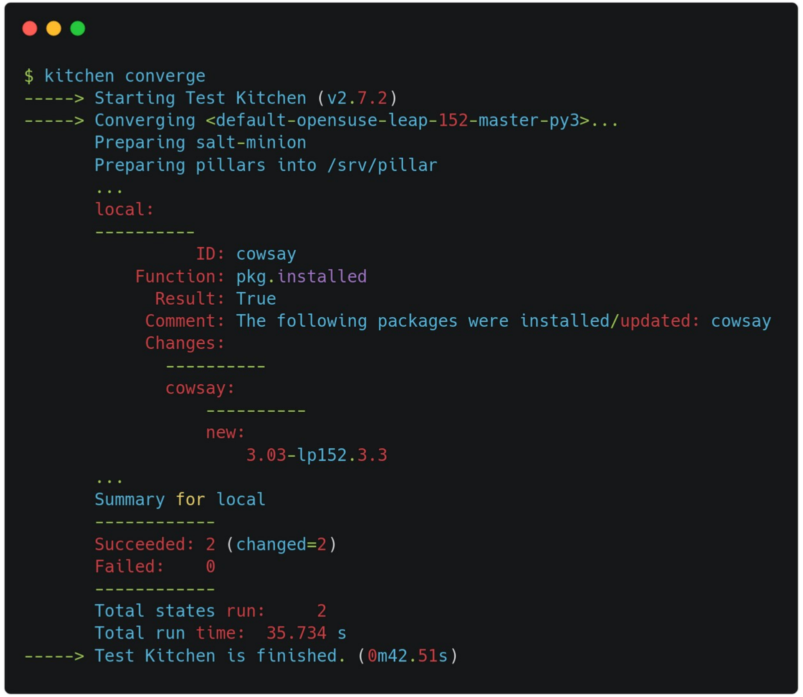
```
$ kitchen verify
-----> Starting Test Kitchen (v2.7.2)
...
Target:  ssh://kitchen@localhost:32769

  ×  demo-01: Package cowsay installed
     ×  System Package cowsay is expected to be installed
     expected that `System Package cowsay` is installed
  ×  demo-02: /etc/motd updated (1 failed)
     ×  File /etc/motd is expected to exist
     expected File /etc/motd to exist
     ✔  File /etc/motd content is expected not to == ""


Profile Summary: 0 successful controls, 2 control failures, 0 controls skipped
Test Summary: 1 successful, 2 failures, 0 skipped
```

Putting it all together...

Running Salt states

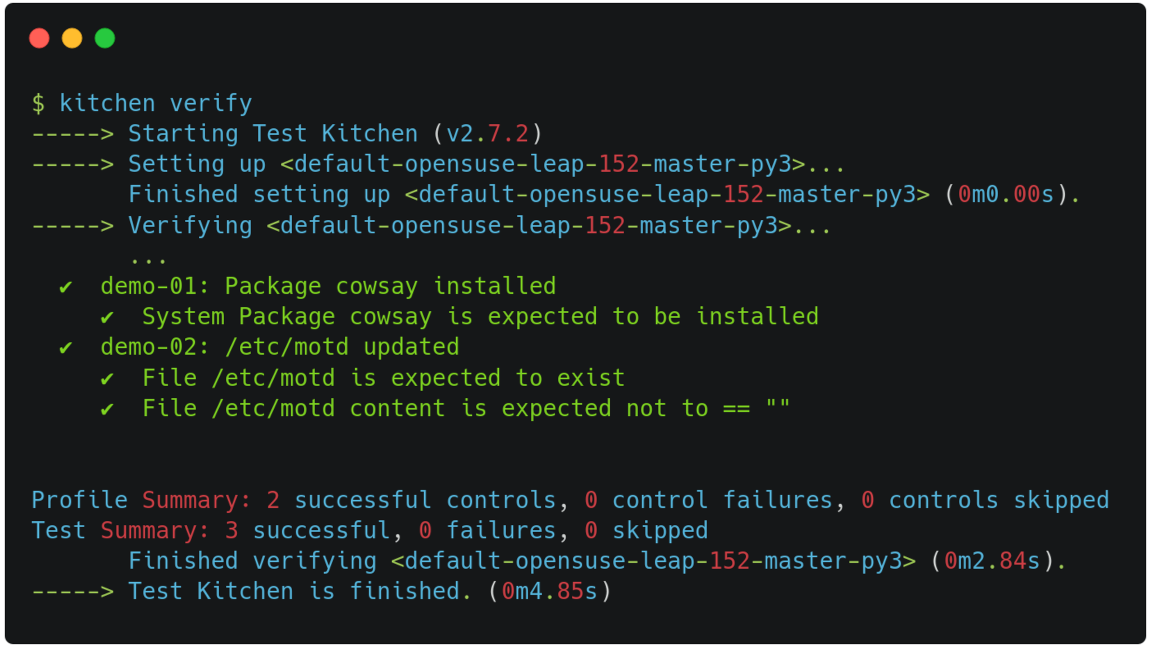


```
$ kitchen converge
-----> Starting Test Kitchen (v2.7.2)
-----> Converging <default-opensuse-leap-152-master-py3>...
       Preparing salt-minion
       Preparing pillars into /srv/pillar
       ...
       local:
       ----------
                 ID: cowsay
           Function: pkg.installed
             Result: True
            Comment: The following packages were installed/updated: cowsay
            Changes:
              ----------
              cowsay:
                  ----------
                  new:
                      3.03-lp152.3.3
       ...
       Summary for local
       ------------
       Succeeded: 2 (changed=2)
       Failed:    0
       ------------
       Total states run:     2
       Total run time:  35.734 s
-----> Test Kitchen is finished. (0m42.51s)
```

SUSECON digital 21

Putting it all together...

Unit tests will now pass

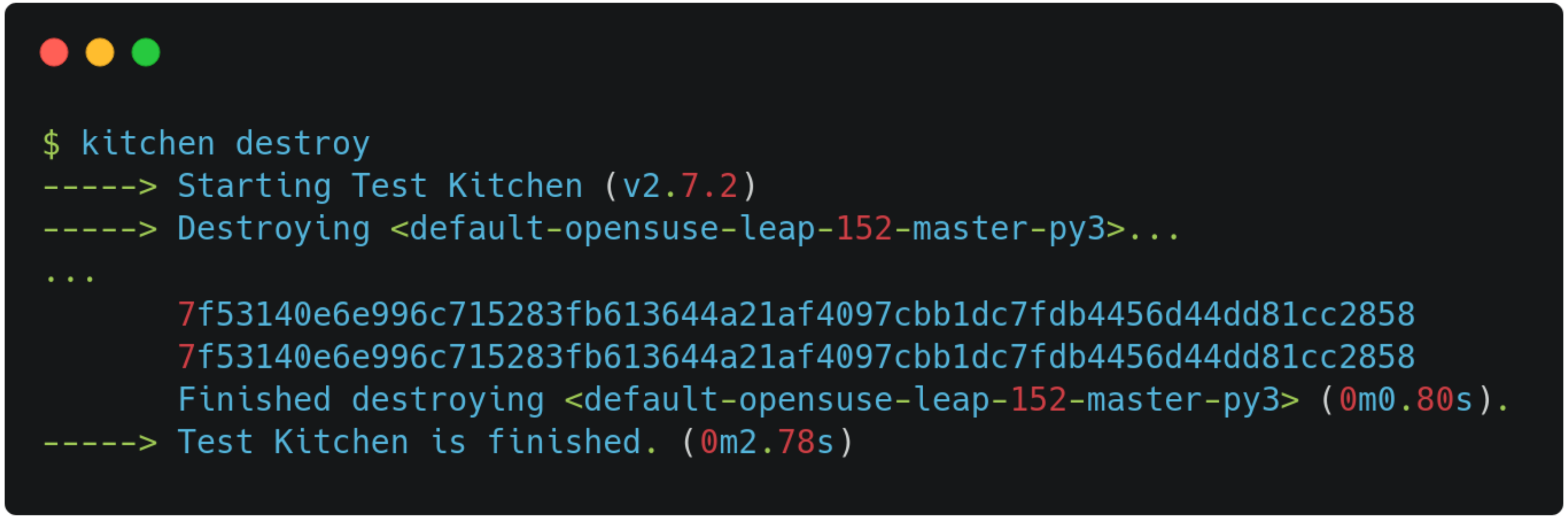


```
$ kitchen verify
-----> Starting Test Kitchen (v2.7.2)
-----> Setting up <default-opensuse-leap-152-master-py3>...
       Finished setting up <default-opensuse-leap-152-master-py3> (0m0.00s).
-----> Verifying <default-opensuse-leap-152-master-py3>...
       ...
  ✔  demo-01: Package cowsay installed
     ✔  System Package cowsay is expected to be installed
  ✔  demo-02: /etc/motd updated
     ✔  File /etc/motd is expected to exist
     ✔  File /etc/motd content is expected not to == ""


Profile Summary: 2 successful controls, 0 control failures, 0 controls skipped
Test Summary: 3 successful, 0 failures, 0 skipped
       Finished verifying <default-opensuse-leap-152-master-py3> (0m2.84s).
-----> Test Kitchen is finished. (0m4.85s)
```

Putting it all together...

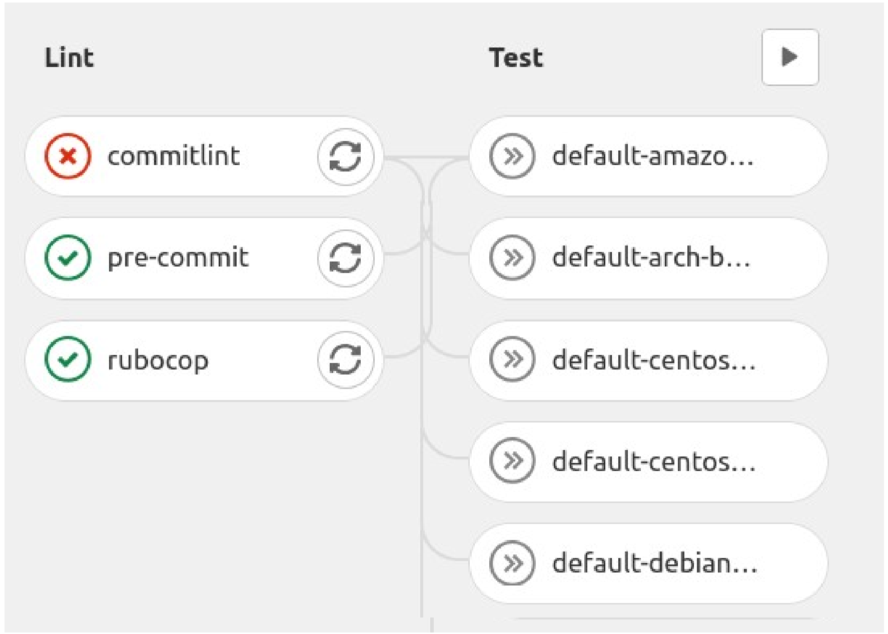
Cleaning up



```
$ kitchen destroy
-----> Starting Test Kitchen (v2.7.2)
-----> Destroying <default-opensuse-leap-152-master-py3>...
...
       7f53140e6e996c715283fb613644a21af4097cbb1dc7fdb4456d44dd81cc2858
       7f53140e6e996c715283fb613644a21af4097cbb1dc7fdb4456d44dd81cc2858
       Finished destroying <default-opensuse-leap-152-master-py3> (0m0.80s).
-----> Test Kitchen is finished. (0m2.78s)
```

Salt Formula Template

- The Salt community offers a pre-built template
- Includes a GitLab pipeline
  - Linting
  - test-kitchen for major Linux distributions
  - dind (Docker in Docker)



SUSECON digital 21

Porting existing Salt Formulas to SUSE Manager

SUSECON digital 21

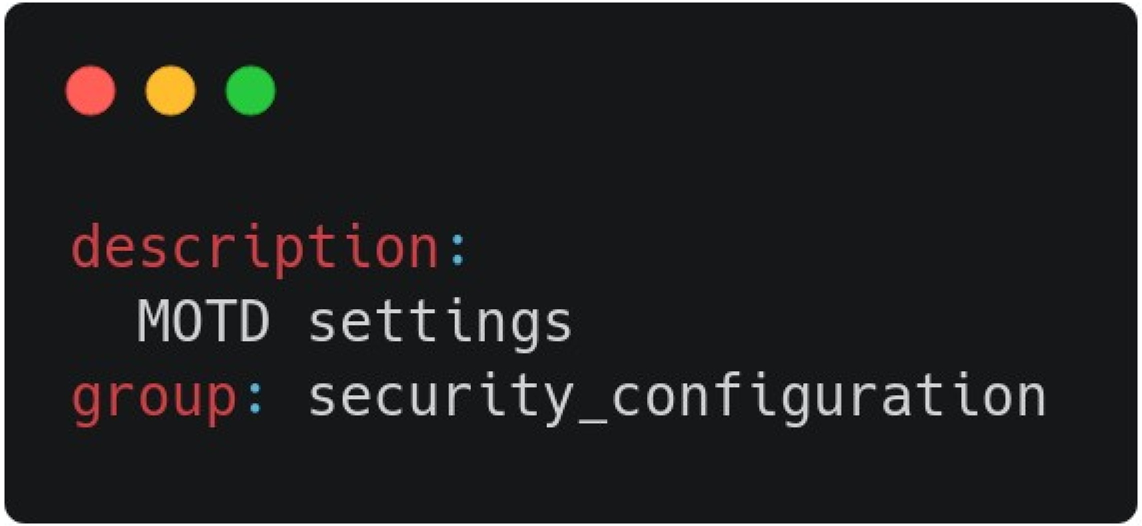
SUSE Manager Formula directories

- Folders per Formula:
  - `/usr/share/susemanager/formulas/metadata/<name>`
  - `/usr/share/susemanager/formulas/states/<name>`
- `states` contains Formula `.sls` files

- `metadata/form.yml` maps Pillar to UI elements
  - Input fields, Groups, Boxes,...
  - Pre-defined values
- `metadata/metadata.yml` defines
  - Formula description
  - Placement in SUMA WebUI

Example

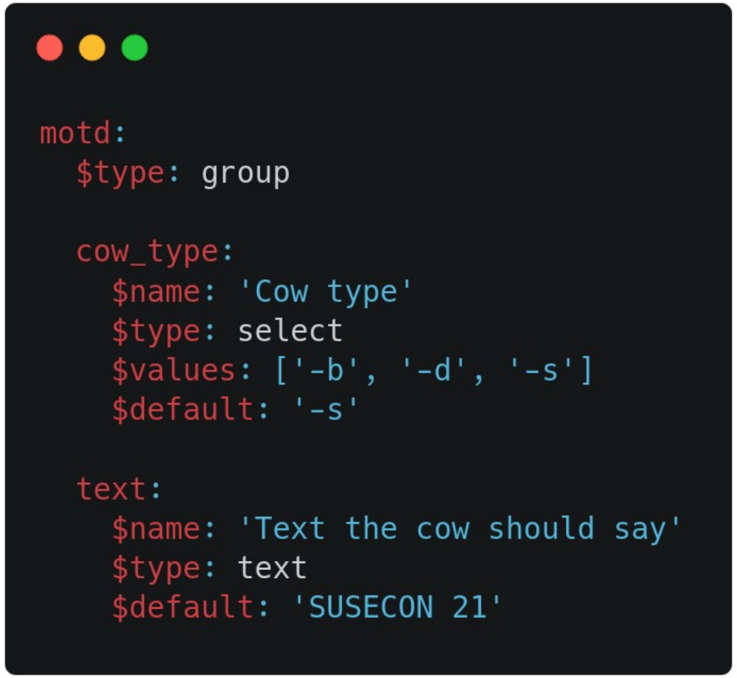
/usr/share/susemanager/formulas/metadata/demo/metadata.yml



```yaml
description:
  MOTD settings
group: security_configuration
```

Example

/usr/share/susemanager/formulas/metadata/demo/form.yml



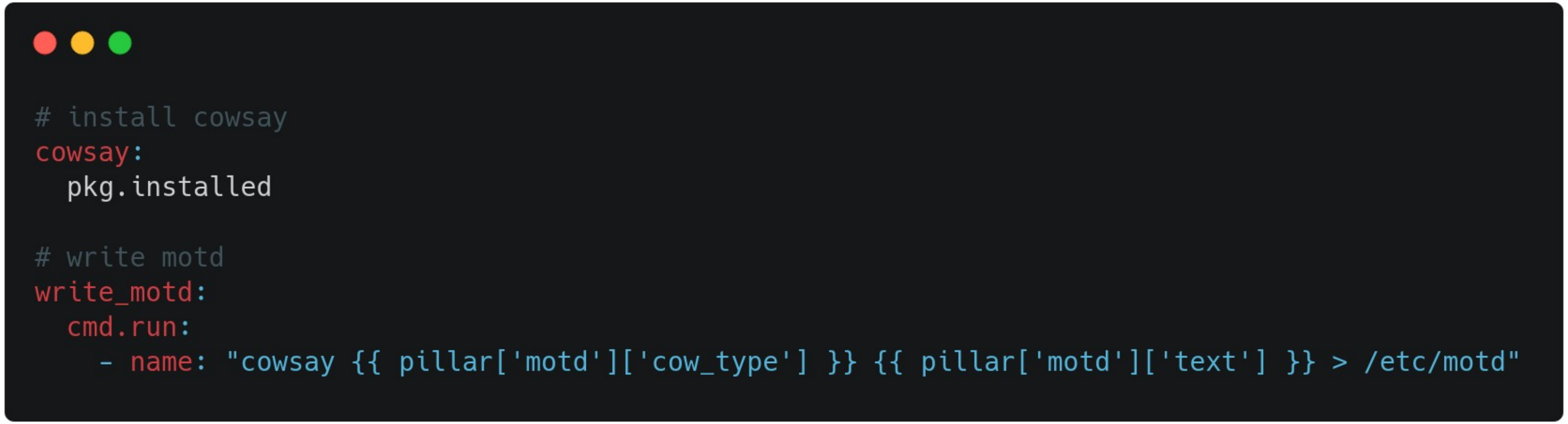
```yaml
motd:
  $type: group

  cow_type:
    $name: 'Cow type'
    $type: select
    $values: ['-b', '-d', '-s']
    $default: '-s'

  text:
    $name: 'Text the cow should say'
    $type: text
    $default: 'SUSECON 21'
```

Example

/usr/share/susemanager/formulas/states/demo/init.sls

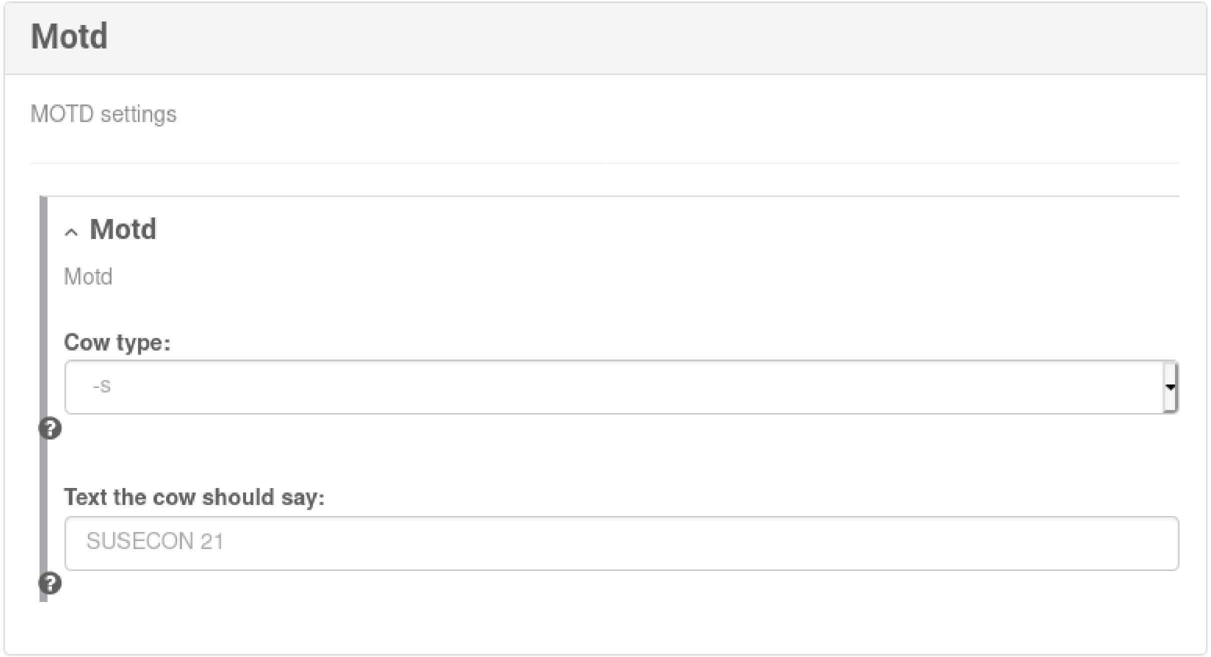


```
# install cowsay
cowsay:
  pkg.installed

# write motd
write_motd:
  cmd.run:
    - name: "cowsay {{ pillar['motd']['cow_type'] }} {{ pillar['motd']['text'] }} > /etc/motd"
```

Example

What it looks like


Motd

MOTD settings

Motd

Motd

Cow type:

-s

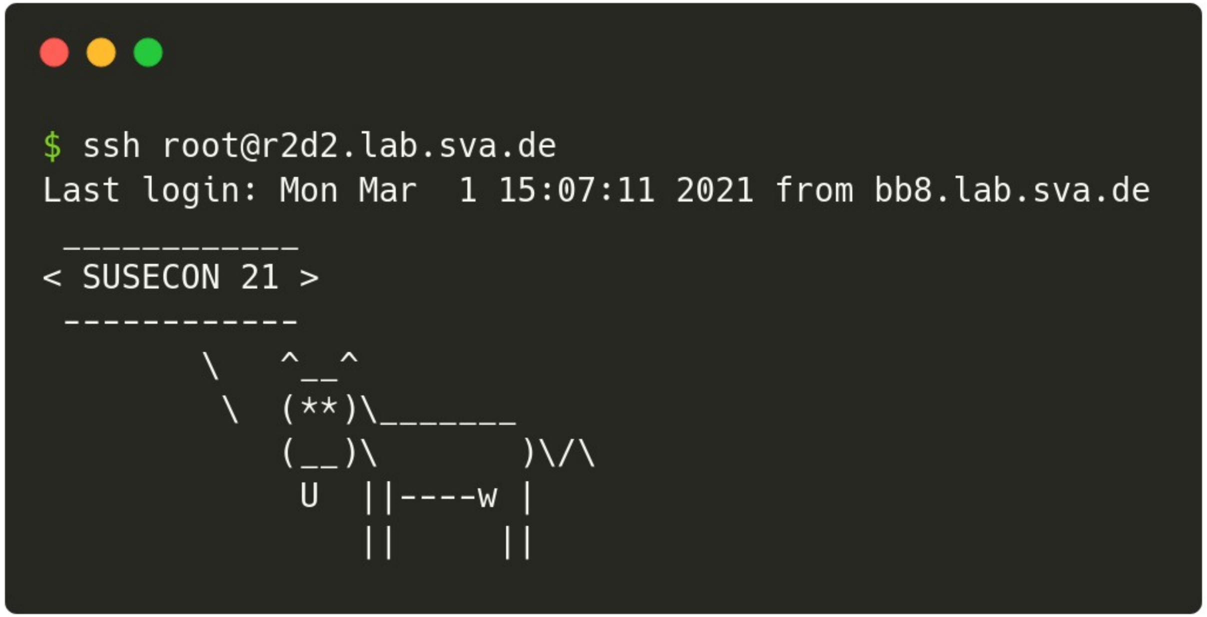
Text the cow should say:

SUSECON 21

SUSECON digital 21

Example

What it looks like



```
$ ssh root@r2d2.lab.sva.de
Last login: Mon Mar  1 15:07:11 2021 from bb8.lab.sva.de
 ____________
< SUSECON 21 >
 ------------
        \   ^__^
         \  (**)\_______
            (__)\       )\/\
             U  ||----w |
                ||     ||
```

Links

- Official SaltStack Formula community (~350 Formulas): https://github.com/saltstack-formulas/
- Inofficial SaltStack Formula community (~180 Formulas): https://github.com/salt-formulas
- Formula Template: https://github.com/saltstack-formulas/template-formula
- Example from this presentation: https://github.com/stdevel/demo-formula

SUSECON digital 21

Thank you

For more information, contact SUSE at:

+1 800 796 3700 (U.S./Canada)

+49 (0)911-740 53-0 (Worldwide)

Maxfeldstrasse 5

90409 Nuremberg

www.suse.com