ansible-role-config-light Documentation

Release 1.4.0

Vladimir Botka

Apr 12, 2020

TABLE OF CONTENTS:

CHAPTER

ONE

QUICK START GUIDE

For those users who want to quickly try the role this guide provides an example of how to install and configure Lighttpd on single FreeBSD host. The procedure is generic and can be easily modified to install and configure other applications on other systems. See examples in the directory *contrib*. The control node of this example is Linux and the user is member of the group *adm*.

- Install the role `vbotka.config_light`

```
shell> ansible-galaxy install vbotka.config_light
```

- Create the playbook `config-light.yml` for single host srv.example.com (2)

```
shell> cat config-light.yml
- hosts: srv.example.com
  gather_facts: true
  connection: ssh
  remote_user: admin
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - vbotka.config_light
```

- Create `host_vars` with customized variables of the role and with the variables of the application.

```
shell> ls -1 host_vars/srv.example.com/config-light-*
host_vars/srv.example.com/config-light-common.yml
host_vars/srv.example.com/config-light-lighttpd.yml
```

- Create *host_vars* with customized variables of the role. To speedup the execution let's set the control-flow variables (2-5) to *false* and disable some steps. Enable the steps selectively when needed. The configuration files of the role will be stored in the directory *conf-light* in the current directory of the playbook (10). Set the ownership and permissions of the directories on the cont-roll node so that the user who is running the playbook will be able both read and write the files, and create the directories (7-9) (11-14).

```
shell> cat host_vars/srv.example.com/config-light-common.yml
cl_sanity: false
cl_setup: false
cl_install: false
cl_debug: false
cl_backup: true
cl_dird_owner: "root"
cl_dird_group: "adm"
cl_dird_dmode: "0770"
```

(continues on next page)

(continued from previous page)

```
cl_dird: "{{ playbook_dir }}/conf-light"
cl_dira_owner: "root"
cl_dira_group: "adm"
cl_dira_dmode: "0770"
cl_dira_fmode: "0660"
```

- Create *host_vars* with the variables of the application. Start the server (2), run the server at boot (3), and configure two files.

```
shell> cat host_vars/srv.example.com/config-light-lighttpd.yml
cl_service_lighttpd_enable: true
cl_service_lighttpd_state: 'started'

# /usr/local/etc/lighttpd/lighttpd.conf
cl_lighttpd_server_port: '80'
cl_lighttpd_server_useipv6: 'disable'
cl_lighttpd_server_username: 'www'
cl_lighttpd_server_groupname: 'www'
cl_lighttpd_server_document_root: "/usr/local/www/lighttpd"
cl_lighttpd_lighttpdconf_lines:
  - regexp: '^\s*server.port\s*=\s*(.*)$'
    line: 'server.port = "{{ cl_lighttpd_server_port }}"'
  - regexp: '^\s*server.use-ipv6\s*=\s*(.*)$'
    line: 'server.use-ipv6 = "{{ cl_lighttpd_server_useipv6 }}"'
  - regexp: '^\s*server.username\s*=\s*(.*)$'
    line: 'server.username = "{{ cl_lighttpd_server_username }}"'
  - regexp: '^\s*server.groupname\s*=\s*(.*)$'
    line: 'server.groupname = "{{ cl_lighttpd_server_groupname }}"'
  - regexp: '^\s*server.document-root\s*=\s*(.*)$'
    line: 'server.document-root = "{{ cl_lighttpd_server_document_root }}"'

# /etc/rc.conf
cl_lighttpd_rcconf_lighttpd_enable: 'YES'
cl_lighttpd_rcconf_lines:
    - regexp: '^lighttpd_enable(.*)$'
      line: 'lighttpd_enable="{{ cl_lighttpd_rcconf_lighttpd_enable }}"'
```

- Create configuration files in the directory `conf-light`.

```
shell> tree conf-light
conf-light/
├── files.d
│   ├── lighttpd-lighttpdconf
│   └── lighttpd-rcconf
├── handlers.d
│   └── lighttpd-freebsd
├── packages.d
│   └── lighttpd
├── services.d
│   └── lighttpd
└── states.d
    └── lighttpd-server-document-root
```

conf-light/files.d

```
shell> cat conf-light/files.d/lighttpd-lighttpdconf
```

(continues on next page)

(continued from previous page)

```
lighttpd-lighttpdconf:
  path: '/usr/local/etc/lighttpd/lighttpd.conf'
  create: true
  owner: 'root'
  group: 'wheel'
  mode: '0644'
  lines: '{{ cl_lighttpd_lighttpdconf_lines }}'
```

```
shell> cat conf-light/files.d/lighttpd-rcconf
lighttpd_rcconf:
  path: '/etc/rc.conf'
  create: true
  owner: 'root'
  group: 'wheel'
  mode: '0644'
  lines: "{{ cl_lighttpd_rcconf_lines }}"
```

conf-light/handlers.d

```
shell> cat conf-light/handlers.d/lighttpd-freebsd
lighttpd_freebsd:
  template: handlers-auto1.yml.j2
  params:

    - handler: 'enable and start lighttpd'
      module: service
      params:
        - 'name: lighttpd'
        - 'state: started'
        - 'enabled: true'

    - handler: 'disable and stop lighttpd'
      module: service
      params:
        - 'name: lighttpd'
        - 'state: stopped'
        - 'enabled: false'

    - handler: 'reload lighttpd'
      module: service
      params:
        - 'name: lighttpd'
        - 'state: reloaded'
      conditions:
        - '- cl_service_lighttpd_enable|bool'

    - handler: 'restart lighttpd'
      module: service
      params:
        - 'name: lighttpd'
        - 'state: restarted'
      conditions:
        - '- cl_service_lighttpd_enable|bool'

    - handler: 'lighttpd check'
      module: command
```

(continues on next page)

(continued from previous page)

```
      params:
    - 'cmd: /usr/local/sbin/lighttpd -t'
```

conf-light/packages.d

```
shell> cat conf-light/packages.d/lighttpd
lighttpd:
  name: 'www/lighttpd'
```

conf-light/services.d

```
shell> cat conf-light/services.d/lighttpd
lighttpd:
  name: 'lighttpd'
  state: '{{ cl_service_lighttpd_state }}'
  enabled: '{{ cl_service_lighttpd_enable }}'
```

conf-light/states.d

```
shell> cat conf-light/states.d/lighttpd-server-document-root
lighttpd_server_document_root:
  state: directory
  path: '{{ cl_lighttpd_server_document_root }}'
  owner: '{{ cl_lighttpd_server_username }}'
  group: '{{ cl_lighttpd_server_groupname }}'
  mode: '0750'
```

- Enable setup and create variables

```
shell> ansible-playbook config-light.yml -t cl_vars -e 'cl_setup=true'
```

Take a look at directory `conf-light/assemble/` what files were created. Also take a look at the directory `roles/vbotka.config_light/handlers` what handlers were created.

- Enable and test sanity

```
shell> ansible-playbook config-light.yml -t cl_sanity -e 'cl_sanity=true'
```

- Display variables

```
shell> ansible-playbook config-light.yml -t cl_debug -e 'cl_debug=true'
```

- Install packages

```
shell> ansible-playbook config-light.yml -t cl_packages -e 'cl_install=true'
```

- Set files' states

```
shell> ansible-playbook config-light.yml -t cl_states
```

- Create and modify files

```
shell> ansible-playbook config-light.yml -t cl_files
```

- Configure services

```
shell> ansible-playbook config-light.yml -t cl_services
```

- The role and the configuration data in the examples are idempotent. Once the application is installed and configured there should be no changes reported by *ansible-playbook* when running the playbook repeatedly. Disable setup, sanity, debug, and install to speedup the playbook

```
shell> ansible-playbook config-light.yml

[...]

PLAY RECAP ********************************************************************
srv.example.com: ok=21 changed=0 unreachable=0 failed=0 skipped=35 rescued=0 ignored=0
```

- Create file `/usr/local/www/lighttpd/index.html`

```
shell> ll /usr/local/www/lighttpd/index.html
-rw-r--r--  1 www  www  51 Apr 12 18:58 /usr/local/www/lighttpd/index.html
shell> cat /usr/local/www/lighttpd/index.html
<html><body><h1>Lighttpd works!</h1></body></html>
```

- Open the page in a browser `http://srv.example.com/`. The content should be

```
Lighttpd works!
```

CHAPTER

TWO

USER'S GUIDE

Table of Contents

2.1 Introduction

- Ansible role: config_light
- Supported systems: FreeBSD, Ubuntu
- Requirements: None

The role installs packages, creates and configures files, services, and handlers. This provides a simple, but flexible framework to apply basic Ansible modules. Substantial part of the control-flow will be determined by the structure of the data. Some attributes of the dictionaries trigger Ansible modules to modify configuration files, configure services and create handlers.

The role can be used with any supported OS to install and configure arbitrary applications. The role is tested with supported releases of FreeBSD and Ubuntu. It can be expected that other BSD and Linux distributions, that support

the Ansible modules mentioned below, should work with minimal changes. Red Hat and Debian *ansible_os_family* should work out of the box.

Used Ansible modules comprise `package` to install Linux packages, and both `pkgng` and `portinstall` to install FreeBSD packages or ports.

Ansible modules `template`, `lineinfile`, `blockinfile` and `ini_file` are used to configure files. Module `service` is used to manage both Linux and FreeBSD services.

The directory `contrib` comprises examples of how to install and configure various applications, and how to create the handlers and templates.

The user of this role is expected to master at least the following Ansible topics:

- Basic Concepts
- Roles
- Working With Playbooks

Feel free to share your feedback and report issues. The contributions to the project are welcome.

2.2 Installation

The most convenient way how to install an Ansible role is to use Ansible Galaxy CLI `ansible-galaxy`. The utility comes with the standard Ansible package and provides the user with a simple interface to the Ansible Galaxy's services. For example take a look at the current status of the role

```
shell> ansible-galaxy info vbotka.config_light
```

and install it

```
shell> ansible-galaxy install vbotka.config_light
```

See also:

- To install specific versions from various sources see Installing content.
- Take a look at other roles `shell> ansible-galaxy search --author=vbotka`

2.3 Playbook

Below is simple playbook that applies this role at single host srv.example.com (2)

```
shell> cat config-light.yml
- hosts: srv.example.com
  gather_facts: true
  connection: ssh
  remote_user: admin
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - vbotka.config_light
```

Note: `gather_facts: true` (3) must be set to gather facts needed to evaluate OS specific options of the role. For example to install packages the variable *ansible_os_family* is needed to select the appropriate Ansible module.

See also:

- For details see Connection Plugins (4-5)
- and Understanding Privilege Escalation (6-8).

2.4 Debug

To see additional debug information enable debug output in the configuration

```
cl_debug: true
```

, or set the extra variable in the command:

```
shell> ansible-playbook config-light.yml -e 'cl_debug=true'
```

Note: The debug output of this role is optimized for the `yaml` callback plugin. Set this plugin for example in the environment `shell> export ANSIBLE_STDOUT_CALLBACK=yaml`.

See also:

- Playbook Debugger

2.5 Tags

The tags provide very useful tool to run selected tasks of the role. To see what tags are available list the tags of the role with the command:

```
shell> ansible-playbook config-light.yml --list-tags

playbook: config-light.yml

  play #1 (srv.example.com): srv.example.com        TAGS: []
  TASK TAGS: [always, cl_debug, cl_files, cl_packages, cl_sanity, cl_services, cl_
↪setup, cl_states, cl_vars]
```

For example display the list of the variables and their values with the tag `cl_debug` (when the debug is enabled `cl_debug: true`). With this tag specified `-t cl_debug` all imported tasks before the task *debug.yml* will also run because of the tag `always` (when sanity testing is enabled `cl_sanity: true` and setup is enabled `cl_setup: true`). This is the default. See *main.yml*.

```
shell> ansible-playbook config-light.yml -t cl_debug
```

See what packages will be installed

```
shell> ansible-playbook config-light.yml -t cl_packages --check
```

Install packages and exit the play

```
shell> ansible-playbook config-light.yml -t cl_packages
```

2.6 Variables

In this chapter we describe role's default variables stored in the directory `defaults`.

See also:

- Ansible variable precedence: Where should I put a variable?

2.6.1 Default variables

Most of the variables are self-explaining. There are four very important variables `cl_handlers`, `cl_packages`, `cl_states`, `cl_services`, and `cl_files` (11-15). These dictionaries which comprise the configuration data of handlers, packages, services, and files will be explained in details. By default these dictionaries are empty.

Best practice is to provide the data either in *host_vars* and *group_vars* or as a files in the directories `cl_handlersd_dir`, `cl_packagesd_dir`, `cl_statesd_dir`, `cl_servicesd_dir`, and `cl_filesd_dir` (22-26). Both methods can be applied at the same time. The variables will be assembled and combined by the tasks `vars_handlers.yml`, `vars_packages.yml`, `vars_states.yml`, `vars_services.yml`, and `vars_files.yml`. The assembled dictionaries customized for each host in the play will be stored in the host-specific files `cl_packagesd`, `cl_statesd`, `cl_servicesd`, and `cl_filesd` (35-38). The variable `cl_handlers` is not host-specific because the handlers will be create at the controller (localhost). Assembled dictionary `cl_handlers` will be stored in the file `cl_handlersd` (34). Take a look at the directory `cl_dira` (33) to see assembled data.

By default the base of the directories is `role_path` (21). The user is expected to put the configuration data to more suitable directory, e.g., to `playbook_dir` directory.

[defaults/main.yml]

```
1  ---
2  # defaults for config_light
3
4  cl_sanity: true               # Import tasks/sanity.yml
5  cl_setup: true                # Import tasks/setup.yml
6  cl_install: true              # Install packages or ports
7  cl_debug: false               # Print debug output
8  cl_backup: false              # Backup files
9
10 # Combine assembled data with these variables
11 cl_handlers: {}
12 cl_packages: {}
13 cl_services: {}
14 cl_files: {}
15 cl_states: {}
16
17 # Assemble data from these directories
18 # cl_dird_owner: root         # no default
19 # cl_dird_group: adm          # no default
20 cl_dird_dmode: "0775"         # default very permissive, restrict if necessary
21 cl_dird: "{{ role_path }}/files"
22 cl_handlersd_dir: "{{ cl_dird }}/handlers.d"
```

(continues on next page)

(continued from previous page)

```
cl_packagesd_dir: "{{ cl_dird }}/packages.d"
cl_servicesd_dir: "{{ cl_dird }}/services.d"
cl_filesd_dir: "{{ cl_dird }}/files.d"
cl_statesd_dir: "{{ cl_dird }}/states.d"

# Assemble inventory_hostname data into these files
# cl_dira_owner: root        # no default
# cl_dira_group: adm         # no default
cl_dira_dmode: "0775"        # default very permissive, restrict if necessary
cl_dira_fmode: "0664"        # default very permissive, restrict if necessary
cl_dira: "{{ cl_dird }}/assemble"
cl_handlersd: "{{ cl_dira }}/handlersd"  # localhost; not inventory_hostname specific
cl_packagesd: "{{ cl_dira }}/packagesd.{{ inventory_hostname }}"
cl_servicesd: "{{ cl_dira }}/servicesd.{{ inventory_hostname }}"
cl_filesd: "{{ cl_dira }}/filesd.{{ inventory_hostname }}"
cl_statesd: "{{ cl_dira }}/statesd.{{ inventory_hostname }}"
cl_assemble_regexp: '^(.*)[^~]$'   # Any string but terminated with ~
# cl_assemble_validate: 'ansible-lint -x 205 %s'  # no default

# Handlers
cl_handlers_validate: 'ansible-lint %s'

# OS common
install_retries: 10
install_delay: 5

# FreeBSD
freebsd_install_method: packages
# freebsd_install_method: ports
freebsd_use_packages: true

# EOF
...
```

Warning: Defaults of the variables *cl_dira_dmode* (31) and *cl_dira_fmode* (32) are very permissive. These are the permissions to access the assembled dictionaries. Restrict the permissions if these dictionaries might comprise classified data.

<TODO: complete description of all default variables>

2.6.2 cl_handlers - dictionary with handlers

- *Synopsis*
- *Parameters*
- *Example*
- *See Also*
- *Notes*

Synopsis

The variable *cl_handlers* is a dictionary of the handlers. Structure of the dictionary depends on the template that is used to create the file with the handlers. For example, the structure below can be used with the template *handlers-auto1.yml.j2*.

Parameters

Parameter	*Type*	*Comments*
template	*string* `required`	Template filename
handler	*string* `required`	Name of the handler
module	*string* `required`	Ansible module in handler
params	*list* `required`	Ansible module parameters
conditions	*list*	List of conditions

Example

FreeBSD handlers for postfix

[contrib/postfix/conf-light/handlers.d/postfix-freebsd]

```
postfix_freebsd:
  template: handlers-auto1.yml.j2
  params:

    - handler: 'enable and start postfix'
      module: service
      params:
        - 'name: postfix'
        - 'state: started'
        - 'enabled: true'
```

(continues on next page)

(continued from previous page)

```
11
12      - handler: 'disable and stop postfix'
13        module: service
14        params:
15          - 'name: postfix'
16          - 'state: stopped'
17          - 'enabled: false'
18
19      - handler: 'reload postfix'
20        module: service
21        params:
22          - 'name: postfix'
23          - 'state: reloaded'
24        conditions:
25          - '- cl_service_postfix_enable|bool'
26
27      - handler: 'restart postfix'
28        module: service
29        params:
30          - 'name: postfix'
31          - 'state: restarted'
32        conditions:
33          - '- cl_service_postfix_enable|bool'
34
35      - handler: 'postfix check'
36        module: command
37        params:
38          - 'cmd: /usr/local/sbin/postfix check'
39
40      - handler: 'newaliases'
41        module: command
42        params:
43          - 'cmd: /usr/bin/newaliases'
44
45  #     - handler: 'postmap smtp sasl passwords'
46  #       module: command
47  #       params:
48  #         - 'cmd: /usr/local/sbin/postmap {{ postfix_main_cf_smtp_sasl_password_maps }}
    ↪'
49
50  #     - handler: 'postmap virtual aliases'
51  #       module: command
52  #       params:
53  #         - cmd: /usr/local/sbin/postmap {{ postfix_virtual }}'
```

See Also

See also:

- See vars-handlers.yml how the variable *cl_handlers* is combined with the content of the directory *cl_handlersd_dir*.
- See setup.yml how the handlers are created.
- For details see the template handlers-auto1.yml.j2.

Notes

Note: The template *handlers-auto1.yml.j2* is available in the role's directory `templates`. The user is expected to create new templates when needed. Feel free to change the structure of the data and to create new templates that might fit the purpose better. Feel free to contribute new templates and configuration examples to the project.

2.6.3 cl_packages - dictionary with packages or BSD ports

Synopsis

The variable *cl_packages* is a dictionary of the packages, or BSD ports to be installed.

Parameters

Parameter	*Type*	*Comments*
name	*string* `required`	Package or BSD port

Example

FreeBSD package for postfix

[contrib/postfix/conf-light/packages.d/postfix]

```
postfix:
  name: 'mail/postfix'
```

See Also

See also:

- See vars-packages.yml how the variable *cl_packages* is combined with the content of the directory *cl_packagesd_dir*.
- See packages.yml how the packages or BSD ports are installed.

2.6.4 cl_states - dictionary of files' states

Synopsis

The variable *cl_states* is a dictionary of the files' states.

Parameters

Parameter	*Type*	*Comments*
state	*string* `required`	State of the filename
path	*string* `required`	Path to file
owner	*string*	Owner of the file
group	*string*	Group of the file
mode	*string*	Mode of the file

<TODO: complete parameters. See tasks/states.yml>

Example

File's states

[contrib/lighttpd/conf-light/states.d/lighttpd-server-document-root]

```
lighttpd_server_document_root:
  state: directory
  path: '{{ cl_lighttpd_server_document_root }}'
  owner: '{{ cl_lighttpd_server_username }}'
  group: '{{ cl_lighttpd_server_groupname }}'
  mode: '0640'
```

See Also

See also:

- See vars-states.yml how the variable *cl_states* is combined with the content of the directory *cl_statesd_dir*.
- See states.yml how the file's states are set.

2.6.5 cl_services - dictionary with services

Synopsis

The variable *cl_services* is a dictionary with the services managed by this role.

Parameters

Parameter	*Type*	*Comments*
name	*string* `required`	Service
state	*string*	State of the service default: started
enabled	*boolean*	Start of boot default: true

Example

FreeBSD services for postfix

[contrib/postfix/conf-light/service.d/postfix]

```
postfix:
  name: 'postfix'
  state: '{{ cl_service_postfix_state }}'
  enabled: '{{ cl_service_postfix_enable }}'
```

[contrib/postfix/conf-light/service.d/sendmail]

```
sendmail:
  name: 'sendmail'
  state: '{{ cl_service_sendmail_state }}'
  enabled: '{{ cl_service_sendmail_enable }}'
```

See Also

See also:

- See vars-services.yml how the variable *cl_services* is combined with the content of the directory *cl_servicesd_dir*.
- See services.yml how the services are configured.

2.6.6 cl_files - dictionary with files

Synopsis

The variable *cl_files* is a dictionary of the files that shall be created or modified by this role. It's optional which Ansible module will be used to create or modify a file and more options can be applied to create and modify the same file. For example, it's possible to create a file by the Ansible module *template* and modify it with the module *lineinfile* later. Several options are available:

1. template: If the attribute *template* is defined in the dictionary
2. lineinfile: If the attribute *lines* is defined in the dictionary
3. blockinfile: If the attribute *blocks* is defined in the dictionary
4. ini_file: If the attribute *ini* is defined in the dictionary

Multiple options, when used to create or modify a file, will be applied in this order.

See Also

See also:

- See vars-files.yml how the variable *cl_files* is combined with the content of the directory *cl_filesd_dir*.
- See files.yml how the files are created and modified.
- See files-create-backup.yml how the backups are created (when enabled by *cl_backup*).
- See files-delete-backup.yml how the backup files are deleted when the files haven't been modified.

Parameters for template

Parameter	*Type*	*Comments*
path	*string* `required`	Path to file
template	*string* `required`	Template filename
owner	*string*	Owner of the file
group	*string*	Group of the file
mode	*string*	Mode of the file
force	*boolean*	Replace when different default: true
validate	*string*	Command to validate file
handlers	*list*	List of handlers

Example of template

File `/etc/mail/mailer.conf` for postfix

[contrib/postfix/conf-light/files.d/mailer-conf]

```
mailerconf:
  path: '/etc/mail/mailer.conf'
  force: true
  owner: 'root'
  group: 'wheel'
  mode: '0644'
  template: 'mailer.conf.j2'
```

See Also

See also:

- See files-template.yml how the files are modified or created by the Ansible module `template`.

Notes

Note: There are couple of templates ready to be used in the directory `templates`. The user is expected to create new templates when needed. Feel free to contribute new templates to the project.

Parameters for lineinfile

Parameter	*Type*	*Comments*
path	*string* `required`	Path to file
lines	*list* `required`	List of regexp and lines
owner	*string*	Owner of the file
group	*string*	Group of the file
mode	*string*	Mode of the file
create	*boolean*	Create if does not exist default: false
validate	*string*	Command to validate file
handlers	*list*	List of handlers

Example of lineinfile

File `/usr/local/etc/postfix/main.cf` for postfix

[contrib/postfix/conf-light/files.d/postfix-main.cf]

```
postfix_main_cf:
  path: '/usr/local/etc/postfix/main.cf'
  create: true
  owner: 'root'
  group: 'wheel'
  mode: '0644'
  handlers:
    - 'postfix_freebsd reload postfix'
  lines:
    - regexp: '^myhostname\s*=\s*(.*)$'
      line: 'myhostname = {{ cl_myhostname }}'
```

See Also

See also:

- See files-lineinfile.yml how the files are modified or created by the Ansible module `lineinfile`.

Parameters for blockinfile

Parameter	*Type*	*Comments*
path	*string* `required`	Path to file
blocks	*list* `required`	List of blocks and markers
owner	*string*	Owner of the file
group	*string*	Group of the file
mode	*string*	Mode of the file
create	*boolean*	Create if does not exist default: false
validate	*string*	Command to validate file
handlers	*list*	List of handlers

Example of blockinfileinfile

<TODO: No example yet>

See Also

See also:

- See files-blockinfile.yml how the files are modified or created by the Ansible module `blockinfile`.

Parameters for ini_file

Parameter	*Type*	*Comments*
path	*string* `required`	Path to file
ini	*list* `required`	List of {section,option, value} dictionaries
owner	*string*	Owner of the file
group	*string*	Group of the file
mode	*string*	Mode of the file
create	*boolean*	Create if does not exist default: true
handlers	*list*	List of handlers

Example of ini_file

<TODO: No example yet>

See Also

See also:

- See files-inifile.yml how the files are modified or created by the Ansible module `ini_file`.

2.7 Best practice

Create variables. Take a look at directory `conf-light/assemble/` what files were created.

```
shell> ansible-playbook config-light.yml -t cl_vars
```

Test sanity. Then disable this task `cl_sanity: false` to speedup the playbook.:

```
shell> ansible-playbook config-light.yml -t cl_sanity
```

Create handlers. Take a look at directory `roles/vbotka.config_light/handlers` what handlers were created. Run this task once to create the handlers. Then disable this task `cl_setup: false` to speedup the playbook.

```
shell> ansible-playbook config-light.yml -t cl_setup
```

Display variables. Display the variables for debug if needed. Then disable this task `cl_debug: false` to speedup the playbook.

```
shell> ansible-playbook config-light.yml -t cl_debug
```

Install packages. Then disable this task `cl_install: false` to speedup the playbook.

```
shell> ansible-playbook config-light.yml -t cl_packages
```

Set files' states.

```
shell> ansible-playbook config-light.yml -t cl_states
```

Create and modify files

```
shell> ansible-playbook config-light.yml -t cl_files
```

Configure services

```
shell> ansible-playbook config-light.yml -t cl_services
```

The role and the configuration data in the examples are idempotent. Once the application is installed and configured there should be no changes reported by *ansible-playbook* when running the playbook repeatedly. Disable setup, sanity, debug, and install to speedup the playbook

```
shell> ansible-playbook config-light.yml
```

CHAPTER

THREE

ANNOTATED SOURCE CODE

Table of Contents

3.1 Tasks

3.1.1 main.yml

Synopsis: Tasks of the playbook.

Description of the task.

[main.yml]

```
---
# tasks for config_light

- import_tasks: setup.yml
  when: cl_setup|bool
  delegate_to: localhost
  run_once: true
  tags: [cl_setup, always]

- import_tasks: vars.yml
  tags: [cl_vars, always]

- import_tasks: sanity.yml
  when: cl_sanity|bool
  tags: [cl_sanity, always]

- import_tasks: debug.yml
  when: cl_debug|bool
  tags: [cl_debug, always]

- import_tasks: packages.yml
  when: cl_install|bool
  tags: cl_packages

- import_tasks: states.yml
  tags: cl_states

- import_tasks: files.yml
  tags: cl_files

- import_tasks: services.yml
  tags: cl_services

# EOF
...
```

3.1.2 vars.yml

Synopsis: Combine dictionaries of packages, services and files.

Description of the task.

[vars.yml]

```
---

```

(continues on next page)

(continued from previous page)

```
- name: "vars: Debug"
  when: cl_debug|bool
  vars:
    msg: |
      cl_handlersd_dir [{{ cl_handlersd_dir }}]
      cl_packagesd_dir [{{ cl_packagesd_dir }}]
      cl_statesd_dir [{{ cl_statesd_dir }}]
      cl_servicesd_dir [{{ cl_servicesd_dir }}]
      cl_filesd_dir [{{ cl_filesd_dir }}]
      cl_dira [{{ cl_dira }}]
      cl_handlersd [{{ cl_handlersd }}]
      cl_packagesd [{{ cl_packagesd }}]
      cl_statesd [{{ cl_statesd }}]
      cl_servicesd [{{ cl_servicesd }}]
      cl_filesd [{{ cl_filesd }}]
  debug:
    msg: "{{ msg.split('\n') }}"

- name: "vars: Packages"
  import_tasks: vars-packages.yml

- name: "vars: States"
  import_tasks: vars-states.yml

- name: "vars: Services"
  import_tasks: vars-services.yml

- name: "vars: Files"
  import_tasks: vars-files.yml

# EOF
...
```

3.1.3 vars-packages.yml

Synopsis: Combine dictionaries of packages.

Description of the task.

[vars-packages.yml]

```
---

- name: "vars-packages: Assemble packages to {{ cl_packagesd }}"
  assemble:
    regexp: "{{ cl_assemble_regexp }}"
    src: "{{ cl_packagesd_dir }}"
    dest: "{{ cl_packagesd }}"
    owner: "{{ cl_dira_owner|default(omit) }}"
    group: "{{ cl_dira_group|default(omit) }}"
    mode: "{{ cl_dira_fmode }}"
    validate: "{{ cl_assemble_validate|default(omit) }}"
  delegate_to: localhost

- name: "vars-packages: Include files from {{ cl_packagesd }} to cl_packagesd_items"
  include_vars:
```

(continues on next page)

(continued from previous page)

```
    file: "{{ cl_packagesd }}"
    name: cl_packagesd_items

- name: "vars-packages: Combine cl_packages with cl_packagesd_items"
  set_fact:
    cl_packages: "{{ cl_packages|combine(cl_packagesd_items|default({})) }}"

- name: "vars-packages: Debug"
  debug:
    var: cl_packages
  when: cl_debug|bool

# EOF
...
```

3.1.4 vars-states.yml

Synopsis: Combine dictionaries of files' states.

Description of the task.

[vars-states.yml]

```
---

- name: "vars-states: Assemble states to {{ cl_states }}"
  assemble:
    regexp: "{{ cl_assemble_regexp }}"
    src: "{{ cl_statesd_dir }}"
    dest: "{{ cl_statesd }}"
    owner: "{{ cl_dira_owner|default(omit) }}"
    group: "{{ cl_dira_group|default(omit) }}"
    mode: "{{ cl_dira_fmode }}"
    validate: "{{ cl_assemble_validate|default(omit) }}"
  delegate_to: localhost

- name: "vars-states: Include files from {{ cl_statesd }} to cl_statessd_items"
  include_vars:
    file: "{{ cl_statesd }}"
    name: cl_statesd_items

- name: "vars-states: Combine cl_statess with cl_statesd_items"
  set_fact:
    cl_states: "{{ cl_states|combine(cl_statesd_items|default({})) }}"

- name: "vars-states: Debug"
  debug:
    var: cl_states
  when: cl_debug|bool

# EOF
...
```

3.1.5 vars-services.yml

Synopsis: Combine dictionaries of services.

Description of the task.

[vars-services.yml]

```yaml
---

- name: "vars-services: Assemble services to {{ cl_servicesd }}"
  assemble:
    regexp: "{{ cl_assemble_regexp }}"
    src: "{{ cl_servicesd_dir }}"
    dest: "{{ cl_servicesd }}"
    owner: "{{ cl_dira_owner|default(omit) }}"
    group: "{{ cl_dira_group|default(omit) }}"
    mode: "{{ cl_dira_fmode }}"
    validate: "{{ cl_assemble_validate|default(omit) }}"
  delegate_to: localhost

- name: "vars-services: Include files from {{ cl_servicesd }} to cl_cervicesd_items"
  include_vars:
    file: "{{ cl_servicesd }}"
    name: cl_servicesd_items

- name: "vars-services: Combine cl_services with cl_servicesd_items"
  set_fact:
    cl_services: "{{ cl_services|combine(cl_servicesd_items|default({})) }}"

- name: "vars-services: Debug"
  debug:
    var: cl_services
  when: cl_debug|bool

# EOF
...
```

3.1.6 vars-files.yml

Synopsis: Combine dictionaries of files.

Description of the task.

[vars-files.yml]

```yaml
---

- name: "vars-files: Assemble files to {{ cl_filesd }}"
  assemble:
    regexp: "{{ cl_assemble_regexp }}"
    src: "{{ cl_filesd_dir }}"
    dest: "{{ cl_filesd }}"
    owner: "{{ cl_dira_owner|default(omit) }}"
    group: "{{ cl_dira_group|default(omit) }}"
    mode: "{{ cl_dira_fmode }}"
    validate: "{{ cl_assemble_validate|default(omit) }}"
  delegate_to: localhost
```

(continues on next page)

(continued from previous page)

```
13
14 - name: "vars-files: Include files from {{ cl_filesd }} to cl_filesd_items"
15   include_vars:
16     file: "{{ cl_filesd }}"
17     name: cl_filesd_items
18
19 - name: "vars-files: Combine cl_files with cl_filesd_items"
20   set_fact:
21     cl_files: "{{ cl_files|combine(cl_filesd_items|default({})) }}"
22
23 - name: "vars-files: Debug"
24   debug:
25     var: cl_files
26   when: cl_debug|bool
27
28 # EOF
29 ...
```

3.1.7 sanity.yml

Synopsis: Test sanity.

Description of the task.

[sanity.yml]

```
1  ---
2
3  - name: "sanity: Directories to assemble data from must exist"
4    block:
5      - name: "sanity: Directories to assemble data from must exist"
6        debug:
7          msg: "{{ msg.split('\n')[:-1] }}"
8        vars:
9          msg: |
10           Directories to assemble data from do not exist. End of host.
11           Hint: Double check existence of the directories
12           {{ cl_handlersd_dir }}
13           {{ cl_packagesd_dir }}
14           {{ cl_statesd_dir }}
15           {{ cl_servicesd_dir }}
16           {{ cl_filesd_dir }}
17     - name: "sanity: End of host"
18       meta: end_host
19   when:
20     - cl_handlersd_dir is not exists or
21       cl_packagesd_dir is not exists or
22       cl_statesd_dir is not exists or
23       cl_servicesd_dir is not exists or
24       cl_filesd_dir is not exists
25
26 - name: "sanity: Check mode not possible without assembled data"
27   block:
28     - name: "sanity: Check mode not possible without assembled data"
29       debug:
30         msg: "{{ msg.split('\n')[:-1] }}"
```

(continues on next page)

(continued from previous page)

```
      vars:
        msg: |
          Check mode not possible without assembled data. End of host.
          Hint: Assemble the variables first.
          Run: ansible-playbook playbook.yml -t cl_vars
    - name: "sanity: End of host"
      meta: end_host
  when:
    - ansible_check_mode
    - cl_packagesd is not exists or
      cl_statesd is not exists or
      cl_servicesd is not exists or
      cl_filesd is not exists

# EOF
...
```

3.1.8 setup.yml

Synopsis: Setup. Create dirs. Create handlers.

Description of the task.

[setup.yml]

```
---

# directories
- name: "setup: Create directories in {{ cl_dird }}"
  file:
    state: directory
    path: "{{ item }}"
    owner: "{{ cl_dird_owner|default(omit) }}"
    group: "{{ cl_dird_group|default(omit) }}"
    mode: "{{ cl_dird_dmode }}"
  loop:
    - "{{ cl_dird }}"
    - "{{ cl_handlersd_dir }}"
    - "{{ cl_packagesd_dir }}"
    - "{{ cl_servicesd_dir }}"
    - "{{ cl_filesd_dir }}"
    - "{{ cl_statesd_dir }}"

- name: "setup: Create directory {{ cl_dira }}"
  file:
    state: directory
    path: "{{ cl_dira }}"
    owner: "{{ cl_dira_owner|default(omit) }}"
    group: "{{ cl_dira_group|default(omit) }}"
    mode: "{{ cl_dira_dmode }}"

# handlers
- name: "setup: Assemble handlers"
  import_tasks: vars-handlers.yml

- name: "setup: Create handlers"
```

(continues on next page)

(continued from previous page)

```
  template:
    dest: "{{ role_path }}/handlers/handlers-auto-{{ item.key }}.yml"
    src: "{{ item.value.template }}"
    validate: "{{ cl_handlers_validate|default(omit) }}"
    backup: "{{ cl_backup }}"
  loop: "{{ cl_handlers|dict2items }}"
  loop_control:
    label: "{{ role_path }}/handlers/handlers-auto-{{ item.key }}.yml"

- name: "setup: Include handlers"
  lineinfile:
    path: "{{ role_path }}/handlers/main.yml"
    line: "- import_tasks: handlers-auto-{{ item.key }}.yml"
    validate: "{{ cl_handlers_validate|default(omit) }}"
    backup: "{{ cl_backup }}"
    create: true
  loop: "{{ cl_handlers|dict2items }}"
  loop_control:
    label: "{{ role_path }}/handlers/handlers-auto-{{ item.key }}.yml"

# EOF
...
```

3.1.9 vars-handlers.yml

Synopsis: Combine dictionaries of handlers.

Description of the task.

[vars-handlers.yml]

```
---

- name: "vars-handlers: Assemble handlers to {{ cl_handlersd }}"
  assemble:
    regexp: "{{ cl_assemble_regexp }}"
    src: "{{ cl_handlersd_dir }}"
    dest: "{{ cl_handlersd }}"
    owner: "{{ cl_dira_owner|default(omit) }}"
    group: "{{ cl_dira_group|default(omit) }}"
    mode: "{{ cl_dira_fmode }}"
    validate: "{{ cl_assemble_validate|default(omit) }}"
  delegate_to: localhost

- name: "vars-handlers: Include files from {{ cl_handlersd }} to cl_handlersd_items"
  include_vars:
    file: "{{ cl_handlersd }}"
    name: cl_handlersd_items

- name: "vars-handlers: Combine cl_handlers with cl_handlersd_items"
  set_fact:
    cl_handlers: "{{ cl_handlers|combine(cl_handlersd_items|default({})) }}"

- name: "vars-handlers: Debug"
  debug:
    var: cl_handlers
```

(continues on next page)

(continued from previous page)

```
  when: cl_debug|bool

# EOF
...
```

3.1.10 debug.yml

Synopsis: Display variables.

Description of the task.

[debug.yml]

```
---

- name: "debug: Config Light"
  vars:
    msg: |
      ansible_os_family [{{ ansible_os_family }}]
      ansible_distribution [{{ ansible_distribution }}]
      ansible_distribution_major_version [{{ ansible_distribution_major_version }}]
      ansible_distribution_version [{{ ansible_distribution_version }}]
      ansible_distribution_release [{{ ansible_distribution_release }}]
      ansible_python_version [{{ ansible_python_version }}]

      cl_sanity [{{ cl_sanity }}]
      cl_setup [{{ cl_setup }}]
      cl_install [{{ cl_install }}]
      cl_backup [{{ cl_backup }}]

      cl_handlers
      {{ cl_handlers|to_nice_yaml }}
      cl_packages
      {{ cl_packages|to_nice_yaml }}
      cl_services
      {{ cl_services|to_nice_yaml }}
      cl_files
      {{ cl_files|to_nice_yaml }}
      cl_states
      {{ cl_states|to_nice_yaml }}

      cl_dird [{{ cl_dird }}]
      cl_dird_owner [{{ cl_dira_owner|default('UNDEFINED') }}]
      cl_dird_group [{{ cl_dira_group|default('UNDEFINED') }}]
      cl_dird_dmode [{{ cl_dira_dmode }}]

      cl_handlersd_dir [{{ cl_handlersd_dir }}]
      cl_packagesd_dir [{{ cl_packagesd_dir }}]
      cl_servicesd_dir [{{ cl_servicesd_dir }}]
      cl_filesd_dir [{{ cl_filesd_dir }}]
      cl_statesd_dir [{{ cl_statesd_dir }}]

      cl_dira [{{ cl_dira }}]
      cl_dira_owner [{{ cl_dira_owner|default('UNDEFINED') }}]
      cl_dira_group [{{ cl_dira_group|default('UNDEFINED') }}]
      cl_dira_dmode [{{ cl_dira_dmode }}]
```

(continues on next page)

(continued from previous page)

```
      cl_dira_fmode [{{ cl_dira_fmode }}]
      cl_assemble_regexp [{{ cl_assemble_regexp }}]

      cl_handlersd [{{ cl_handlersd }}]
      cl_packagesd [{{ cl_packagesd }}]
      cl_servicesd [{{ cl_servicesd }}]
      cl_filesd [{{ cl_filesd }}]
      cl_statesd [{{ cl_statesd }}]

      cl_assemble_validate [{{ cl_assemble_validate|default('UNDEFINED') }}]
      cl_handlers_validate [{{ cl_handlers_validate|default('UNDEFINED') }}]

      install_retries [{{ install_retries }}]
      install_delay [{{ install_delay }}]

      freebsd_install_method [{{ freebsd_install_method }}]
      freebsd_use_packages [{{ freebsd_use_packages }}]

  debug:
    msg: "{{ msg.split('\n') }}"

# EOF
...
```

3.1.11 packages.yml

Synopsis: Install packages.

Description of the task.

[packages.yml]

```
---

# FreeBSD ----------------------------------------------------------------
- name: "packages: Install packages FreeBSD"
  block:
    - name: "packages: Install packages FreeBSD"
      pkgng:
        name: "{{ item.value.name }}"
        state: "{{ item.value.state|default(omit) }}"
        annotation: "{{ item.value.annotation|default(omit) }}"
        autoremove: "{{ freebsd_pkgng_autoremove|default(omit) }}"
        cached: "{{ freebsd_pkgng_cached|default(omit) }}"
        chroot: "{{ freebsd_pkgng_chroot|default(omit) }}"
        jail: "{{ freebsd_pkgng_jail|default(omit) }}"
        pkgsite: "{{ freebsd_pkgng_pkgsite|default(omit) }}"
        rootdir: "{{ freebsd_pkgng_rootdir|default(omit) }}"
      loop: "{{ cl_packages|dict2items }}"
      loop_control:
        label: "{{ item.key }}"
      register: result
      until: result is succeeded
      retries: "{{ install_retries }}"
      delay: "{{ install_delay }}"
    - name: "packages: Debug FreeBSD packages"
```

(continues on next page)

(continued from previous page)

```
      debug:
        var: result
      when: cl_debug|bool
  when:
    - ansible_os_family == "FreeBSD"
    - freebsd_install_method|lower == "packages"

- name: "packages: Install ports FreeBSD"
  block:
    - name: "packages: Install ports FreeBSD"
      portinstall:
        name: "{{ item.value.name }}"
        state: "{{ item.value.state|default(omit) }}"
        use_packages: "{{ item.value.use_packages|default(freebsd_use_packages) }}"
      loop: "{{ cl_packages|dict2items }}"
      loop_control:
        label: "{{ item.key }}"
      register: result
      until: result is succeeded
      retries: "{{ install_retries }}"
      delay: "{{ install_delay }}"
    - name: "packages: Debug FreeBSD ports"
      debug:
        var: result
      when: cl_debug|bool
  when:
    - ansible_os_family == "FreeBSD"
    - freebsd_install_method|lower == "ports"

# Linux ------------------------------------------------------------------
- name: "packages: Install packages Linux"
  block:
    - name: "packages: Install packages Linux"
      package:
        name: "{{ item.value.name }}"
        state: "{{ item.value.state|default('present') }}"
        use: "{{ item.value.use|default('auto') }}"
      loop: "{{ cl_packages|dict2items }}"
      loop_control:
        label: "{{ item.key }}"
      register: result
      until: result is succeeded
      retries: "{{ install_retries }}"
      delay: "{{ install_delay }}"
    - name: "packages: Debug Linux"
      debug:
        var: result
      when: cl_debug|bool
  when: ansible_os_family == "RedHat" or
        ansible_os_family == "Debian"

# EOF
...
```

3.1.12 states.yml

Synopsis: Configure states of files.

Description of the task.

[states.yml]

```
---

- name: "states: Apply states"
  file:
    state: "{{ item.value.state }}"
    path: "{{ item.value.path }}"
    src: "{{ item.value.src|default(omit) }}"
    owner: "{{ item.value.owner|default(omit) }}"
    group: "{{ item.value.group|default(omit) }}"
    mode: "{{ item.value.mode|default(omit) }}"
    attributes: "{{ item.value.attributes|default(omit) }}"
    recurse: "{{ item.value.recurse|default(omit) }}"
    force: "{{ item.value.force|default(omit) }}"
    follow: "{{ item.value.follow|default(omit) }}"
    access_time: "{{ item.value.access_time|default(omit) }}"
    access_time_format: "{{ item.value.access_time_format|default(omit) }}"
    modification_time: "{{ item.value.modification_time|default(omit) }}"
    modification_time_format: "{{ item.value.modification_time_format|default(omit) }}
↪"
    unsafe_writes: "{{ item.value.unsafe_writes|default(omit) }}"
  loop: "{{ cl_states|dict2items }}"
  loop_control:
    label: "{{ item.value.path }}"

# EOF
...
```

3.1.13 files.yml

Synopsis: Create or modify files.

Description of the task.

[files.yml]

```
---

- name: "files: Create backup"
  import_tasks: files-create-backup.yml
  when: cl_backup|bool

- name: "files: Template"
  import_tasks: files-template.yml

- name: "file: Lineinfile"
  import_tasks: files-lineinfile.yml

- name: "file: Blockinfile"
  import_tasks: files-blockinfile.yml

```

(continues on next page)

(continued from previous page)

```yaml
- name: "files: INI file"
  import_tasks: files-inifile.yml

- name: "file: Delete backup"
  import_tasks: files-delete-backup.yml
  when:
    - cl_backup|bool
    - not ansible_check_mode

# EOF
...
```

3.1.14 files-create-backup.yml

Synopsis: Create backup files.

Description of the task.

[files-create-backup.yml]

```yaml
---

- name: "file-create-backup: Create time-stamp"
  set_fact:
    cl_timestamp: "{{ '%Y-%m-%d_%H_%M_%S'|strftime }}"

- name: "file-create-backup: Stat {{ item.path }}"
  stat:
    path: "{{ item.path }}"
  loop: "{{ cl_files.values() }}"
  loop_control:
    label: "{{ item.path }}"
  register: result

- name: "file-create-backup: Debug result"
  debug:
    var: result
  when: cl_debug|bool

- name: "file-create-backup: Create backup files"
  copy:
    remote_src: true
    src: "{{ item.item.path }}"
    dest: "{{ item.item.path }}_{{ cl_timestamp }}.bak"
    mode: "preserve"
  loop: "{{ result.results }}"
  loop_control:
    label: "{{ item.item.path }}"
  when: item.stat.exists
  changed_when: false

# EOF
...
```

3.1.15 files-delete-backup.yml

Synopsis: Delete backup files if the files haven't been modified.

Description of the task.

[files-delete-backup.yml]

```
---

- name: "files-delete-backup: Delete backup files that did not change"
  when: cl_backup|bool
  file:
    state: absent
    path: "{{ item }}_{{ cl_timestamp }}.bak"
  loop: "{{ cl_files.values()|json_query('[].path')|
            difference(
            cl_results_template.results|default([])|
            json_query('[?changed==`true`].invocation.module_args.path'))|unique|
            difference(
            cl_results_lines.results|default([])|
            json_query('[?changed==`true`].invocation.module_args.path'))|unique|
            difference(
            cl_results_blocks.results|default([])|
            json_query('[?changed==`true`].invocation.module_args.path'))|unique|
            difference(
            cl_results_inifile.results|default([])|
            json_query('[?changed==`true`].invocation.module_args.path'))|unique
            }}"
  changed_when: false

# EOF
...
```

3.1.16 files-template.yml

Synopsis: Create or modify files by Ansible module `template`.

Description of the task.

[files-template.yml]

```
---
# TODO: Complete parameters
- name: "files-template: Template"
  template:
    dest: "{{ item.path }}"
    src: "{{ item.template }}"
    owner: "{{ item.owner|default(omit) }}"
    group: "{{ item.group|default(omit) }}"
    mode: "{{ item.mode|default(omit) }}"
    force: "{{ item.force|default(omit) }}"
    validate: "{{ item.validate|default(omit) }}"
    # backup: "{{ cl_backup }}"
  loop: "{{ cl_files.values()|selectattr('template', 'defined')|list }}"
  loop_control:
    label: "{{ item.path }}"
  notify: "{{ item.handlers|default(omit) }}"
```

(continues on next page)

(continued from previous page)

```
  register: cl_results_template

- name: "files-template: Debug"
  block:
    - name: Debug cl_results_template
      debug:
        var: cl_results_template
    - name: Debug changed template path
      debug:
        msg: "{{ cl_results_template|default([])|
                 json_query('[?changed==`true`].invocation.module_args.path')
                 }}"
  when: cl_debug|bool

# EOF
...
```

3.1.17 files-lineinfile.yml

Synopsis: Create or modify files by Ansible module `lineinfile`.

Description of the task.

[files-lineinfile.yml]

```
---
# TODO: Complete parameters
- name: "files-lineinfile: Lineinfile"
  lineinfile:
    path: "{{ item.0.path }}"
    regexp: "{{ item.1.regexp }}"
    line: "{{ item.1.line }}"
    owner: "{{ item.0.owner|default(omit) }}"
    group: "{{ item.0.group|default(omit) }}"
    mode: "{{ item.0.mode|default(omit) }}"
    create: "{{ item.0.create|default(omit) }}"
    validate: "{{ item.0.validate|default(omit) }}"
    # backup: "{{ cl_backup }}"
  loop: "{{ cl_files.values()|subelements('lines', skip_missing=true) }}"
  loop_control:
    label: "{{ item.0.path }}"
  notify: "{{ item.0.handlers|default(omit) }}"
  register: cl_results_lines

- name: "files-lineinfile: Debug"
  block:
    - name: Debug cl_results_lines
      debug:
        var: cl_results_lines
    - name: Debug changed lines paths
      debug:
        msg: "{{ cl_results_lines.results|default([])|
                 json_query('[?changed==`true`].invocation.module_args.path')
                 }}"
  when: cl_debug|bool

```

(continues on next page)

(continued from previous page)

```
# EOF
...
```

3.1.18 files-blockinfile.yml

Synopsis: Create or modify files by Ansible module `blockinfile`.

Description of the task.

[files-blockinfile.yml]

```yaml
---
# TODO: Complete parameters
- name: "files-blockinfile: Blockinfile"
  blockinfile:
    path: "{{ item.0.path }}"
    marker: "# {mark} ANSIBLE MANAGED BLOCK {{ item.1.marker }}"
    block: "{{ item.1.block }}"
    owner: "{{ item.0.owner|default(omit) }}"
    group: "{{ item.0.group|default(omit) }}"
    mode: "{{ item.0.mode|default(omit) }}"
    create: "{{ item.0.create|default(omit) }}"
    validate: "{{ item.0.validate|default(omit) }}"
    # backup: "{{ cl_backup }}"
  loop: "{{ cl_files.values()|subelements('blocks', skip_missing=true) }}"
  loop_control:
    label: "{{ item.0.path }}"
  notify: "{{ item.0.handlers|default(omit) }}"
  register: cl_results_blocks

- name: "files-blockinfile: Debug"
  block:
    - name: Debug cl_results_blocks
      debug:
        var: cl_results_blocks
    - name: Debug changed blocks paths
      debug:
        msg: "{{ cl_results_blocks.results|default([])|
                 json_query('[?changed==`true`].invocation.module_args.path')
                 }}"
  when: cl_debug|bool

# EOF
...
```

3.1.19 files-inifile.yml

Synopsis: Create or modify files by Ansible module `ini_infile`.

Description of the task.

[files-inifile.yml]

```yaml
---
# TODO: Complete parameters
- name: "files-inifile: INI files"
  ini_file:
    path: "{{ item.0.path }}"
    section: "{{ item.1.section }}"
    option: "{{ item.1.option }}"
    value: "{{ item.1.value }}"
    owner: "{{ item.0.owner|default(omit) }}"
    group: "{{ item.0.group|default(omit) }}"
    mode: "{{ item.0.mode|default(omit) }}"
    create: "{{ item.0.create|default(omit) }}"
    # backup: "{{ cl_backup }}"
  loop: "{{ cl_files.values()|subelements('ini', skip_missing=true) }}"
  loop_control:
    label: "{{ item.0.path }}"
  notify: "{{ item.0.handlers|default(omit) }}"
  register: cl_results_ini

- name: "files-inifile: Debug"
  block:
    - name: Debug cl_results_ini
      debug:
        var: cl_results_ini
    - name: Debug changed ini paths
      debug:
        msg: "{{ cl_results_ini.results|default([])|
                 json_query('[?changed==`true`].invocation.module_args.path')
                 }}"
  when: cl_debug|bool

# EOF
...
```

3.1.20 services.yml

Synopsis: Configure services.

Description of the task.

[services.yml]

```yaml
---

# FreeBSD ---------------------------------------------------------------------
- name: "services: FreeBSD"
  block:

    - name: "services: Enable service in rc.conf FreeBSD"
      lineinfile:
```

(continues on next page)

(continued from previous page)

```
        dest: "/etc/rc.conf"
        regexp: "^\\s*{{ item.value.name }}_enable\\s*="
        line: "{{ item.value.name }}_enable=\"YES\""
        backup: "{{ cl_backup }}"
      loop: "{{ cl_services|dict2items }}"
      loop_control:
        label: "{{ item.key }}"
      when: "item.value.enabled|default(true)|bool"

    - name: "services: Disable service in rc.conf FreeBSD"
      lineinfile:
        dest: "/etc/rc.conf"
        regexp: "^\\s*{{ item.value.name }}_enable\\s*="
        line: "{{ item.value.name }}_enable=\"NO\""
        backup: "{{ cl_backup }}"
      loop: "{{ cl_services|dict2items }}"
      loop_control:
        label: "{{ item.key }}"
      when: "not item.value.enabled|default(true)|bool"

  when: "ansible_os_family == 'FreeBSD'"

# All ------------------------------------------------------------------------
- name: "services: Manage services"
  block:

    - name: "services: Manage services"
      service:
        name: "{{ item.value.name }}"
        state: "{{ item.value.state|default('started') }}"
        enabled: "{{ item.value.enabled|default(true) }}"
      loop: "{{ cl_services|dict2items }}"
      loop_control:
        label: "{{ item.key }}"

  when: "ansible_os_family == 'RedHat' or
         ansible_os_family == 'Debian' or
         ansible_os_family == 'FreeBSD'"

# EOF
...
```

CHAPTER

FOUR

EXAMPLES

Table of Contents

4.1 FreeBSD Postfix

4.1.1 Handlers

contrib/postfix/conf-light/handlers.d/postfix-freebsd

Synopsis: Create handlers for Postfix.

Use template (2) to create handlers.

[contrib/postfix/conf-light/handlers.d/postfix-freebsd]

```
postfix_freebsd:
  template: handlers-auto1.yml.j2
  params:

    - handler: 'enable and start postfix'
      module: service
      params:
        - 'name: postfix'
        - 'state: started'
        - 'enabled: true'

    - handler: 'disable and stop postfix'
      module: service
      params:
        - 'name: postfix'
        - 'state: stopped'
        - 'enabled: false'

    - handler: 'reload postfix'
      module: service
      params:
        - 'name: postfix'
        - 'state: reloaded'
      conditions:
        - '- cl_service_postfix_enable|bool'

    - handler: 'restart postfix'
      module: service
      params:
        - 'name: postfix'
        - 'state: restarted'
      conditions:
        - '- cl_service_postfix_enable|bool'

    - handler: 'postfix check'
      module: command
      params:
        - 'cmd: /usr/local/sbin/postfix check'

    - handler: 'newaliases'
      module: command
      params:
        - 'cmd: /usr/bin/newaliases'

#    - handler: 'postmap smtp sasl passwords'
```

(continues on next page)

(continued from previous page)

```
#       module: command
#       params:
#         - 'cmd: /usr/local/sbin/postmap {{ postfix_main_cf_smtp_sasl_password_maps }}
↪'

#     - handler: 'postmap virtual aliases'
#       module: command
#       params:
#         - cmd: /usr/local/sbin/postmap {{ postfix_virtual }}'
```

See also:

See *setup.yml* how the handlers are created.

contrib/postfix/conf-light/handlers.d/sendmail-freebsd

Synopsis: Create handlers for Sendmail.

Use template (2) to create handlers.

[contrib/postfix/conf-light/handlers.d/sendmail-freebsd]

```
sendmail_freebsd:
  template: handlers-auto1.yml.j2
  params:

    - handler: 'enable and start sendmail'
      module: service
      params:
        - 'name: sendmail'
        - 'state: started'
        - 'enabled: true'

    - handler: 'disable and stop sendmail'
      module: service
      params:
        - 'name: sendmail'
        - 'state: stopped'
        - 'enabled: false'

    - handler: 'reload sendmail'
      module: service
      params:
        - 'name: sendmail'
        - 'state: reloaded'
      conditions:
        - '- cl_service_sendmail_enable|bool'

    - handler: 'restart sendmail'
      module: service
      params:
        - 'name: sendmail'
        - 'state: restarted'
      conditions:
        - '- cl_service_sendmail_enable|bool'

    - handler: 'start sendmail'
```

(continues on next page)

(continued from previous page)

```
      module: service
      params:
        - 'name: sendmail'
        - 'state: started'

    - handler: 'stop sendmail'
      module: service
      params:
        - 'name: sendmail'
        - 'state: stopped'
```

See also:

See *setup.yml* how the handlers are created.

4.1.2 Packages

contrib/postfix/conf-light/packages.d/postfix

Synopsis: Install Postfix.

Use package or port (2) to install Postfix.

[contrib/postfix/conf-light/packages.d/postfix]

```
postfix:
  name: 'mail/postfix'
```

See also:

See *packages.yml* how the FreeBSD packages or ports are installed.

4.1.3 Services

contrib/postfix/conf-light/services.d/postfix

Synopsis: Configure Postfix service.

Set service (2) state (3). Run the service on boot (4).

[contrib/postfix/conf-light/services.d/postfix]

```
postfix:
  name: 'postfix'
  state: '{{ cl_service_postfix_state }}'
  enabled: '{{ cl_service_postfix_enable }}'
```

See also:

See custom Postfix variables *contrib/postfix/config-light-postfix.yml*. See *services.yml* how the services are configured.

contrib/postfix/conf-light/services.d/sendmail

Synopsis: Configure Sendmail service.

Set service (2) state (3). Do not run the service on boot (4).

[contrib/postfix/conf-light/services.d/sendmail]

```
sendmail:
  name: 'sendmail'
  state: '{{ cl_service_sendmail_state }}'
  enabled: '{{ cl_service_sendmail_enable }}'
```

See also:

See custom Postfix variables *contrib/postfix/config-light-postfix.yml*.

4.1.4 Files

contrib/postfix/config-light-postfix.yml

Synopsis: Custom variables for Postfix.

Put the host-specific variables (6) into the `host_vars`. Optionally other variables might be put into the `group_vars`.

[contrib/postfix/config-light-postfix.yml]

```
---

# 28.4.2. Replace the Default MTA
# https://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/mail-changingmta.html

cl_myhostname: host99.region9.example.com

# conf-light/files.d/mailer-conf
# Execute the Postfix sendmail program, named /usr/local/sbin/sendmail
cl_mailerconf:
  - 'sendmail      /usr/local/sbin/sendmail'
  - 'send-mail     /usr/local/sbin/sendmail'
  - 'mailq         /usr/local/sbin/sendmail'
  - 'newaliases    /usr/local/sbin/sendmail'

# conf-light/files.d/rc-rconf
cl_rcconf_postfix_enable: 'YES'
cl_rcconf_sendmail_enable: 'NO'
cl_rcconf_sendmail_submit_enable: 'NO'
cl_rcconf_sendmail_outbound_enable: 'NO'
cl_rcconf_sendmail_msp_queue_enable: 'NO'

# conf-light/files.d/periodic-conf
cl_periodicconf_daily_clean_hoststat_enable: "NO"
cl_periodicconf_daily_status_mail_rejects_enable: "NO"
cl_periodicconf_daily_status_include_submit_mailq: "NO"
cl_periodicconf_daily_submit_queuerun: "NO"

# Services
cl_service_sendmail_enable: false
```

(continues on next page)

(continued from previous page)

```
cl_service_sendmail_state: 'stopped'
cl_service_postfix_enable: true
cl_service_postfix_state: 'started'

# If you are using SASL, you need to make sure that postfix has access
# to read the sasldb file.  This is accomplished by adding postfix to
# group mail and making the /usr/local/etc/sasldb* file(s) readable by
# group mail (this should be the default for new installs).

# EOF
...
```

contrib/postfix/conf-light/files.d/mailer-conf

Synopsis: Create file.

Create file (2) from the template (7).

[contrib/postfix/conf-light/files.d/mailer-conf]

```
mailerconf:
  path: '/etc/mail/mailer.conf'
  force: true
  owner: 'root'
  group: 'wheel'
  mode: '0644'
  template: 'mailer.conf.j2'
```

contrib/postfix/conf-light/files.d/periodic-conf

Synopsis: Modify file.

Modify file (2) with the lines (7).

[contrib/postfix/conf-light/files.d/periodic-conf]

```
periodic_conf:
  path: '/etc/periodic.conf'
  create: true
  owner: 'root'
  group: 'wheel'
  mode: '0644'
  lines:
    - regexp: '^daily_clean_hoststat_enable(.*)$'
      line: 'daily_clean_hoststat_enable="{{ cl_periodicconf_daily_clean_hoststat_
↪enable }}"'
    - regexp: '^daily_status_mail_rejects_enable(.*)$'
      line: 'daily_status_mail_rejects_enable="{{ cl_periodicconf_daily_status_mail_
↪rejects_enable }}"'
    - regexp: '^daily_status_include_submit_mailq(.*)$'
      line: 'daily_status_include_submit_mailq="{{ cl_periodicconf_daily_status_
↪include_submit_mailq }}"'
    - regexp: '^daily_submit_queuerun(.*)$'
      line: 'daily_submit_queuerun="{{ cl_periodicconf_daily_submit_queuerun }}"'
```

contrib/postfix/conf-light/files.d/postfix-main-cf

Synopsis: Modify file and notify handlers.

Modify file (2) with the lines (9) and notify handlers (7).

[contrib/postfix/conf-light/files.d/postfix-main-cf]

```
postfix_main_cf:
  path: '/usr/local/etc/postfix/main.cf'
  create: true
  owner: 'root'
  group: 'wheel'
  mode: '0644'
  handlers:
    - 'postfix_freebsd reload postfix'
  lines:
    - regexp: '^myhostname\s*=\s*(.*)$'
      line: 'myhostname = {{ cl_myhostname }}'
```

contrib/postfix/conf-light/files.d/rc-conf

Synopsis: Modify file.

Modify file (2) with the lines (7).

[contrib/postfix/conf-light/files.d/rc-conf]

```
rcconf:
  path: '/etc/rc.conf'
  create: true
  owner: 'root'
  group: 'wheel'
  mode: '0644'
  lines:
    - regexp: '^sendmail_enable(.*)$'
      line: 'sendmail_enable="{{ cl_rcconf_sendmail_enable }}"'
    - regexp: '^sendmail_submit_enable(.*)$'
      line: 'sendmail_submit_enable="{{ cl_rcconf_sendmail_submit_enable }}"'
    - regexp: '^sendmail_outbound_enable(.*)$'
      line: 'sendmail_outbound_enable="{{ cl_rcconf_sendmail_outbound_enable }}"'
    - regexp: '^sendmail_msp_queue_enable(.*)$'
      line: 'sendmail_msp_queue_enable="{{ cl_rcconf_sendmail_msp_queue_enable }}"'
    - regexp: '^postfix_enable(.*)$'
      line: 'postfix_enable="{{ cl_rcconf_postfix_enable }}"'
```

4.2 Armbian Simple SMTP

4.2.1 Packages

contrib/ssmtp/conf-light/packages.d/ssmtp

Synopsis: Install Simple SMTP.

Use package (2) to install sSMTP.

[contrib/ssmtp/conf-light/packages.d/ssmtp]

```
ssmtp:
  name: 'ssmtp'
```

See also:

See *packages.yml* how the Linux packages are installed.

4.2.2 Files

contrib/ssmtp/config-light-ssmtp.yml

Synopsis: Custom variables for sSMTP.

Put the host-specific variables (7) into the `host_vars`. Optionally other variables might be put into the `group_vars`.

[contrib/ssmtp/config-light-ssmtp.yml]

```
---

# sSMTP - Simple SMTP
# https://wiki.debian.org/sSMTP

# linux-postinstall FQDN
lp_fqdn: "host99.region9.example.com"

# NEVER USE PLAINTEXT PASSWORD. USE VAULT INSTEAD
smtp_client_password_mail_example_com: "PASSWORD"

# conf-light/files.d/ssmtp-conf
cl_ssmtp_srv: "mail.example.com"
cl_ssmtp_srv_domain: "example.com"

cl_ssmtp_postmaster_address: "postmaster@{{ lp_fqdn }}"
cl_ssmtp_mailhub: "{{ cl_ssmtp_srv }}:587"
cl_ssmtp_rewriteDomain: "{{ cl_ssmtp_srv_domain }}"

cl_ssmtp_UseTLS: "Yes"
cl_ssmtp_UseSTARTTLS: "Yes"

cl_ssmtp_AuthUser: "smtp_client"
cl_ssmtp_AuthPass: "{{ smtp_client_password_mail_example_com }}"
cl_ssmtp_AuthMethod: "LOGIN"

cl_ssmtp_FromLineOverride: "yes"

# conf-light/files.d/revaliases
cl_ssmtp_revaliases:
  - "root:root@{{ cl_ssmtp_srv_domain }}:{{ cl_ssmtp_srv }}:587"
  - "admin:admin@{{ cl_ssmtp_srv_domain }}:{{ cl_ssmtp_srv }}:587"
  - "user1:user1@{{ cl_ssmtp_srv_domain }}:{{ cl_ssmtp_srv }}:587"

# EOF
...
```

contrib/ssmtp/conf-light/files.d/revaliases

Synopsis: Create file.

Create file (2) from the template (7).

[contrib/ssmtp/conf-light/files.d/revaliases]

```
revaliases:
  path: '/etc/ssmtp/revaliases'
  force: true
  owner: 'root'
  group: 'mail'
  mode: 'u=rw,g=r'
  template: 'revaliases.j2'
```

See also:

See template revaliases.j2. See how files are created from template *files-template.yml*.

contrib/ssmtp/conf-light/files.d/ssmtp-conf

Synopsis: Create file.

Create file (2) from the template (7).

[contrib/ssmtp/conf-light/files.d/ssmtp-conf]

```
ssmtp_conf:
  path: '/etc/ssmtp/ssmtp.conf'
  force: true
  owner: 'root'
  group: 'mail'
  mode: 'u=rw,g=r'
  template: 'ssmtp.conf.j2'
```

See also:

See template ssmtp.conf.j2. See how files are created from template *files-template.yml*.

CHAPTER

FIVE

COPYRIGHT

Redistribution and use in source (RST DocUtils) and 'compiled' forms (XML, HTML, PDF, PostScript, RTF and so forth) with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code (RST Docutils) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

Redistributions in compiled form (transformed to other DTDs, converted to PDF, PostScript, RTF and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Important: THIS DOCUMENTATION IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PROVIDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER

SIX

LEGAL NOTICE

All product names, logos, and brands are property of their respective owners.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search