ansible-role-linux-postinstall Documentation

Release 1.14.3

Vladimir Botka

Jul 26, 2020

TABLE OF CONTENTS:

This role and the documentation is work in progess. Feel free to share your feedback and report issues. Contributions are welcome.

CHAPTER

ONE

QUICK START GUIDE

For those users who want to quickly try the role this guide provides an example of how to create users, install packages and configure services.

- Install the role `vbotka.linux_postinstall`

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

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

```
1  shell> cat linux-postinstall.yml
2  - hosts: srv.example.com
3    gather_facts: true
4    connection: ssh
5    remote_user: admin
6    become: yes
7    become_user: root
8    become_method: sudo
9    roles:
10     - vbotka.linux_postinstall
```

- Create `host_vars` with customized variables

```
1  shell> ls -1 host_vars/srv.example.com/lp-*
2  host_vars/srv.example.com/lp-common.yml
3  host_vars/srv.example.com/lp-users.yml
4  host_vars/srv.example.com/lp-passwords.yml
5  host_vars/srv.example.com/lp-packages.yml
6  host_vars/srv.example.com/lp-service.yml
```

- To speedup the execution let's set some variables (2-4) to *false*

```
1  shell> cat host_vars/srv.example.com/lp-common.yml
2  lp_debug: false
3  lp_backup_conf: false
4  lp_flavors_enable: false
```

- Create users

```
1  shell> cat host_vars/srv.example.com/lp-users.yml
2  lp_users:
3    - {name: ansible,
4       shell: /bin/sh}
5    - {name: admin,
6       shell: /bin/bash}
```

(continues on next page)

(continued from previous page)

```
lp_users_groups:
  - {name: admin,
     groups: "adm, dialout"}
```

- Configure passwords

```
shell> cat host_vars/srv.example.com/lp-passwords.yml
lp_passwords: true
lp_passwordstore: true
lp_passwordstore_create: false
lp_passwordstore_overwrite: false
```

- Install packages and enable autoremove

```
shell> cat host_vars/srv.example.com/lp-packages.yml
lp_packages_autoremove: true
lp_packages_install:
  - ansible
  - ansible-lint
  - ansible-tower-cli
```

- Configure services

```
shell> cat host_vars/srv.example.com/lp-service.yml
lp_service_debug: true
lp_service:
  - {name: ssh, state: started, enabled: true}
```

- Test syntax

```
shell> ansible-playbook linux-postinstall.yml --syntax-check
```

- See what variables will be included

```
shell> ansible-playbook linux-postinstall.yml -t lp_debug -e 'lp_debug=True'
```

- Install packages

```
shell> ansible-playbook linux-postinstall.yml -t lp_packages
```

- Dry-run, display differencies and display variables

```
shell> ansible-playbook linux-postinstall.yml -e 'lp_debug=True' --check --diff
```

- Run the playbook

```
shell> ansible-playbook linux-postinstall.yml
```

Warning: The host has not been secured by this playbook and should be used for testing only.

CHAPTER

TWO

USER'S GUIDE

Table of Contents

2.1 Introduction

- Ansible role: linux_postinstall
- Supported systems: Ubuntu
- Requirements: ansible_lib

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.linux_postinstall
```

and install it

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

Install the library of tasks

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

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 a simple playbook that calls this role at a single host srv.example.com (2)

```
1  shell> cat linux-postinstall.yml
2  - hosts: srv.example.com
3    gather_facts: true
4    connection: ssh
5    remote_user: admin
6    become: yes
7    become_user: root
8    become_method: sudo
9    roles:
10     - vbotka.linux_postinstall
```

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)

- See also Understanding Privilege Escalation (6-8)

2.4 Debug

To see additional debug information enable debug output in the configuration

```
lp_debug: true
```

, or set the extra variable in the command

```
shell> ansible-playbook linux_postinstall.yml -e 'lp_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 the user with a 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 linux-postinstall.yml --list-tags

playbook: linux-postinstall.yml

play #1 (srv.example.com): srv.example.com TAGS: []
   TASK TAGS: [always, lp_acpi, lp_acpi_actions, lp_acpi_events,
   lp_aliases, lp_apparmor, lp_apparmor_disable,
   lp_apparmor_enforce, lp_apparmor_packages, lp_apparmor_profiles,
   lp_apparmor_service, lp_authorized_keys, lp_auto_upgrades,
   lp_autofs, lp_bluetooth, lp_bluetooth_conf, lp_bluetooth_debug,
   lp_bluetooth_disable, lp_bluetooth_enable, lp_cron, lp_cron_tab,
   lp_cron_var, lp_debsums, lp_debsums_debug,
   lp_debsums_default_conf, lp_debsums_ignore_conf,
   lp_debsums_packages, lp_debug, lp_fstab, lp_gpg,
   lp_gpg_agent_conf, lp_gpg_conf, lp_gpg_debug,
   lp_gpg_dirmngr_conf, lp_gpg_packages, lp_gpsd, lp_gpsd_bt_rfcom,
   lp_gpsd_config, lp_gpsd_group, lp_gpsd_packages,
   lp_gpsd_service, lp_grub, lp_grub_conf, lp_grub_debug,
   lp_hostname, lp_hosts, lp_hosts_conf, lp_hosts_debug,
   lp_iptables, lp_kvm, lp_kvm_debug, lp_kvm_packages, lp_latex,
   lp_latex_dir, lp_latex_labels, lp_latex_macros,
   lp_latex_packages, lp_libvirt, lp_libvirt_conf,
   lp_libvirt_debug, lp_libvirt_guests_service,
   lp_libvirt_libvirtd_service, lp_libvirt_pkg, lp_lid,
   lp_logrotate, lp_modemmanager, lp_modules, lp_netplan, lp_nfsd,
   lp_nfsd_exports, lp_nfsd_packages, lp_nfsd_service, lp_packages,
   lp_packages_auto, lp_packages_debug, lp_packages_install,
   lp_packages_remove, lp_packages_selections_postinstall,
```

(continues on next page)

(continued from previous page)

```
   lp_packages_selections_preinstall, lp_passwords,
   lp_passwords_debug, lp_passwords_passwordstore, lp_pm,
   lp_postfix, lp_postfix_conf, lp_postfix_debug,
   lp_postfix_service, lp_reboot, lp_repos, lp_repos_debug,
   lp_repos_keys_manage, lp_repos_manage, lp_resolvconf,
   lp_resolvconf_confd_head, lp_resolvconf_debug,
   lp_resolvconf_packages, lp_resolvconf_service, lp_service,
   lp_service_auto, lp_service_debug, lp_service_general, lp_smart,
   lp_smart_conf, lp_smart_packages, lp_smart_service, lp_speechd,
   lp_ssh, lp_ssh_conf, lp_ssh_debug, lp_sshd, lp_sshd_config,
   lp_sshd_debug, lp_sshd_service, lp_sudoers, lp_sudoers_conf,
   lp_sudoers_dconf, lp_sudoers_debug, lp_swap, lp_swap_debug,
   lp_swap_fstab, lp_swap_swapfile, lp_sysctl, lp_sysctl_debug,
   lp_timesyncd, lp_timesyncd_conf, lp_timesyncd_debug,
   lp_timesyncd_service, lp_timezone, lp_timezone_debug,
   lp_timezone_set, lp_tlp, lp_tlp_conf, lp_tlp_debug,
   lp_tlp_packages, lp_tlp_service, lp_udev, lp_udev_debug,
   lp_udev_hciname, lp_udev_hcirun, lp_udev_persistentnet,
   lp_udev_rules, lp_udev_service, lp_ufw, lp_ufw_conf,
   lp_ufw_debug, lp_ufw_packages, lp_ufw_reload, lp_ufw_reset,
   lp_ufw_service, lp_users, lp_users_accounts, lp_users_debug,
   lp_users_groups, lp_vars, lp_virtualbox, lp_virtualbox_debug,
   lp_virtualbox_keys, lp_virtualbox_pkg, lp_virtualbox_repos,
   lp_virtualbox_services, lp_wpa_action_script_dir,
   lp_wpa_action_script_file, lp_wpagui, lp_wpagui_debug,
   lp_wpagui_disableNM, lp_wpagui_mask_NM, lp_wpagui_packages,
   lp_wpasupplicant, lp_wpasupplicant_conf, lp_wpasupplicant_debug,
   lp_wpasupplicant_packages, lp_xen, lp_xen_debug,
   lp_xen_default_grub, lp_xen_global, lp_xen_packages, lp_xorg,
   lp_xorg_conf, lp_xorg_debug, lp_zeitgeist, lp_zfs, lp_zfs_debug,
   lp_zfs_manage, lp_zfs_mountpoints, lp_zfs_packages,
   lp_zfs_services]
```

For example, display the list of the variables and their values with the tag `lp_debug` (when the debug is enabled `lp_debug: true`)

```
shell> ansible-playbook linux_postinstall.yml -t lp_debug
```

See what packages will be installed

```
shell> ansible-playbook linux_postinstall.yml -t lp_packages --check
```

Install packages and exit the play

```
shell> ansible-playbook linux_postinstall.yml -t lp_packages
```

2.6 Tasks

Test single tasks at single remote host *test_01*. Create a playbook

```
shell> cat linux-postinstall.yml
- hosts: test_01
  become: true
```

(continues on next page)

(continued from previous page)

```
  roles:
    - vbotka.linux_postinstall
```

Customize configuration in `host_vars/test_01/lp-*.yml` and check the syntax

```
shell> ansible-playbook linux-postinstall.yml --syntax-check
```

Then dry-run the selected task and see what will be changed. Replace <tag> with valid tag.

```
shell> ansible-playbook linux-postinstall.yml -t <tag> --check --diff
```

When all seems to be ready run the command. Run the command twice and make sure the playbook and the configuration is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t <tag>
```

2.6.1 Netplan

Synopsis

The network configuration abstraction renderer.

See also:

- Annotated Source code *netplan.yml*
- Project website netplan.io

Examples

Example 1: Enable ethernet interface by Netplan

Create a playbook

```
shell> cat linux-postinstall.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.linux_postinstall
```

Create *host_vars/test_01/lp-netplan.yml*

```
shell> cat host_vars/test_01/lp-netplan.yml
lp_netplan: true
lp_netplan_renderer: "networkd"
lp_netplan_conf:
  - file: "10-ethernet.yaml"
    category: "ethernets"
    conf: |
      eth0:
        optional: true
        set-name: eth0
        dhcp4: true
        dhcp6: false
```

(continues on next page)

(continued from previous page)

```
        match:
        macaddress: "<sanitized>"
```

Configure network

```
shell> ansible-playbook linux-postinstall.yml -t lp_netplan

TASK [vbotka.linux_postinstall : netplan: Configure files in /etc/netplan] **
ok: [test_01] => (item={'file': '10-ethernet.yaml', 'category': 'ethernets',
                        'conf': 'eth0:  optional: true  set-name: eth0
                                 dhcp4: true  dhcp6: false  match:
                                 macaddress: "<sanitized>"'})
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_netplan
...
PLAY RECAP *******************************************************************
test_01: ok=6 changed=0 unreachable=0 failed=0 skipped=4 rescued=0 ignored=0
```

Show the configuration of netplan at the remote host

```
test_01> tree /etc/netplan/
/etc/netplan/
├── 01-network-manager-all.yaml
└── 10-ethernet.yaml

test_01> cat /etc/netplan/01-network-manager-all.yaml
# Ansible managed
network:
  version: 2
  renderer: networkd

test_01> cat /etc/netplan/10-ethernet.yaml
# Ansible managed
network:
  version: 2
  renderer: networkd
  ethernets:
    {
    "eth0": {
        "dhcp4": true,
        "dhcp6": false,
        "match": {
            "macaddress": "<sanitized>"
        },
        "optional": true,
        "set-name": "eth0"
    }
  }
```

Example 1: Enable ethernet interface by Netplan

Create a playbook

```
shell> cat linux-postinstall.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.linux_postinstall
```

Create *host_vars/test_01/lp-netplan.yml*

```
shell> cat host_vars/test_01/lp-netplan.yml
lp_netplan: true
lp_netplan_renderer: "networkd"
lp_netplan_conf:
  - file: "10-ethernet.yaml"
    category: "ethernets"
    conf: |
      eth0:
        optional: true
        set-name: eth0
        dhcp4: true
        dhcp6: false
        match:
        macaddress: "<sanitized>"
```

Configure network

```
shell> ansible-playbook linux-postinstall.yml -t lp_netplan

TASK [vbotka.linux_postinstall : netplan: Configure files in /etc/netplan] **
ok: [test_01] => (item={'file': '10-ethernet.yaml', 'category': 'ethernets',
                        'conf': 'eth0:  optional: true  set-name: eth0
                                 dhcp4: true  dhcp6: false  match:
                                 macaddress: "<sanitized>"'})
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_netplan
...
PLAY RECAP ************************************************************
test_01: ok=6 changed=0 unreachable=0 failed=0 skipped=4 rescued=0 ignored=0
```

Show the configuration of netplan at the remote host

```
test_01> tree /etc/netplan/
/etc/netplan/
├── 01-network-manager-all.yaml
└── 10-ethernet.yaml

test_01> cat /etc/netplan/01-network-manager-all.yaml
# Ansible managed
network:
  version: 2
  renderer: networkd

test_01> cat /etc/netplan/10-ethernet.yaml
# Ansible managed
network:
  version: 2
```

(continues on next page)

(continued from previous page)

```
  renderer: networkd
  ethernets:
    {
    "eth0": {
        "dhcp4": true,
        "dhcp6": false,
        "match": {
            "macaddress": "<sanitized>"
        },
        "optional": true,
        "set-name": "eth0"
    }
  }
```

2.6.2 Passwords

Synopsis

Manage user's passwords. At the moment only passwordstore framework is available.

See also:

- Annotated Source code *passwords.yml*
- Project website passwordstore

Passwordstore

Create, or update passwords of selected users at remote hosts by the passwordstore.org `pass` utility. See details of the included task al_pws_user_host.yml

Note:

- Utility pass is required at controller
- GnuPG is required by `pass`

Examples

Example 1: Update passwords or create them if do not exist

Let's start with no passwords stored in passwordstore for users at host *test_01*. The command shows no results

```
shell> pass test_01
```

Create a playbook

```
shell> cat linux-postinstall.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.linux_postinstall
```

Create *host_vars/test_01/lp-users.yml* with two users *user1* and *user2*

```
shell> cat host_vars/test_01/lp-users.yml
lp_users:
  - {name: user1, shell: /bin/sh}
  - {name: user2, shell: /bin/bash}
```

Create users. This step will create these two users and configure their login shell. Other paramteres of the Annsible module user will be ommited because the only required parameter is *name*. It's a good idea to create one account with the login shell */bin/sh* and use it as Ansible remote_user.

```
shell> ansible-playbook linux-postinstall.yml -t lp_users
...
TASK [vbotka.linux_postinstall : users: Manage user accounts] *********
changed: [test_01] => (item=user1)
changed: [test_01] => (item=user2)
```

Create *host_vars/test_01/lp-passwords.yml*

```
shell> cat host_vars/test_01/lp-passwords.yml
lp_passwords: true
lp_passwordstore: true
lp_passwordstore_create: false
lp_passwordstore_overwrite: false
```

Create passwords. This step will use *passwordstore* to create the passwords and configure them. New passwords will be created only if allowed by the configuration of *lp_passwordstore_create*. We set this variable to *True* in this command but keep it *False* in the configuration to keep the passwords once created. The value of *lp_passwordstore_overwrite* is *False*. New passwords will be assigned to the users if no passwords have been assigned to the users before. To change the passwords in the future set both variables *True* on the commandline.

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords \
                                       -e 'lp_passwordstore_create=True'
...

TASK [vbotka.ansible_lib : al_pws_user_host: Retrieve, create or update ...]
ok: [test_01] => (item=user1)
ok: [test_01] => (item=user2)
...
TASK [vbotka.linux_postinstall : users: Manage user accounts] **********
changed: [test_01] => (item=user1)
changed: [test_01] => (item=user2)
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords
...
PLAY RECAP ***********************************************************
test_01: ok=18 changed=0 unreachable=0 failed=0 skipped=20 rescued=0 ...
```

Show the passwords stored in *passwordstore* at the controller

```
shell> pass test_01
test_01
├── user1
└── user2

```

(continues on next page)

(continued from previous page)

```
shell> pass test_01/user1
1rLy0eVpJiTpzj-4
lookup_pass: First generated by ansible on 01/07/2020 16:59:00

shell> pass test_01/user2
u4FLTCkKOHAyJxkg
lookup_pass: First generated by ansible on 01/07/2020 16:59:00
```

Show the *passwordstore* log at the controller

```
shell> cd ~/.password-store
shell> git log

commit 61bb8bcd7c2a359f53c8b3d4bacb8854b4dd9f89 (HEAD -> master)
Author: Vladimir Botka <vbotka@gmail.com>
Date:   Wed Jul 1 16:59:00 2020 +0200

    Add given password for test_01/user2 to store.

commit 97b23a5221e721fb892d739b2817923a6db8614b
Author: Vladimir Botka <vbotka@gmail.com>
Date:   Wed Jul 1 16:59:00 2020 +0200

    Add given password for test_01/user1 to store.
```

Show the created users at the remote host

```
test_01> grep user /etc/passwd
user1:x:1003:1003::/home/user1:/bin/sh
user2:x:1004:1004::/home/user2:/bin/bash
```

Example 2: Update passwords submitted in the variable

Update passwords of users at host *test_01*. Use the same playbook and variables as in Example 1. Update the variable *lp_users* with the new passwords stored in the attribute `userpass`

```
 shell> cat host_vars/test_01/lp-users.yml
 lp_users:
   - {name: user1, shell: /bin/sh, userpass: user1_password}
   - {name: user2, shell: /bin/bash, userpass: user2_password}
```

Update the passwords

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords \
                                    -e 'lp_passwordstore_overwrite=True'
...
TASK [vbotka.linux_postinstall : users: Manage user accounts] **********
changed: [test_01] => (item=user1)
changed: [test_01] => (item=user2)
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords \
                                    -e 'lp_passwordstore_overwrite=True'
```

(continues on next page)

(continued from previous page)

```
...

PLAY RECAP *************************************************************
test_01: ok=18 changed=0 unreachable=0 failed=0 skipped=20 rescued=0 ...
```

Show the passwords stored in *passwordstore* at the controller

```
shell> pass test_01
test_01
├── user1
└── user2

shell> pass test_01/user1
user1_password
lookup_pass: First generated by ansible on 01/07/2020 16:59:00

shell> pass test_01/user2
user2_password
lookup_pass: First generated by ansible on 01/07/2020 16:59:00
```

See the *passwordstore* log at the controller `git log` and test the new passwords at *test_01*.

Example 3: Update passwords by passwordstore

Update passwords of users at host *test_01*. New passwords will be created by the `pass` utility and will be stored in `passwordstore`. Use the same playbook and variables as in Example 1. Remove the attributes `userpass` from the variable *lp_users*. The only required attribute is the `name` of the user.

```
 shell> cat host_vars/test_01/lp-users.yml
 lp_users:
   - name: user1
   - name: user2
```

Update the passwords

```
shell> ansible-playbook linux-postinstall.yml \
       -t lp_passwords \
       -e 'lp_passwordstore_create=True lp_passwordstore_overwrite=True'
...
TASK [vbotka.linux_postinstall : users: Manage user accounts] **********
changed: [test_01] => (item=user1)
changed: [test_01] => (item=user2)
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords
...

PLAY RECAP *************************************************************
test_01: ok=18 changed=0 unreachable=0 failed=0 skipped=20 rescued=0 ...
```

Show the passwords stored in *passwordstore* at the controller

```
shell> pass test_01
test_01
```

(continues on next page)

(continued from previous page)

```
├── user1
└── user2

shell> pass test_01/user1
A,5bH5NtdYQ9FCO:
lookup_pass: First generated by ansible on 01/07/2020 16:59:00

shell> pass test_01/user2
gUp-cn5C.cse6Cx0
lookup_pass: First generated by ansible on 01/07/2020 16:59:00
```

See the *passwordstore* log at the controller `git log` and test the new passwords at *test_01*.

Example 1: Update passwords or create them if do not exist

Let's start with no passwords stored in passwordstore for users at host *test_01*. The command shows no results

```
shell> pass test_01
```

Create a playbook

```
shell> cat linux-postinstall.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.linux_postinstall
```

Create *host_vars/test_01/lp-users.yml* with two users *user1* and *user2*

```
shell> cat host_vars/test_01/lp-users.yml
lp_users:
  - {name: user1, shell: /bin/sh}
  - {name: user2, shell: /bin/bash}
```

Create users. This step will create these two users and configure their login shell. Other paramteres of the Annsible module user will be ommited because the only required parameter is *name*. It's a good idea to create one account with the login shell */bin/sh* and use it as Ansible remote_user.

```
shell> ansible-playbook linux-postinstall.yml -t lp_users
...
TASK [vbotka.linux_postinstall : users: Manage user accounts] *********
changed: [test_01] => (item=user1)
changed: [test_01] => (item=user2)
```

Create *host_vars/test_01/lp-passwords.yml*

```
shell> cat host_vars/test_01/lp-passwords.yml
lp_passwords: true
lp_passwordstore: true
lp_passwordstore_create: false
lp_passwordstore_overwrite: false
```

Create passwords. This step will use *passwordstore* to create the passwords and configure them. New passwords will be created only if allowed by the configuration of *lp_passwordstore_create*. We set this variable to *True* in this command but keep it *False* in the configuration to keep the passwords once created. The value of *lp_passwordstore_overwrite* is *False*. New passwords will be assigned to the users if no passwords have been assigned to the users before. To change the passwords in the future set both variables *True* on the commandline.

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords \
                                       -e 'lp_passwordstore_create=True'
...

TASK [vbotka.ansible_lib : al_pws_user_host: Retrieve, create or update ...]
ok: [test_01] => (item=user1)
ok: [test_01] => (item=user2)
...
TASK [vbotka.linux_postinstall : users: Manage user accounts] **********
changed: [test_01] => (item=user1)
changed: [test_01] => (item=user2)
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords
...
PLAY RECAP *************************************************************
test_01: ok=18 changed=0 unreachable=0 failed=0 skipped=20 rescued=0 ...
```

Show the passwords stored in *passwordstore* at the controller

```
shell> pass test_01
test_01
├── user1
└── user2

shell> pass test_01/user1
1rLy0eVpJiTpzj-4
lookup_pass: First generated by ansible on 01/07/2020 16:59:00

shell> pass test_01/user2
u4FLTCkKOHAyJxkg
lookup_pass: First generated by ansible on 01/07/2020 16:59:00
```

Show the *passwordstore* log at the controller

```
shell> cd ~/.password-store
shell> git log

commit 61bb8bcd7c2a359f53c8b3d4bacb8854b4dd9f89 (HEAD -> master)
Author: Vladimir Botka <vbotka@gmail.com>
Date:   Wed Jul 1 16:59:00 2020 +0200

    Add given password for test_01/user2 to store.

commit 97b23a5221e721fb892d739b2817923a6db8614b
Author: Vladimir Botka <vbotka@gmail.com>
Date:   Wed Jul 1 16:59:00 2020 +0200

    Add given password for test_01/user1 to store.
```

Show the created users at the remote host

```
test_01> grep user /etc/passwd
user1:x:1003:1003::/home/user1:/bin/sh
```

(continues on next page)

(continued from previous page)

```
user2:x:1004:1004::/home/user2:/bin/bash
```

Example 2: Update passwords submitted in the variable

Update passwords of users at host *test_01*. Use the same playbook and variables as in Example 1. Update the variable *lp_users* with the new passwords stored in the attribute `userpass`

```
shell> cat host_vars/test_01/lp-users.yml
lp_users:
  - {name: user1, shell: /bin/sh, userpass: user1_password}
  - {name: user2, shell: /bin/bash, userpass: user2_password}
```

Update the passwords

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords \
                                    -e 'lp_passwordstore_overwrite=True'
...
TASK [vbotka.linux_postinstall : users: Manage user accounts] **********
changed: [test_01] => (item=user1)
changed: [test_01] => (item=user2)
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords \
                                    -e 'lp_passwordstore_overwrite=True'
...

PLAY RECAP **************************************************************
test_01: ok=18 changed=0 unreachable=0 failed=0 skipped=20 rescued=0 ...
```

Show the passwords stored in *passwordstore* at the controller

```
shell> pass test_01
test_01
├── user1
└── user2

shell> pass test_01/user1
user1_password
lookup_pass: First generated by ansible on 01/07/2020 16:59:00

shell> pass test_01/user2
user2_password
lookup_pass: First generated by ansible on 01/07/2020 16:59:00
```

See the *passwordstore* log at the controller `git log` and test the new passwords at *test_01*.

Example 3: Update passwords by passwordstore

Update passwords of users at host *test_01*. New passwords will be created by the `pass` utility and will be stored in `passwordstore`. Use the same playbook and variables as in Example 1. Remove the attributes `userpass` from the variable *lp_users*. The only required attribute is the `name` of the user.

```
shell> cat host_vars/test_01/lp-users.yml
lp_users:
  - name: user1
  - name: user2
```

Update the passwords

```
shell> ansible-playbook linux-postinstall.yml \
       -t lp_passwords \
       -e 'lp_passwordstore_create=True lp_passwordstore_overwrite=True'
...
TASK [vbotka.linux_postinstall : users: Manage user accounts] **********
changed: [test_01] => (item=user1)
changed: [test_01] => (item=user2)
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_passwords
...

PLAY RECAP ************************************************************
test_01: ok=18 changed=0 unreachable=0 failed=0 skipped=20 rescued=0 ...
```

Show the passwords stored in *passwordstore* at the controller

```
shell> pass test_01
test_01
├── user1
└── user2

shell> pass test_01/user1
A,5bH5NtdYQ9FCO:
lookup_pass: First generated by ansible on 01/07/2020 16:59:00

shell> pass test_01/user2
gUp-cn5C.cse6Cx0
lookup_pass: First generated by ansible on 01/07/2020 16:59:00
```

See the *passwordstore* log at the controller `git log` and test the new passwords at *test_01*.

2.6.3 ZFS

Synopsis

Manages ZFS file systems, volumes, clones and snapshots.

See also:

- Annotated Source code *zfs.yml*
- Project website openzfs.org

Examples

Example 1: Mount ZFS filesystems

Create a playbook

```
shell> cat linux-postinstall.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.linux_postinstall
```

Create *host_vars/test_01/lp-zfs.yml*

```
shell> cat host_vars/test_01/lp-zfs.yml
lp_zfs: true
lp_zfs_debug: false

lp_zfs_manage:
  - name: zroot/test1
    state: present
    extra_zfs_properties:
      compression: on
  - name: zroot/images
    state: present
    extra_zfs_properties:
      compression: on
      mountpoint: /var/lib/libvirt/images

lp_zfs_mountpoints:
  - mountpoint: /var/lib/libvirt/images
    owner: root
    group: root
    mode: "0711"
```

Mount the ZFS filesystems

```
shell> ansible-playbook linux-postinstall.yml -t lp_zfs

TASK [vbotka.linux_postinstall : zfs: Manage zfs] ***************************
ok: [test_01] => (item={u'state': u'present', u'extra_zfs_properties':
                       {u'compression': True}, u'name': u'zroot/test1'})
ok: [test_01] => (item={u'state': u'present', u'extra_zfs_properties':
                       {u'mountpoint': u'/var/lib/libvirt/images', u'compression':␣
↪True},
                        u'name': u'zroot/images'})

TASK [vbotka.linux_postinstall : zfs: Set mode and ownership of zfs mountpoints]
changed: [test_01] => (item={u'owner': u'root', u'mountpoint': u'/var/lib/libvirt/
↪images',
                           u'group': u'root', u'mode': u'0711'})
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_zfs
...
PLAY RECAP *****************************************************************
test_01: ok=11 changed=0 unreachable=0 failed=0 skipped=3 rescued=0 ignored=0
```

Show the ZFS mountpoints at the remote host

```
test_01> zfs list
NAME            USED  AVAIL  REFER  MOUNTPOINT
zroot           421M   107G    24K  /zroot
zroot/images    419M   107G   419M  /var/lib/libvirt/images
zroot/test1      24K   107G    24K  /zroot/test1
```

Example 2: Enable ZFS services

Create a playbook

```
shell> cat linux-postinstall.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.linux_postinstall
```

Create *host_vars/test_01/lp-zfs.yml*

```
shell> cat host_vars/test_01/lp-zfs.yml
lp_zfs: true
lp_zfs_debug: false

lp_zfs_services:
  - {name: zfs-mount, enabled: true, state: started}
  - {name: zfs-share, enabled: true, state: started}
  - {name: zfs-zed, enabled: true, state: started}
```

Show status of ZFS services at the remote host

```
test_01> service --status-all | grep zfs
 [ - ]  zfs-import
 [ - ]  zfs-mount
 [ - ]  zfs-share
 [ - ]  zfs-zed
```

Enable ZFS services

```
shell> ansible-playbook linux-postinstall.yml -t lp_zfs

TASK [vbotka.linux_postinstall : zfs: Manage zfs services] ******************
changed: [test_01] => (item={'name': 'zfs-mount', 'enabled': True, 'state': 'started'}
↪)
changed: [test_01] => (item={'name': 'zfs-share', 'enabled': True, 'state': 'started'}
↪)
changed: [test_01] => (item={'name': 'zfs-zed', 'enabled': True, 'state': 'started'})
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_zfs
...
PLAY RECAP *****************************************************************
test_01: ok=6 changed=0 unreachable=0 failed=0 skipped=8 rescued=0 ignored=0
```

Show status of ZFS services at the remote host

```
test_01> service --status-all | grep zfs
 [ - ]  zfs-import
 [ + ]  zfs-mount
 [ + ]  zfs-share
 [ + ]  zfs-zed
```

Example 1: Mount ZFS filesystems

Create a playbook

```
shell> cat linux-postinstall.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.linux_postinstall
```

Create *host_vars/test_01/lp-zfs.yml*

```
shell> cat host_vars/test_01/lp-zfs.yml
lp_zfs: true
lp_zfs_debug: false

lp_zfs_manage:
  - name: zroot/test1
    state: present
    extra_zfs_properties:
      compression: on
  - name: zroot/images
    state: present
    extra_zfs_properties:
      compression: on
      mountpoint: /var/lib/libvirt/images

lp_zfs_mountpoints:
  - mountpoint: /var/lib/libvirt/images
    owner: root
    group: root
    mode: "0711"
```

Mount the ZFS filesystems

```
shell> ansible-playbook linux-postinstall.yml -t lp_zfs

TASK [vbotka.linux_postinstall : zfs: Manage zfs] ***************************
ok: [test_01] => (item={u'state': u'present', u'extra_zfs_properties':
                       {u'compression': True}, u'name': u'zroot/test1'})
ok: [test_01] => (item={u'state': u'present', u'extra_zfs_properties':
                       {u'mountpoint': u'/var/lib/libvirt/images', u'compression':␣
↪True},
                        u'name': u'zroot/images'})

TASK [vbotka.linux_postinstall : zfs: Set mode and ownership of zfs mountpoints]
changed: [test_01] => (item={u'owner': u'root', u'mountpoint': u'/var/lib/libvirt/
↪images',
                           u'group': u'root', u'mode': u'0711'})
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_zfs
...
PLAY RECAP ****************************************************************
test_01: ok=11 changed=0 unreachable=0 failed=0 skipped=3 rescued=0 ignored=0
```

Show the ZFS mountpoints at the remote host

```
test_01> zfs list
NAME            USED  AVAIL  REFER  MOUNTPOINT
zroot           421M   107G    24K  /zroot
zroot/images    419M   107G   419M  /var/lib/libvirt/images
zroot/test1      24K   107G    24K  /zroot/test1
```

Example 2: Enable ZFS services

Create a playbook

```
shell> cat linux-postinstall.yml
- hosts: test_01
  become: true
  roles:
    - vbotka.linux_postinstall
```

Create *host_vars/test_01/lp-zfs.yml*

```
shell> cat host_vars/test_01/lp-zfs.yml
lp_zfs: true
lp_zfs_debug: false

lp_zfs_services:
  - {name: zfs-mount, enabled: true, state: started}
  - {name: zfs-share, enabled: true, state: started}
  - {name: zfs-zed, enabled: true, state: started}
```

Show status of ZFS services at the remote host

```
test_01> service --status-all | grep zfs
 [ - ]  zfs-import
 [ - ]  zfs-mount
 [ - ]  zfs-share
 [ - ]  zfs-zed
```

Enable ZFS services

```
shell> ansible-playbook linux-postinstall.yml -t lp_zfs

TASK [vbotka.linux_postinstall : zfs: Manage zfs services] ******************
changed: [test_01] => (item={'name': 'zfs-mount', 'enabled': True, 'state': 'started'}
↪)
changed: [test_01] => (item={'name': 'zfs-share', 'enabled': True, 'state': 'started'}
↪)
changed: [test_01] => (item={'name': 'zfs-zed', 'enabled': True, 'state': 'started'})
```

The command is idempotent

```
shell> ansible-playbook linux-postinstall.yml -t lp_zfs
...
PLAY RECAP *******************************************************************
test_01: ok=6 changed=0 unreachable=0 failed=0 skipped=8 rescued=0 ignored=0
```

Show status of ZFS services at the remote host

```
test_01> service --status-all | grep zfs
 [ - ]  zfs-import
 [ + ]  zfs-mount
 [ + ]  zfs-share
 [ + ]  zfs-zed
```

2.7 Variables

See also:

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

2.7.1 Default variables

The common variables for all distributions are in the file `defaults/main.yml`. These variables can be customized in the file `vars/main.yml`. The file `vars/main.yml` will be preserved by the update of the role.

Warning:

- Don't make any changes to the file *defaults/main.yml*. The changes will be overwritten by the update of the role. Customize the default values in the file *vars/main.yml*.
- Default value of *lp_passwords_debug_classified* and *lp_wpasupplicant_debug_classified* is *False*. Passwords will be displayed if these variables are enabled.

See also:

- The examples of the customization `vars/main.yml.sample`

[defaults/main.yml]

```
---
# linux_postinstall defaults

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Put distro and flavor specific vars and overrides to the vars
# directory.

lp_vars_distro: firstfound
lp_vars_distro_firstfound_skip: true  # issue #43833
lp_vars_flavors: firstfound

lp_hostname: ""
lp_fqdn: ""

lp_debug: false
```

(continues on next page)

(continued from previous page)

```
lp_backup_conf: false

lp_install_retries: 5
lp_install_delay: 2

lp_repos_debug: false
lp_repos: []
lp_repos_keys: []

lp_packages_auto: false
lp_packages_debug: false
lp_packages_install: []
lp_packages_remove: []
lp_packages_selections_preinstall: []
lp_packages_selections_postinstall: []
lp_packages_autoremove: false

lp_package_state: present               # apt and yum support "latest"
lp_package_state_remove: absent         # see "remove" vs. "purge"
lp_package_install_dryrun: false

lp_modules_debug: false
lp_modules: []
lp_modules_blacklist: []
lp_modules_blacklist_path: /etc/modprobe.d
lp_modules_options: []
lp_modules_options_path: /etc/modprobe.d

lp_sysctl_debug: false
lp_sysctl_vars: []

lp_udev: true
lp_udev_debug: false
lp_udev_enable: true
lp_udev_rules_dir: /etc/udev/rules.d
lp_udev_rules_template: udev-rules.j2
lp_udev_rules: {}
lp_udev_persistent_net_template: persistent-net.rules.j2
lp_udev_persistent_net_rules_file: 70-persistent-net.rules
lp_udev_persistent_net_rules: []
lp_udev_hci_name_rules_file: 71-hci-name.rules
lp_udev_hci_name_rules: []
lp_udev_hci_run_rules_file: 72-hci-run.rules
lp_udev_hci_run_rules: []

lp_wpagui: false
lp_wpagui_debug: false
lp_wpagui_nm_override: manual

lp_iptables: false
lp_iptables_type: default
lp_iptables_wan_if: eth0
lp_iptables_lan_if: eth1
lp_iptables_lan: 10.1.0.0/24
lp_iptables_INPUT_if: []
lp_iptables_INPUT_net: []

```

(continues on next page)

(continued from previous page)

```
lp_users_debug: false
lp_users: []
lp_users_groups: []

lp_passwords: false
lp_passwords_debug: false
lp_passwords_debug_classified: false
lp_passwords_fail_gracefully: false
lp_password_update_password: always
lp_passwordstore: false
lp_passwordstore_debug: false
lp_passwordstore_install: true
lp_passwordstore_backup: false
lp_passwordstore_create: false
lp_passwordstore_length: 16
lp_passwordstore_nosymbols: false
lp_passwordstore_overwrite: false
lp_passwordstore_passwordstore: ~/.password-store
lp_passwordstore_returnall: false
lp_passwordstore_subkey: password
lp_passwordstore_idempotent_password_hash: true

lp_aliases: false
lp_aliases_config: []

lp_authorized_key: []

lp_hosts_debug: false
lp_hosts: []

lp_grub: true
lp_grub_debug: false
lp_grub_default: []

lp_kvm: false
lp_kvm_debug: false

lp_xen: false
lp_xen_debug: false
lp_xen_dom0_mem: 512M
lp_xen_dom0_mem_max: 512M
lp_xen_XEN_OVERRIDE_GRUB_DEFAULT: 0
lp_xen_default_grub_conf:
  - key: GRUB_CMDLINE_XEN_DEFAULT
    value: "\"dom0_mem={{ lp_xen_dom0_mem }}, max:{{ lp_xen_dom0_mem_max }}\""
  - key: XEN_OVERRIDE_GRUB_DEFAULT
    value: "{{ lp_xen_XEN_OVERRIDE_GRUB_DEFAULT }}"
lp_xen_global: []

lp_latex: false
lp_latex_download_timeout: 20
lp_latex_macros: []
lp_latex_get_url_ignore_errors: false

lp_auto_upgrades: false
lp_auto_upgrades_enable: false

```

(continues on next page)

(continued from previous page)

```
lp_auto_upgrades_Update_Package_Lists: 0
lp_auto_upgrades_Unattended_Upgrade: 0

lp_pm: false
lp_pm_sleepd: {}

lp_ssh: false
lp_ssh_debug: false
lp_ssh_config: []

lp_sshd: false
lp_sshd_debug: false
lp_sshd_enable: false
lp_sshd_config: []

lp_bluetooth: false
lp_bluetooth_debug: false
lp_bluetooth_enable: false
lp_bluetooth_main_conf: []

lp_xorg_debug: false
lp_xorg_conf_dir: /usr/share/X11/xorg.conf.d
lp_xorg_conf: []

lp_cron_tab: []
lp_cron_var: []

lp_modemmanager: true
lp_modemmanager_enable: true
lp_modemmanager_override: ""

lp_gpsd: false
lp_gpsd_enable: false
lp_gpsd_START_DAEMON: "true"
lp_gpsd_USBAUTO: "true"
lp_gpsd_DEVICES: /dev/rfcomm0
lp_gpsd_GPSD_OPTIONS: -b
lp_gpsd_bt_rfcomm: []

lp_postfix: true
lp_postfix_debug: false
lp_postfix_enable: true
lp_postfix_main_conf: []

lp_smart: false
lp_smart_debug: false
lp_smart_enable: false
lp_smart_packages:
  - smartmontools
lp_smart_service: smartmontools
lp_smart_devicescan: false
lp_smart_conf_file: /etc/smartd.conf
lp_smart_conf_owner: root
lp_smart_conf_group: root
lp_smart_conf_mode: "0644"
lp_smart_devices: []

```

(continues on next page)

(continued from previous page)

```
lp_virtualbox: false
lp_virtualbox_debug: false
lp_virtualbox_enable: false
lp_virtualbox_ignore_errors: false
lp_virtualbox_version: 5.2
lp_virtualbox_keys:
  - https://www.virtualbox.org/download/oracle_vbox_2016.asc
  - https://www.virtualbox.org/download/oracle_vbox.asc
lp_virtualbox_packages:
  - "virtualbox-{{ lp_virtualbox_version }}"

lp_zeitgeist: true

lp_lid: false
lp_lid_logind_conf: /etc/systemd/logind.conf
lp_lid_logind_conf_vars: []
lp_lid_upower_conf: /etc/UPower/UPower.conf
lp_lid_upower_conf_vars: []

lp_acpi: false
lp_acpi_dir: /etc/acpi
lp_acpi_owner: root
lp_acpi_group: root
lp_acpi_event_mode: "0644"
lp_acpi_action_mode: "0755"
lp_acpi_events: {}
lp_acpi_actions: {}

lp_speechd: true
lp_speechd_debug: false
lp_speechd_enable: true

lp_sudoers_debug: false
lp_sudoers_owner: root
lp_sudoers_group: root
lp_sudoers_mode: "0440"
lp_sudoers_conf:
  - {line: "#includedir /etc/sudoers.d", state: "present"}
lp_sudoers_01: []

lp_nfsd: false
lp_nfsd_enable: false
lp_nfsd_exports: []

lp_netplan: false
lp_netplan_root: /etc/netplan
lp_netplan_owner: root
lp_netplan_group: root
lp_netplan_version: 2
lp_netplan_renderer: NetworkManager
lp_netplan_conf: []

lp_apparmor: true
lp_apparmor_disable: []
lp_apparmor_complain: []
lp_apparmor_enforce: []

```

(continues on next page)

(continued from previous page)

```
lp_swap: false
lp_swap_debug: false
lp_swap_enable: false

lp_timezone: false
lp_timezone_debug: false
lp_timezone_zoneinfo: UTC

lp_timesyncd: true
lp_timesyncd_debug: false
lp_timesyncd_enable: true
lp_timesyncd_NTP: ""
lp_timesyncd_FallbackNTP: ntp.ubuntu.com
lp_timesyncd_RootDistanceMaxSec: 5
lp_timesyncd_PollIntervalMinSec: 32
lp_timesyncd_PollIntervalMaxSec: 2048

lp_gpg: false
lp_gpg_debug: false
lp_gpg_install: true
lp_gpg_packages_extra: []
lp_gpg_conf_default: []
lp_gpg_conf: []
lp_gpg_agent_conf_default: []
lp_gpg_agent_conf: []
lp_gpg_dirmngr_conf_default: []
lp_gpg_dirmngr_conf: []

lp_wpasupplicant: true
lp_wpasupplicant_debug: false
lp_wpasupplicant_debug_classified: false
lp_wpasupplicant_conf_only: false
lp_wpasupplicant_conf_owner: root
lp_wpasupplicant_conf_group: root
lp_wpasupplicant_conf_mode: "0600"
lp_wpasupplicant_conf_dir: /etc/wpa_supplicant
lp_wpasupplicant_conf_file: wpa_supplicant.conf
lp_wpasupplicant_conf_global:
  - {key: ctrl_interface, value: "{{ lp_wpasupplicant_conf_ctrl_interface }}"}
lp_wpasupplicant_conf: []
lp_wpa_action_script: false
lp_wpa_action_script_dir: /root/bin
lp_wpa_action_script_file: "{{ lp_wpa_action_script_dir }}/wpa_action.sh"
lp_wpa_action_script_owner: root
lp_wpa_action_script_group: root
lp_wpa_action_script_mode: "0770"
lp_wpa_action_script_dhclient: "{{ lp_dhclient }}"
lp_wpa_action_script_pidfile: /var/run/dhclient.$ifname.pid
lp_wpa_action_script_options_connect: "-4 -nw -pf $pidfile -v"
lp_wpa_action_script_options_disconnect: "-4 -r -pf $pidfile -v"
lp_wpa_action_script_logfile: "/tmp/wpa_action.$ifname"

lp_logrotate_conf_file: /etc/logrotate.conf
lp_logrotate_conf_dir: /etc/logrotate.d
lp_logrotate_conf_lines:
  - {line: "include /etc/logrotate.d", state: "present"}
lp_logrotate_conf_blocks: []
```

(continues on next page)

(continued from previous page)

```yaml
lp_logrotate_confd: []

lp_tlp: false
lp_tlp_debug: false
lp_tlp_enable: false
lp_tlp_thinkpad: false
lp_tlp_config: []

lp_autofs: false
lp_autofs_enable: false
lp_autofs_conf_file: /etc/autofs.conf
lp_autofs_conf: []
lp_autofs_master_conf_file: /etc/auto.master
lp_autofs_master_conf: []
lp_autofs_misc_conf_file: /etc/auto.misc
lp_autofs_misc_conf: []

lp_libvirt: false
lp_libvirt_debug: false
lp_libvirt_guests_enable: false
lp_libvirt_libvirtd_enable: false
lp_libvirt_conf: {}

lp_zfs: false
lp_zfs_install: true
lp_zfs_debug: false
lp_zfs_manage: []
lp_zfs_mountpoints: []

lp_service_debug: false
lp_service: []
lp_service_enable:
  - udev
  - auto_upgrades
  - sshd
  - bluetooth
  - gpsd
  - postfix
  - smart
  - speechd
  - timesyncd
  - autofs
  - libvirt_libvirtd
  - libvirt_guests

lp_ufw: true
lp_ufw_enable: true
lp_ufw_debug: false
lp_ufw_reset: false
lp_ufw_reload: false
lp_ufw_service: ufw
lp_ufw_conf:
  - {state: enabled, policy: allow}
  - {logging: "on"}

lp_debsums: false
lp_debsums_debug: false
```

(continues on next page)

(continued from previous page)

```
lp_debsums_default_file: /etc/default/debsums
lp_debsums_default_conf:
  - {key: CRON_CHECK, value: never}
lp_debsums_ignore_file: /etc/debsums-ignore
lp_debsums_ignore_conf: []

lp_fstab_entries: []

lp_resolvconf: false
lp_resolvconf_enable: false
lp_resolvconf_debug: false
lp_resolvconf_confd_head: []

lp_reboot: false
lp_reboot_debug: false
lp_reboot_force: false
lp_reboot_required_ignore: true
lp_reboot_required_file: /var/run/reboot-required
lp_reboot_required_command: /sbin/needs-restarting -r
lp_reboot_command: "sleep 5 && shutdown -r now"
lp_reboot_wait_connect_timeout: 20
lp_reboot_wait_sleep: 5
lp_reboot_wait_delay: 5
lp_reboot_wait_timeout: 300

# Include default vars for various flavors. For example put vars into
# one of the files below. First found will be included.
#
#     vars/flavors/armbian-<VERSION>-<BOARD>.yml
#     vars/flavors/armbian-<VERSION>.yml
#     vars/flavors/armbian.yml
#     vars/defaults.yml
#
# 1) File with service tasks task/sub/vars-flavors-<flavor>.yml is
#    needed when new flavor is added to lp_flavors. See
#    tasks/sub/vars-flavors-common.yml
# 2) For precedence of vars see tasks/vars.yml

lp_flavors_enable: true
lp_flavors_dir: "{{ inventory_dir ~ '/flavors' }}"
lp_flavors_dir_owner: admin
lp_flavors_dir_group: adm
lp_flavors_dir_mode: "0775"
lp_flavors:
  lsb:
    release_file: /etc/lsb-release
    file_labels: [DISTRIB_ID, DISTRIB_CODENAME]
  os:
    release_file: /etc/os-release
    file_labels: [ID, UBUNTU_CODENAME]
  armbian:
    release_file: /etc/armbian-release
    file_labels: [VERSION, BOARD]

# userland paths
lp_dhclient: /sbin/dhclient

```

(continues on next page)

(continued from previous page)

```
# TODO:
# * lp_virtualbox_services lp_virtualbox_enable
# * lp_tlp_services lp_tlp_enable
# * lp_nfsd_services lp_nfsd_enable

# EOF
...
```

2.7.2 Default OS specific variables

The files in the directories `vars/defaults` and `vars/defaults.incr` keep OS specific variables. The difference between them is the method how the files are included.

- **firstfound**: A file in the directory `vars/defaults` in the order `ansible_distribution_release`, `ansible_distribution`, and `ansible_os_family` is included with the *lookup plugin* `first_found`.
- **incremental**: All files in the directory `vars/defaults.incr` in the order `ansible_os_family`, `ansible_distribution`, and `ansible_distribution_release` are included in the *loop*. The variables in the files overwrite each other.

In addition, also files `defaults.yml` and/or `default.yml` can be included.

The method is determined by the variable `lp_vars_distro` *(default: firstfound)*

```
lp_vars_distro: firstfound
```

Warning:

- Don't change these file. The changes will be overwritten by the update of the role. Customize the default OS values in the files placed in the directory `vars/`

See also:

- Annotated Source code *vars.yml*
- Annotated Source code *vars-firstfound.yml*
- Annotated Source code *vars-incremental.yml*

2.7.3 Custom OS specific variables

Os specific variables can be customized in files placed in the directory `vars/`. These files will be preserved by the update.

See also:

- The examples of the customization `vars/defaults.yml.sample`

2.7.4 Flavors specific variables (WIP)

The files in the directories `vars/flavors` and `vars/flavors.incr` keep flavors specific variables. The difference between them is the method how the files are included.

- **firstfound**: A file in the directory `vars/flavors` in the order <TBD> is included with the *lookup plugin* `first_found`.
- **incremental**: All files in the directory `vars/defaults.incr` in the order <TBD> are included in the *loop*. The variables in the files overwrite each other.

In addition, also files `defaults.yml` and/or `default.yml` can be included.

The method is determined by the variable `lp_vars_flavor` *(default: firstfound)*

```
lp_vars_flavor: firstfound
```

See also:

- Annotated Source code *sub/vars-flavors.yml*
- Annotated Source code *sub/vars-flavors-common.yml*

2.8 Best practice

2.8.1 Recommended configuration after the installation of OS

Test syntax

```
shell> ansible-playbook linux-postinstall.yml --syntax-check
```

See what variables will be included

```
shell> ansible-playbook linux-postinstall.yml -t lp_debug -e 'lp_debug=True'
```

Dry-run, display differences and display variables

```
shell> ansible-playbook linux-postinstall.yml -e 'lp_debug=True' --check --diff
```

Configure hostname, users, sudoers, network and reboot

```
shell> ansible-playbook linux-postinstall.yml -t lp_hostname
shell> ansible-playbook linux-postinstall.yml -t lp_users
shell> ansible-playbook linux-postinstall.yml -t lp_sudoers
shell> ansible-playbook linux-postinstall.yml -t lp_udev
shell> ansible-playbook linux-postinstall.yml -t lp_netplan
shell> ansible-playbook linux-postinstall.yml -t lp_wpasupplicant
shell> ansible-playbook linux-postinstall.yml -t lp_reboot \
       -e "lp_reboot=true lp_reboot_force=true"
```

Configure firewall

```
shell> ansible-playbook linux-postinstall.yml -t lp_iptables
```

Test the installation of packages

```
shell> ansible-playbook linux-postinstall.yml -t cl_packages \
       -e "lp_package_install_dryrun=True"
```

Install packages

```
shell> ansible-playbook linux-postinstall.yml -t cl_packages
```

Run the playbook

```
shell> ansible-playbook linux-postinstall.yml
```

Test the idem-potency. The role and the configuration data shall be idempotent. Once the installation and configuration have passed there should be no changes reported by *ansible-playbook* when running the playbook repeatedly. Disable debug, and install to speedup the playbook and run the playbook again.

```
 shell> ansible-playbook linux-postinstall.yml
```

2.8.2 Flavors

<TBD>

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]

```yaml
---
# tasks linux-postinstall

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

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

- import_tasks: swap.yml
  when: ((ansible_os_family == "RedHat") or
         (ansible_os_family == "Debian")) and lp_swap|bool
  tags: lp_swap

- import_tasks: modules.yml
  when: (ansible_os_family == "RedHat") or
        (ansible_os_family == "Debian")
  tags: lp_modules

- import_tasks: udev.yml
  when: ((ansible_os_family == "RedHat") or
         (ansible_os_family == "Debian")) and lp_udev|bool
  tags: lp_udev

- import_tasks: fstab.yml
  when: ((ansible_os_family == "RedHat") or
         (ansible_os_family == "Debian"))
  tags: lp_fstab

- import_tasks: netplan.yml
  when: (ansible_os_family == "Debian") and lp_netplan|bool
  tags: lp_netplan

- import_tasks: timezone.yml
  when: ((ansible_os_family == "RedHat") or
         (ansible_os_family == "Debian")) and lp_timezone|bool
  tags: lp_timezone
```

(continues on next page)

(continued from previous page)

```

- import_tasks: timesyncd.yml
  when: (ansible_os_family == "Debian") and lp_timesyncd|bool
  tags: lp_timesyncd

- import_tasks: repos.yml
  when: ansible_os_family == "Debian"
  tags: lp_repos

- import_tasks: packages.yml
  when: (ansible_os_family == "RedHat") or
        (ansible_os_family == "Debian")
  tags: lp_packages

- import_tasks: auto_upgrades.yml
  when: (ansible_os_family == "Debian") and lp_auto_upgrades|bool
  tags: lp_auto_upgrades

- import_tasks: sysctl.yml
  when: (ansible_os_family == "RedHat") or
        (ansible_os_family == "Debian")
  tags: lp_sysctl

- import_tasks: zfs.yml
  when: (ansible_os_family == "Debian") and lp_zfs|bool
  tags: lp_zfs

- import_tasks: hostname.yml
  when: (ansible_os_family == "RedHat") or
        (ansible_os_family == "Debian")
  tags: lp_hostname

- import_tasks: hosts.yml
  when: (ansible_os_family == "RedHat") or
        (ansible_os_family == "Debian")
  tags: lp_hosts

- import_tasks: iptables.yml
  when: (ansible_os_family == "Debian") and lp_iptables|bool
  tags: lp_iptables

- import_tasks: grub.yml
  when: (ansible_os_family == "Debian") and lp_grub|bool
  tags: lp_grub
  # https://unix.stackexchange.com/questions/152222/
  # equivalent-of-update-grub-for-rhel-fedora-centos-systems

- import_tasks: users.yml
  when: (ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" )
  tags: lp_users

- import_tasks: gpg.yml
  when: (ansible_os_family == "Debian") and lp_gpg|bool
  tags: lp_gpg

- import_tasks: passwords.yml
```

(continues on next page)

(continued from previous page)

```yaml
  when: (ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" ) and lp_passwords|bool
  tags: lp_passwords

- import_tasks: sudoers.yml
  when: (ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" )
  tags: lp_sudoers

- import_tasks: authorized_keys.yml
  when: (ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" )
  tags: lp_authorized_keys

- import_tasks: aliases.yml
  when: ((ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" )) and lp_aliases|bool
  tags: lp_aliases

- import_tasks: pm-utils.yml
  when: (ansible_os_family == "Debian") and lp_pm|bool
  tags: lp_pm

- import_tasks: ssh.yml
  when: ((ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" )) and lp_ssh|bool
  tags: lp_ssh

- import_tasks: sshd.yml
  when: ((ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" )) and lp_sshd|bool
  tags: lp_sshd

- import_tasks: bluetooth.yml
  when: (ansible_os_family == "Debian") and lp_bluetooth|bool
  tags: lp_bluetooth

- import_tasks: xorg.yml
  when: ansible_os_family == "Debian"
  tags: lp_xorg

- import_tasks: cron.yml
  when: (ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" )
  tags: lp_cron

- import_tasks: modemmanager.yml
  when: (ansible_os_family == "Debian") and lp_modemmanager|bool
  tags: lp_modemmanager

- import_tasks: gpsd.yml
  when: (ansible_os_family == "Debian") and lp_gpsd|bool
  tags: lp_gpsd

- import_tasks: postfix.yml
  when: ((ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" )) and lp_postfix|bool
```

(continues on next page)

(continued from previous page)

```
  tags: lp_postfix

- import_tasks: smart.yml
  when: (ansible_os_family == "Debian") and lp_smart|bool
  tags: lp_smart

- import_tasks: apparmor.yml
  when: (ansible_os_family == "Debian") and lp_apparmor|bool
  tags: lp_apparmor

- meta: flush_handlers

- import_tasks: zeitgeist.yml
  when: ansible_os_family == "Debian"
  tags: lp_zeitgeist

- import_tasks: lid.yml
  when: (ansible_os_family == "Debian") and lp_lid|bool
  tags: lp_lid

- import_tasks: acpi.yml
  when: (ansible_os_family == "Debian") and lp_acpi|bool
  tags: lp_acpi

- import_tasks: speechd.yml
  when: (ansible_os_family == "Debian") and lp_speechd|bool
  tags: lp_speechd

- import_tasks: nfsd.yml
  when: (ansible_os_family == "Debian") and lp_nfsd|bool
  tags: lp_nfsd

- meta: flush_handlers

- import_tasks: latex.yml
  when: (ansible_os_family == "Debian") and lp_latex|bool
  tags: lp_latex

- import_tasks: kvm.yml
  when: (ansible_os_family == "Debian") and lp_kvm|bool
  tags: lp_kvm

- import_tasks: xen.yml
  when: (ansible_os_family == "Debian") and lp_xen|bool
  tags: lp_xen

- import_tasks: virtualbox.yml
  when: (ansible_os_family == "Debian") and lp_virtualbox|bool
  tags: lp_virtualbox

- import_tasks: wpagui.yml
  when: (ansible_os_family == "Debian") and lp_wpagui|bool
  tags: lp_wpagui

- import_tasks: wpasupplicant.yml
  when: ((ansible_os_family == "RedHat" ) or
         (ansible_os_family == "Debian" )) and lp_wpasupplicant|bool
```

(continues on next page)

(continued from previous page)

```
  tags: lp_wpasupplicant

- import_tasks: logrotate.yml
  when: (ansible_os_family == "RedHat" ) or
        (ansible_os_family == "Debian" )
  tags: lp_logrotate

- import_tasks: tlp.yml
  when: (ansible_os_family == "Debian") and lp_tlp|bool
  tags: lp_tlp

- import_tasks: autofs.yml
  when: (ansible_os_family == "Debian") and lp_autofs|bool
  tags: lp_autofs

- import_tasks: libvirt.yml
  when: (ansible_os_family == "Debian") and lp_libvirt|bool
  tags: lp_libvirt

- import_tasks: ufw.yml
  when: (ansible_os_family == "Debian") and lp_ufw|bool
  tags: lp_ufw

- import_tasks: debsums.yml
  when: (ansible_os_family == "Debian") and lp_debsums|bool
  tags: lp_debsums

- meta: flush_handlers

- import_tasks: service.yml
  tags: lp_service

- import_tasks: resolvconf.yml
  when: (ansible_os_family == "Debian") and lp_resolvconf|bool
  tags: lp_resolvconf

- import_tasks: reboot.yml
  when: ((ansible_os_family == "RedHat" ) or
         (ansible_os_family == "Debian")) and lp_reboot|bool
  tags: lp_reboot

# EOF
...
```

3.1.2 acpi.yml

Synopsis: Configure acpi.

Description of the task.

[acpi.yml]

```
---
# linux-postinstall acpi

- name: "acpi: Configure {{ lp_acpi_dir }}/events"
```

(continues on next page)

(continued from previous page)

```
  template:
    src: "{{ item.value.template }}"
    dest: "{{ lp_acpi_dir }}/events/{{ item.value.file }}"
    owner: "{{ lp_acpi_owner }}"
    group: "{{ lp_acpi_group }}"
    mode: "{{ lp_acpi_event_mode }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_acpi_events|dict2items }}"
  tags: lp_acpi_events

- name: "acpi: Create actions in {{ lp_acpi_dir }}"
  template:
    src: "{{ item.value.template }}"
    dest: "{{ lp_acpi_dir }}/{{ item.value.file }}"
    owner: "{{ lp_acpi_owner }}"
    group: "{{ lp_acpi_group }}"
    mode: "{{ lp_acpi_action_mode }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_acpi_actions|dict2items }}"
  tags: lp_acpi_actions

# EOF
...
```

3.1.3 aliases.yml

Synopsis: Configure aliases.

Description of the task.

[aliases.yml]

```
---
# linux-postinstall aliases

- name: "aliases: Configure /etc/aliases"
  template:
    src: aliases.j2
    dest: /etc/aliases
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
  notify: newaliases

# EOF
...
```

3.1.4 apparmor.yml

Synopsis: Configure apparmor.

Description of the task.

[apparmor.yml]

```yaml
---
# linux-postinstall apparmor

- name: "apparmor: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_apparmor_packages }}"
  tags: lp_apparmor_packages

- name: "apparmor: Create list of profiles"
  block:
    - name: "apparmor: List profiles"
      shell: "aa-status --json | jq .profiles | jq to_entries"
      register: result
      changed_when: false
    - name: "apparmor: Create list of enforced profiles"
      set_fact:
        lp_apparmor_profiles_enforce: "{{ lp_apparmor_profiles_enforce|default([]) +␣
↪[ item.key ] }}"
      loop: "{{ result.stdout|default([]) }}"
      when: item.value == 'enforce'
    - name: "apparmor: Create list of complained profiles"
      set_fact:
        lp_apparmor_profiles_complain: "{{ lp_apparmor_profiles_complain|default([])␣
↪+ [ item.key ] }}"
      loop: "{{ result.stdout|default([]) }}"
      when: item.value == 'complain'
    - name: "apparmor: Debug: List enforced profiles"
      debug:
        var: lp_apparmor_profiles_enforce
      when: lp_debug
    - name: "apparmor: Debug: List complained profiles"
      debug:
        var: lp_apparmor_profiles_complain
      when: lp_debug
  tags: lp_apparmor_profiles

- name: "apparmor: Disable profiles"
  command: aa-disable "{{ item }}"
  loop: "{{ lp_apparmor_disable }}"
  when: item in lp_apparmor_profiles_enforce|default([]) or
        item in lp_apparmor_profiles_complain|default([])
  tags: lp_apparmor_disable

- name: "apparmor: Enforce profiles"
  command: aa-enforce "{{ item }}"
  loop: "{{ lp_apparmor_enforce }}"
  when: item not in lp_apparmor_profiles_enforce|default([])
  tags: lp_apparmor_enforce

- name: "apparmor: Complain profiles"
  command: aa-complain "{{ item }}"
  loop: "{{ lp_apparmor_complain }}"
  when: item not in lp_apparmor_profiles_complain|default([])
  tags: lp_apparmor_enforce

- name: "apparmor: Start and enable apparmor"
  service:
```

(continues on next page)

(continued from previous page)

```
    name: apparmor
    state: started
    enabled: true
  when: lp_apparmor|bool
  tags: lp_apparmor_service

- name: "apparmor: Stop and disable apparmor"
  service:
    name: apparmor
    state: stopped
    enabled: false
  when: not lp_apparmor|bool
  tags: lp_apparmor_service

# EOF
...
```

3.1.5 authorized_keys.yml

Synopsis: Configure authorized_keys.

Description of the task.

[authorized_keys.yml]

```
---
# linux-postinstall authorized_keys

- name: "authorized_key: Configure authorized_keys"
  authorized_key:
    user: "{{ item.user }}"
    key: "{{ item.key }}"
    manage_dir: true
  loop: "{{ lp_authorized_keys }}"

# EOF
...
```

3.1.6 autofs.yml

Synopsis: Configure autofs.

Description of the task.

[autofs.yml]

```
---
# linux-postinstall autofs

- name: "autofs: Debug"
  debug:
    msg: "lp_autofs_enable [ {{ lp_autofs_enable }} ]"
  when: lp_debug|bool

- name: "autofs: Configure {{ lp_autofs_conf_file }}"
```

(continues on next page)

(continued from previous page)

```
  lineinfile:
    dest: "{{ lp_autofs_conf_file }}"
    regexp: "^{{ item.key }}\\s*=\\s*(.*)$"
    line: "{{ item.key }} = {{ item.value }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_autofs_conf }}"
  notify: reload autofs

- name: "autofs: Configure {{ lp_autofs_master_conf_file }}"
  lineinfile:
    dest: "{{ lp_autofs_master_conf_file }}"
    regexp: "^{{ item.key }}\\s*(.*)$"
    line: "{{ item.key }} {{ item.value }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_autofs_master_conf }}"
  notify: reload autofs

- name: "autofs: Configure {{ lp_autofs_misc_conf_file }}"
  lineinfile:
    dest: "{{ lp_autofs_misc_conf_file }}"
    regexp: "^{{ item.key }}\\s*(.*)$"
    line: "{{ item.key }} {{ item.value }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_autofs_misc_conf }}"
  notify: reload autofs

- name: "autofs: Enable and start autofs"
  systemd:
    name: "{{ lp_autofs_service }}"
    enabled: true
    state: started
  when: lp_autofs_enable|bool

- name: "autofs: Stop and disable autofs"
  systemd:
    name: "{{ lp_autofs_service }}"
    enabled: false
    state: stopped
  when: not lp_autofs_enable|bool

# EOF
...
```

3.1.7 auto_upgrades.yml

Synopsis: Configure auto_upgrades.

Description of the task.

[auto_upgrades.yml]

```
---
# linux-postinstall auto_upgrades

- name: "auto_upgrades: Configure /etc/apt/apt.conf.d/20auto-upgrades"
  template:
```

(continues on next page)

(continued from previous page)

```yaml
    src: auto-upgrades.j2
    dest: /etc/apt/apt.conf.d/20auto-upgrades
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"

- name: "auto_upgrades: Disable and stop unattended-upgrades"
  systemd:
    name: "{{ lp_auto_upgrades_service }}"
    state: stopped
    enabled: false
  when: not lp_auto_upgrades_enable|bool

- name: "auto_upgrades: Enable and start unattended-upgrades"
  systemd:
    name: "{{ lp_auto_upgrades_service }}"
    state: started
    enabled: true
  when: lp_auto_upgrades_enable|bool

# EOF
...
```

3.1.8 bluetooth.yml

Synopsis: Configure bluetooth.

Description of the task.

[bluetooth.yml]

```yaml
---
# linux-postinstall bluetooth

- name: "bluetooth: Debug"
  debug:
    msg: "lp_bluetooth_enable [{{ lp_bluetooth_enable }}]"
  when: lp_bluetooth_debug|bool
  tags: lp_bluetooth_debug

- name: "bluetooth: Configure /etc/bluetooth/main.conf"
  lineinfile:
    dest: /etc/bluetooth/main.conf
    regexp: "^{{ item.key }}\\s*=(.*)$"
    insertbefore: "^{{ '#' }}{{ item.key }}\\s*=(.*)$"
    line: "{{ item.key }} = {{ item.value }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_bluetooth_main_conf }}"
  notify: restart bluetooth
  tags: lp_bluetooth_conf

- name: "bluetooth: Enable and start bluetooth"
  systemd:
    name: "{{ lp_bluetooth_service }}"
    enabled: true
```

(continues on next page)

(continued from previous page)

```yaml
    state: started
  when: lp_bluetooth_enable|bool
  tags: lp_bluetooth_enable

- name: "bluetooth: Stop and disable bluetooth"
  systemd:
    name: "{{ lp_bluetooth_service }}"
    enabled: false
    state: stopped
  when: not lp_bluetooth_enable|bool
  tags: lp_bluetooth_disable

# EOF
...
```

3.1.9 cron.yml

Synopsis: Configure cron.

Description of the task.

[cron.yml]

```yaml
---
# linux-postinstall cron

- name: "cron: Configure cron variables"
  cronvar:
    name: "{{ item.name }}"
    value: "{{ item.value }}"
    user: "{{ item.user }}"
  loop: "{{ lp_cron_var }}"
  tags: lp_cron_var

- name: "cron: Configure cron"
  cron:
    state: "{{ item.state }}"
    user: "{{ item.user }}"
    name: "{{ item.name }}"
    minute: "{{ item.minute }}"
    hour: "{{ item.hour }}"
    day: "{{ item.day }}"
    month: "{{ item.month }}"
    weekday: "{{ item.weekday }}"
    job: "{{ item.command }}"
  loop: "{{ lp_cron_tab }}"
  tags: lp_cron_tab

# EOF
...
```

3.1.10 debsums.yml

Synopsis: Configure debsums.

Description of the task.

[debsums.yml]

```yaml
---
# linux-postinstall debsums

- name: "debsums: Debug"
  vars:
    msg: |
      lp_debsums_default_file [{{ lp_debsums_default_file }}]
      lp_debsums_default_conf
      {{ lp_debsums_default_conf|to_yaml }}
      lp_debsums_ignore_file [{{ lp_debsums_ignore_file }}]
      lp_debsums_ignore_conf
      {{ lp_debsums_ignore_conf|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_debsums_debug|bool
  tags: lp_debsums_debug

- name: "debsums: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_debsums_packages }}"
  tags: lp_debsums_packages

- name: "debsums: Configure {{ lp_debsums_default_file }}"
  lineinfile:
    dest: "{{ lp_debsums_default_file }}"
    state: "{{ item.state|default(omit) }}"
    regexp: '^\s*{{ item.key }}\s*=(.*)$'
    line: "{{ item.key }}={{ item.value }}"
    backup: "{{ lp_backup_conf }}"
    create: true
  loop: "{{ lp_debsums_default_conf }}"
  tags: lp_debsums_default_conf

- name: "debsums: Configure {{ lp_debsums_ignore_file }}"
  lineinfile:
    dest: "{{ lp_debsums_ignore_file }}"
    state: "{{ item.state|default(omit) }}"
    line: "{{ item }}"
    backup: "{{ lp_backup_conf }}"
    create: true
  loop: "{{ lp_debsums_ignore_conf }}"
  tags: lp_debsums_ignore_conf

# EOF
...
```

3.1.11 debug.yml

Synopsis: Configure debug.

Description of the task.

[debug.yml]

```yaml
---
# Hint: Get readable output with stdout_callback = yaml

- name: "Debug"
  vars:
    msg: |
      ansible_architecture [{{ ansible_architecture }}]
      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 }}]

      lp_vars_distro [{{ lp_vars_distro }}]
      lp_vars_distro_firstfound_skip [{{ lp_vars_distro_firstfound_skip }}]
      lp_vars_flavors [{{ lp_vars_flavors }}]
      lp_flavors_enable [{{ lp_flavors_enable }}]
      {{ my_release|default([])|to_nice_yaml }}

      lp_backup_conf [{{ lp_backup_conf }}]

      lp_acpi [{{ lp_acpi }}]
      lp_aliases [{{ lp_aliases }}]
      lp_apparmor [{{ lp_apparmor }}]
      lp_auto_upgrades [{{ lp_auto_upgrades }}] lp_auto_upgrades_enable [{{ lp_auto_upgrades_enable }}]
      lp_autofs [{{ lp_autofs }}] lp_autofs_enable [{{ lp_autofs_enable }}]
      lp_bluetooth [{{ lp_bluetooth }}] lp_bluetooth_enable [{{ lp_bluetooth_enable }}]
      lp_debsums [{{ lp_debsums }}]
      lp_gpg [{{ lp_gpg }}]
      lp_gpsd [{{ lp_gpsd }}] lp_gpsd_enable [{{ lp_gpsd_enable }}]
      lp_grub [{{ lp_grub }}]
      lp_iptables [{{ lp_iptables }}]
      lp_kvm [{{ lp_kvm }}]
      lp_latex [{{ lp_latex }}]
      lp_libvirt [{{ lp_libvirt }}]
      lp_libvirt_guests_enable [{{ lp_libvirt_guests_enable }}]
      lp_libvirt_libvirtd_enable [{{ lp_libvirt_libvirtd_enable }}]
      lp_lid [{{ lp_lid }}]
      lp_modemmanager [{{ lp_modemmanager }}] lp_modemmanager_enable [{{ lp_modemmanager_enable }}]
      lp_netplan [{{ lp_netplan }}]
      lp_nfsd [{{ lp_nfsd }}] lp_nfsd_enable [{{ lp_nfsd_enable }}]
      lp_packages_autoremove [{{ lp_packages_autoremove }}]
      lp_passwords [{{ lp_passwords }}]
      lp_pm [{{ lp_pm }}]
      lp_postfix [{{ lp_postfix }}] lp_postfix_enable [{{ lp_postfix_enable }}]
      lp_reboot [{{ lp_reboot }}]
      lp_resolvconf [{{ lp_resolvconf }}]
      lp_smart [{{ lp_smart }}] lp_smart_enable [{{ lp_smart_enable }}]
      lp_speechd [{{ lp_speechd }}] lp_speechd_enable [{{ lp_speechd_enable }}]
      lp_ssh [{{ lp_ssh }}]
      lp_sshd [{{ lp_sshd }}] lp_sshd_enable [{{ lp_sshd_enable }}]
      lp_swap [{{ lp_swap }}] lp_swap_enable [{{ lp_swap_enable }}]
      lp_timesyncd [{{ lp_timesyncd }}] lp_timesyncd_enable [{{ lp_timesyncd_enable }}]
```

(continues on next page)

(continued from previous page)

```
      lp_timezone [{{ lp_timezone }}]
      lp_tlp [{{ lp_tlp }}] lp_tlp_enable [{{ lp_tlp_enable }}]
      lp_ufw [{{ lp_ufw }}]
      lp_virtualbox [{{ lp_virtualbox }}] lp_virtualbox_enable [{{ lp_virtualbox_
↪enable }}]
      lp_wpagui [{{ lp_wpagui }}]
      lp_wpasupplicant [{{ lp_wpasupplicant }}]
      lp_xen [{{ lp_xen }}]
      lp_zeitgeist [{{ lp_zeitgeist }}]
      lp_zfs [{{ lp_zfs }}]

      lp_service
      {{ lp_service|to_yaml }}
      lp_package_state [{{ lp_package_state }}]
      lp_package_state_remove [{{ lp_package_state_remove }}]

  debug:
    msg: "{{ msg.split('\n')[:-1] }}"

# EOF
...
```

3.1.12 fstab.yml

Synopsis: Configure fstab.

Description of the task.

[fstab.yml]

```
---
# linux-postinstall fstab

- name: "fstab: Configure fstab entries"
  mount:
    name: "{{ item.name }}"
    state: "{{ item.state|default('mounted') }}"
    src: "{{ item.src|default(omit) }}"
    fstype: "{{ item.fstype|default(omit) }}"
    opts: "{{ item.opts|default(omit) }}"
    dump: "{{ item.dump|default(omit) }}"
    passno: "{{ item.passno|default(omit) }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_fstab_entries }}"

# EOF
...
```

3.1.13 gpg.yml

Synopsis: Configure gpg.

Description of the task.

[gpg.yml]

```yaml
---
# linux-postinstall gpg

- name: "gpg: Debug"
  vars:
    msg: |
      lp_gpg_install [{{ lp_gpg_install }}]
      lp_gpg_packages
      {{ lp_gpg_packages|to_nice_yaml }}
      lp_gpg_packages_extra
      {{ lp_gpg_packages_extra|to_nice_yaml }}

      lp_gpg_conf_default
      {{ lp_gpg_conf_default|to_yaml }}
      lp_gpg_conf
      {{ lp_gpg_conf|to_yaml }}

      lp_gpg_agent_conf_default
      {{ lp_gpg_agent_conf_default|to_yaml }}
      lp_gpg_agent_conf
      {{ lp_gpg_agent_conf|to_yaml }}

      lp_gpg_dirmngr_conf_default
      {{ lp_gpg_dirmngr_conf_default|to_yaml }}
      lp_gpg_dirmngr_conf
      {{ lp_gpg_dirmngr_conf|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_gpg_debug|bool
  tags: lp_gpg_debug

- name: "gpg: Install packages"
  include_tasks: fn/install-package.yml
  loop:
    - "{{ lp_gpg_packages }}"
    - "{{ lp_gpg_packages_extra }}"
  when: lp_gpg_install|bool
  tags: lp_gpg_packages

- name: "gpg: Configure gpg.conf"
  template:
    src: gpg.conf.j2
    dest: "{{ item.dest|default('/home/' ~ item.owner ~ '/.gnupg/gpg.conf') }}"
    owner: "{{ item.owner }}"
    group: "{{ item.owner }}"
    mode: "0600"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_gpg_conf }}"
  loop_control:
    label: "{{ item.owner }}"
  tags: lp_gpg_conf

- name: "gpg: Configure gpg-agent.conf"
  template:
    src: gpg-agent.conf.j2
    dest: "{{ item.dest|default('/home/' ~ item.owner ~ '/.gnupg/gpg-agent.conf') }}"
    owner: "{{ item.owner }}"
```

(continues on next page)

(continued from previous page)

```
    group: "{{ item.owner }}"
    mode: "0600"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_gpg_agent_conf }}"
  loop_control:
    label: "{{ item.owner }}"
  register: lp_gpg_agent_conf_changes
  notify: gpgconf kill gpg-agent
  tags: lp_gpg_agent_conf

- name: "gpg: Configure dirmngr.conf"
  template:
    src: gpg-dirmngr.conf.j2
    dest: "{{ item.dest|default('/home/' ~ item.owner ~ '/.gnupg/dirmngr.conf') }}"
    owner: "{{ item.owner }}"
    group: "{{ item.owner }}"
    mode: "0600"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_gpg_dirmngr_conf }}"
  loop_control:
    label: "{{ item.owner }}"
  register: lp_gpg_dirmngr_conf_changes
  notify: gpgconf kill dirmngr
  tags: lp_gpg_dirmngr_conf

# TODO: import keys

# EOF
...
```

3.1.14 gpsd.yml

Synopsis: Configure gpsd.

Description of the task.

[gpsd.yml]

```
---
# linux-postinstall gpsd

- name: "gpsd: Install packages for gpsd"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_gpsd_packages }}"
  tags: lp_gpsd_packages

- name: "gpsd: Add user gpsd to group dialout"
  user:
    name: gpsd
    groups: dialout
    append: true
  tags: lp_gpsd_group

- name: "gpsd: Configure /etc/bluetooth/rfcomm.conf"
  blockinfile:
    dest: /etc/bluetooth/rfcomm.conf
```

(continues on next page)

(continued from previous page)

```
    marker: "# {mark} ANSIBLE MANAGED BLOCK rfcomm{{ item.rfcomm }}"
    insertafter: EOF
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
    block: |
      rfcomm{{ item.rfcomm }} {
        bind {{ item.bind }}
        device {{ item.device }}
        channel {{ item.channel }}
        comment "{{ item.comment }}"
        }
  loop: "{{ lp_gpsd_bt_rfcomm }}"
  notify: restart bluetooth
  tags: lp_gpsd_bt_rfcom

- name: "gpsd: Configure /etc/default/gpsd"
  template:
    src: gpsd.j2
    dest: /etc/default/gpsd
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
  notify: restart gpsd
  tags: lp_gpsd_config

- name: "gpsd: Stop and disable gpsd"
  systemd:
    name: "{{ lp_gpsd_service }}"
    state: stopped
    enabled: false
  when: not lp_gpsd_enable|bool
  tags: lp_gpsd_service

- name: "gpsd: Start and enable gpsd"
  systemd:
    name: "{{ lp_gpsd_service }}"
    state: started
    enabled: true
  when: lp_gpsd_enable|bool
  tags: lp_gpsd_service

# EOF
...
```

3.1.15 grub.yml

Synopsis: Configure grub.

Description of the task.

[grub.yml]

```
---

- name: "grub: Debug"
  vars:
    msg: |
      lp_grub_default
      {{ lp_grub_default|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_grub_debug|bool
  tags: lp_grub_debug

- name: "grub: Configure /etc/default/grub"
  lineinfile:
    dest: /etc/default/grub
    regexp: '^\s*{{ item.var }}\s*=(.*)$'
    line: "{{ item.var }}={{ item.value }}"
    backup: "{{ lp_backup_conf|bool }}"
  loop: "{{ lp_grub_default }}"
  notify: update grub
  tags: lp_grub_conf

# EOF
...
```

3.1.16 hostname.yml

Synopsis: Configure hostname.

Description of the task.

[hostname.yml]

```
---
# linux-postinstall hostname

# TODO:
# 1) SET/DONT_SET hostname via DHCP
# /etc/dhcp/dhclient.conf
# #send host-name = gethostname();
# request host-name = "myhostname";
# https://askubuntu.com/questions/104918/how-to-get-the-hostname-from-a-dhcp-server
# http://blog.schlomo.schapiro.org/2013/11/setting-hostname-from-dhcp-in-debian.html
# https://askubuntu.com/questions/757423/how-to-force-dhcp-client-to-allow-a-self-
↪defined-domain-name

- name: "hostname: Configure hostname in /etc/hostname"
  template:
    src: hostname.j2
    dest: /etc/hostname
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
  when:
    - lp_hostname|length > 0
```

(continues on next page)

(continued from previous page)

```
    - ansible_os_family == "Debian"
#  notify: set hostname

- name: "hostname: Configure hostname"
  hostname:
    name: "{{ lp_hostname }}"
  when: lp_hostname|length > 0

# EOF
...
```

3.1.17 hosts.yml

Synopsis: Configure hosts.

Description of the task.

[hosts.yml]

```
---
# linux-postinstall hosts

- name: "hosts: Debug"
  vars:
    msg: |
      lp_hosts_default_override
      {{ lp_hosts_default_override|default('UNDEFINED')|to_yaml }}
      lp_hosts_default
      {{ lp_hosts_default|to_yaml }}
      lp_hosts
      {{ lp_hosts|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_hosts_debug|bool
  tags: lp_hosts_debug

- name: "hosts: Configure hosts in /etc/hosts"
  template:
    src: hosts.j2
    dest: /etc/hosts
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
  tags: lp_hosts_conf

# EOF
...
```

3.1.18 iptables.yml

Synopsis: Configure iptables.

Description of the task.

[iptables.yml]

```yaml
---
# linux-postinstall iptables

- name: "iptables: Create /etc/network/if-pre-up.d/iptables"
  template:
    src: iptables-restore.j2
    dest: /etc/network/if-pre-up.d/iptables
    owner: root
    group: root
    mode: "0755"

- name: "iptables: Create /etc/network/iptables
                   using {{ lp_iptables_type }}-iptables.j2"
  template:
    src: "{{ lp_iptables_type }}-iptables.j2"
    dest: /etc/network/iptables
    owner: root
    group: root
    mode: "0644"
  notify: reload iptables

# EOF
...
```

3.1.19 kvm.yml

Synopsis: Configure kvm.

Description of the task.

[kvm.yml]

```yaml
---
# linux-postinstall kvm

- name: "kvm: Debug"
  vars:
    msg: |
      lp_kvm_packages
      {{ lp_kvm_packages|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_kvm_debug|bool
  tags: lp_kvm_debug

- name: "kvm: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_kvm_packages }}"
  tags: lp_kvm_packages

# EOF
...
```

3.1.20 latex.yml

Synopsis: Configure latex.

Description of the task.

[latex.yml]

```yaml
---
# linux-postinstall LaTeX

- name: "latex: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_latex_packages }}"
  tags: lp_latex_packages

- name: "latex: Create directory /usr/share/texmf/tex/latex"
  file:
    state: directory
    path: /usr/share/texmf/tex/latex
  tags: lp_latex_dir

- name: "latex: Create directories for macros"
  file:
    state: directory
    path: "{{ item.dest }}"
  loop: "{{ lp_latex_macros }}"
  tags: lp_latex_macros

- name: "latex: Download macros"
  get_url:
    url: "{{ item.url }}"
    dest: "{{ item.dest }}"
    timeout: "{{ lp_latex_download_timeout }}"
  loop: "{{ lp_latex_macros }}"
  ignore_errors: "{{ lp_latex_get_url_ignore_errors }}"
  changed_when: false
  tags: lp_latex_labels

# get_url: check mode reports changes with force enabled
# https://github.com/ansible/ansible/issues/25418

# TODO:
# 1) Compile and register labels.sty
# cd /usr/share/texmf/tex/latex/labels/
# latex labels.ins
# texhash /usr/share/texmf

# EOF
...
```

3.1.21 libvirt-conf.yml

Synopsis: Configure libvirt-conf.

Description of the task.

[libvirt-conf.yml]

```yaml
---
# linux-postinstall libvirt-conf
```

(continues on next page)

(continued from previous page)

```

- name: "libvirt-conf: Debug"
  debug:
    msg: "{{ item }}"
  when: lp_libvirt_debug|bool

- name: "libvirt-conf: Configure {{ lp_libvirt_conf_dir }}/{{ item.key }}"
  lineinfile:
    dest: "{{ lp_libvirt_conf_dir }}/{{ item.key }}"
    regexp: '^{{ conf[0] }}(\s|=)(.*)$'
    line: "{{ conf[0] }} = {{ conf[1] }}"
    state: "{{ conf[2] | default('present') }}"
    backup: "{{ lp_backup_conf }}"
    create: true
  loop: "{{ item.value }}"
  loop_control:
    loop_var: conf
  notify:
    - reload libvirtd
    - reload libvirt_guests

# EOF
...
```

3.1.22 libvirt.yml

Synopsis: Configure libvirt.

Description of the task.

[libvirt.yml]

```
---
# linux-postinstall libvirt

- name: "libvirt Debug"
  vars:
    msg: |
      lp_libvirt_guests_enable [{{ lp_libvirt_guests_enable }}]
      lp_libvirt_libvirtd_enable [{{ lp_libvirt_libvirtd_enable }}]
      lp_libvirt_conf_owner [{{ lp_libvirt_conf_owner }}]
      lp_libvirt_conf_group [{{ lp_libvirt_conf_group }}]
      lp_libvirt_conf_mode [{{ lp_libvirt_conf_mode }}]
      lp_libvirt_conf_dir [{{ lp_libvirt_conf_dir }}]
      lp_libvirt_packages
      {{ lp_libvirt_packages|to_nice_yaml }}
      lp_libvirt_conf
      {{ lp_libvirt_conf|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_libvirt_debug|bool
  tags: lp_libvirt_debug

- name: "libvirt: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_libvirt_packages }}"
```

(continues on next page)

(continued from previous page)

```
  tags: lp_libvirt_pkg

- name: "libvirt: Configure"
  include_tasks: libvirt-conf.yml
  loop: "{{ lp_libvirt_conf|dict2items }}"
  tags: lp_libvirt_conf

- name: "libvirt: Start and enable {{ lp_libvirt_libvirtd_service }}"
  service:
    name: "{{ lp_libvirt_libvirtd_service }}"
    state: started
    enabled: true
  when: lp_libvirt_libvirtd_enable|bool
  tags: lp_libvirt_libvirtd_service

- name: "libvirt: Stop and disable {{ lp_libvirt_libvirtd_service }}"
  service:
    name: "{{ lp_libvirt_libvirtd_service }}"
    state: stopped
    enabled: false
  when: not lp_libvirt_libvirtd_enable|bool
  tags: lp_libvirt_libvirtd_service

- name: "libvirt: Start and enable {{ lp_libvirt_guests_service }}"
  service:
    name: "{{ lp_libvirt_guests_service }}"
    state: started
    enabled: true
  when: lp_libvirt_guests_enable|bool
  tags: lp_libvirt_guests_service

- name: "libvirt: Stop and disable {{ lp_libvirt_guests_service }}"
  service:
    name: "{{ lp_libvirt_guests_service }}"
    state: stopped
    enabled: false
  when: not lp_libvirt_guests_enable|bool
  tags: lp_libvirt_guests_service

# EOF
...
```

3.1.23 lid.yml

Synopsis: Configure lid.

Description of the task.

[lid.yml]

```
---
# linux-postinstall lid

- name: "lid: Configure {{ lp_lid_logind_conf }}"
  lineinfile:
    dest: "{{ lp_lid_logind_conf }}"
```

(continues on next page)

(continued from previous page)

```yaml
    regexp: '^\s*{{ item.var }}\s*=\s*(.*)$'
    line: "{{ item.var }}={{ item.value }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_lid_logind_conf_vars }}"
  notify: logind message reboot

- name: "lid: Configure {{ lp_lid_upower_conf }}"
  lineinfile:
    dest: "{{ lp_lid_upower_conf }}"
    regexp: '^\s*{{ item.var }}\s*=\s*(.*)$'
    line: "{{ item.var }}={{ item.value }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_lid_upower_conf_vars }}"

# EOF
...
```

3.1.24 logrotate.yml

Synopsis: Configure logrotate.

Description of the task.

[logrotate.yml]

```yaml
---
# linux-postinstall logrotate

- name: "logrotate: Install packages for logrotate"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_logrotate_packages }}"

- name: "logrotate: Configure blocks in {{ lp_logrotate_conf_file }}"
  blockinfile:
    path: "{{ lp_logrotate_conf_file }}"
    mark: "{{ item.mark }}"
    block: "{{ item.block }}"
    state: "{{ item.state }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_logrotate_conf_blocks }}"

- name: "logrotate: Configure lines in {{ lp_logrotate_conf_file }}"
  lineinfile:
    path: "{{ lp_logrotate_conf_file }}"
    line: "{{ item.line }}"
    state: "{{ item.state }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_logrotate_conf_lines }}"

- name: "logrotate: Configure {{ lp_logrotate_conf_dir }}"
  blockinfile:
    path: "{{ lp_logrotate_conf_dir }}/{{ item.path }}"
    block: "{{ item.conf }}"
    create: true
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_logrotate_confd }}"
```

(continues on next page)

(continued from previous page)

```

# EOF
...
```

3.1.25 modemmanager.yml

Synopsis: Configure modemmanager.

Description of the task.

[modemmanager.yml]

```
---
# linux-postinstall ModemManager

- name: "modem_manager: Configure /etc/init/modemmanager.override"
  template:
    src: modem-manager-override.j2
    dest: /etc/init/modemmanager.override
    owner: root
    group: root
    mode: "0644"

- name: "modem_manager: Stop and disable ModemManager"
  service:
    name: ModemManager
    state: stopped
    enabled: false
  when: not lp_modemmanager_enable|bool

- name: "modem_manager: Start and enable ModemManager"
  service:
    name: ModemManager
    state: started
    enabled: true
  when: lp_modemmanager_enable|bool

# EOF
...
```

3.1.26 modules.yml

Synopsis: Configure modules.

Description of the task.

[modules.yml]

```
---
# linux-postinstall modules

- name: "modules: Debug"
  vars:
    msg: |
      lp_modules_conf [{{ lp_modules_conf }}]
```

(continues on next page)

(continued from previous page)

```
      lp_modules
      {{ lp_modules|to_yaml }}
      lp_modules_options_path [ {{ lp_modules_options_path }}]
      lp_modules_options
      {{ lp_modules_options|to_nice_yaml }}
      lp_modules_blacklist_path [{{ lp_modules_blacklist_path }}]
      lp_modules_blacklist
      {{ lp_modules_blacklist|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_modules_debug|bool

- name: "modules: modprobe modules"
  modprobe:
    name: "{{ item.name }}"
    params: "{{ item.params }}"
    state: "{{ item.state|default('present') }}"
  loop: "{{ lp_modules }}"

# Debian
- name: "modules: Configure {{ lp_modules_conf }} in Debian"
  lineinfile:
    dest: "{{ lp_modules_conf }}"
    regexp: '^\s*{{ item.name }}\s*(.*)$'
    line: "{{ item.name }} {{ item.params }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_modules }}"
  when:
    - ansible_os_family == "Debian"
    - item.state|default("present") == "present"

# RedHat
- name: "modules: Configure {{ lp_modules_conf }} in RedHat"
  lineinfile:
    dest: "{{ lp_modules_conf }}"
    regexp: '^\s*modprobe\s+{{ item.name }}\s*(.*)$'
    line: "modprobe {{ item.name }} {{ item.params }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_modules }}"
  when:
    - ansible_os_family == "RedHat"
    - item.state|default("present") == "present"

- name: "modules: Blacklist modules in {{ lp_modules_blacklist_path }}"
  template:
    src: blacklist-module.j2
    dest: "{{ lp_modules_blacklist_path }}/blacklist-{{ item }}.conf"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_modules_blacklist }}"
  notify: update initramfs

- name: "modules: Set modules options in {{ lp_modules_options_path }}"
  template:
    src: options-module.j2
    dest: "{{ lp_modules_options_path }}/{{ item.module }}.conf"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_modules_options }}"
```

(continues on next page)

(continued from previous page)

```
  notify: update initramfs

# EOF
...
```

3.1.27 netplan.yml

Synopsis: Configure netplan.

Description of the task.

[netplan.yml]

```
---
# linux-postinstall netplan

# Configure 01-network-manager-all.yaml only if it already exists
- name: "netplan: Stat {{ lp_netplan_default }}"
  stat:
    path: "{{ lp_netplan_root }}/{{ lp_netplan_default }}"
  register: result

- name: "netplan: Configure {{ lp_netplan_root }}/{{ lp_netplan_default }}"
  template:
    src: netplan-default.j2
    dest: "{{ lp_netplan_root }}/{{ lp_netplan_default }}"
    owner: "{{ lp_netplan_owner }}"
    group: "{{ lp_netplan_group }}"
    mode: "{{ lp_netplan_mode }}"
    backup: "{{ lp_backup_conf }}"
  notify: netplan apply
  when: result.stat.exists|default(false)

- name: "netplan: Configure files in {{ lp_netplan_root }}"
  template:
    src: netplan-conf.j2
    dest: "{{ lp_netplan_root }}/{{ item.file }}"
    owner: "{{ item.owner | default(lp_netplan_owner) }}"
    group: "{{ item.group | default(lp_netplan_group) }}"
    mode: "{{ item.mode | default(lp_netplan_mode) }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_netplan_conf }}"
  notify: netplan apply

# EOF
...
```

3.1.28 nfsd.yml

Synopsis: Configure nfsd.

Description of the task.

[nfsd.yml]

```yaml
---
# linux-postinstall nfsd

- name: "nfsd: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_nfsd_packages }}"
  tags: lp_nfsd_packages

- name: "nfsd: Configure exports"
  template:
    src: exports.j2
    dest: /etc/exports
    owner: root
    group: root
    mode: "0644"
  notify: reload nfsd
  tags: lp_nfsd_exports

- name: "nfsd: Enable and start nfsd services"
  systemd:
    name: "{{ item }}"
    enabled: true
    state: started
  loop: "{{ lp_nfsd_services }}"
  when:
    - lp_nfsd_enable|bool
    - lp_nfsd_services|length > 0
  tags: lp_nfsd_service

- name: "nfsd: Stop and disable nfsd services"
  systemd:
    name: "{{ item }}"
    enabled: false
    state: stopped
  loop: "{{ lp_nfsd_services }}"
  when:
    - not lp_nfsd_enable|bool
    - lp_nfsd_services|length > 0
  tags: lp_nfsd_service

# EOF
...
```

3.1.29 packages-auto.yml

Synopsis: Configure packages-auto.

Description of the task.

[packages-auto.yml]

```yaml
---
# linux-postinstall packages-auto

- name: "packages-auto: Init variable local_pkg_lists"
  set_fact:
```

(continues on next page)

(continued from previous page)

```yaml
    local_pkg_lists: []
    local_pkg_list: []
  tags: lp_packages_auto

- name: "packages-auto: List variables ^lp_.*_packages$"
  set_fact:
    local_pkg_lists: "{{ local_pkg_lists +
                         [{'install': item.split('_')[0] + '_' + item.split('_')[1],
                           'packages': item}] }}"
  loop: "{{ hostvars[inventory_hostname].keys()|
            select('match', '^lp_.*_packages$')|
            list }}"
  tags: lp_packages_auto

- name: "packages-auto: Debug local_pkg_lists"
  debug:
    msg: "[{{ lookup('vars', item.install, default='false')|bool }}]
          packages {{ lookup('vars', item.packages) }}"
  loop: "{{ local_pkg_lists }}"
  when: lp_packages_debug|bool
  tags: lp_packages_auto

- name: "packages-auto: Create local_pkg_list"
  set_fact:
    local_pkg_list: "{{ local_pkg_list + lookup('vars', item.packages) }}"
  loop: "{{ local_pkg_lists }}"
  when: lookup('vars', item.install, default='False')|bool
  tags: lp_packages_auto

- name: "packages-auto: Debug local_pkg_list"
  debug:
    var: local_pkg_list
  when: lp_packages_debug|bool
  tags: lp_packages_auto

- name: "packages-auto: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ local_pkg_list }}"
  tags: lp_packages_auto

# EOF
...
```

3.1.30 packages.yml

Synopsis: Configure packages.

Description of the task.

[packages.yml]

```yaml
---
# linux-postinstall packages

- name: "packages: Debug"
  vars:
```

(continues on next page)

(continued from previous page)

```
    msg: |
      ansible_os_family [{{ ansible_os_family }}]
      lp_packages_auto [{{ lp_packages_auto }}]
      lp_packages_autoremove [{{ lp_packages_autoremove }}]
      lp_packages_selections_preinstall
      {{ lp_packages_selections_preinstall|to_nice_yaml }}
      lp_packages_install
      {{ lp_packages_install|to_nice_yaml }}
      lp_packages_remove
      {{ lp_packages_remove|to_nice_yaml }}
      lp_packages_selections_postinstall
      {{ lp_packages_selections_postinstall|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_packages_debug|bool
  tags: lp_packages_debug

- name: "packages: Configure package selections before Install/Remove"
  dpkg_selections:
    name: "{{ item.name }}"
    selection: "{{ item.selection }}"
  loop: "{{ lp_packages_selections_preinstall }}"
  when: ansible_os_family == "Debian"
  tags: lp_packages_selections_preinstall

- name: "packages: Install packages listed in variables lp_*_packages"
  include_tasks: packages-auto.yml
  when: lp_packages_auto|bool
  tags: lp_packages_auto

- name: "packages: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_packages_install }}"
  tags: lp_packages_install

- name: "packages: Remove packages"
  include_tasks: fn/remove-package.yml
  loop: "{{ lp_packages_remove }}"
  tags: lp_packages_remove

- name: "packages: Configure package selections after Install/Remove"
  dpkg_selections:
    name: "{{ item.name }}"
    selection: "{{ item.selection }}"
  loop: "{{ lp_packages_selections_postinstall }}"
  when: ansible_os_family == "Debian"
  tags: lp_packages_selections_postinstall

# EOF
...
```

3.1.31 passwords.yml

Synopsis: Configure passwords.

Description of the task.

[passwords.yml]

```
---
# linux-postinstall passwords

- name: "passwords: Debug"
  vars:
    msg: |
      lp_passwords_fail_gracefully [{{ lp_passwords_fail_gracefully }}]
      lp_password_update_password [{{ lp_password_update_password }}]
      lp_users
      {% if lp_passwords_debug_classified|bool %}
      {{ lp_users|default([])|to_nice_yaml }}
      {% else %}
      {% for user in lp_users|default([]) %}
      - userpass: ************
      {% for k,v in user.items() %}
      {% if k not in ['userpass'] %}
        {{ k }}: {{ v }}
      {% endif %}
      {% endfor %}
      {% endfor %}
      {% endif %}

      lp_passwordstore [{{ lp_passwordstore }}]
      lp_passwordstore_install [{{ lp_passwordstore_install }}]
      lp_passwordstore_debug [{{ lp_passwordstore_debug }}]
      lp_passwordstore_backup [{{ lp_passwordstore_backup }}]
      lp_passwordstore_create [{{ lp_passwordstore_create }}]
      lp_passwordstore_length [{{ lp_passwordstore_length }}]
      lp_passwordstore_nosymbols [{{ lp_passwordstore_nosymbols }}]
      lp_passwordstore_overwrite [{{ lp_passwordstore_overwrite }}]
      lp_passwordstore_passwordstore [{{ lp_passwordstore_passwordstore }}]
      lp_passwordstore_returnall  [{{ lp_passwordstore_returnall }}]
      lp_passwordstore_subkey [{{ lp_passwordstore_subkey }}]
      lp_passwordstore_idempotent_password_hash [{{ lp_passwordstore_idempotent_
↪password_hash }}]
      lp_passwordstore_packages
      {{ lp_passwordstore_packages|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_passwords_debug|bool
  tags: lp_passwords_debug

- name: "passwords: Passwordstore"
  block:
    - name: "passwords: Passwordstore: Install packages"
      include_tasks: fn/install-package.yml
      loop:
        - "{{ lp_passwordstore_packages }}"
        - "{{ lp_gpg_packages }}"
        - "{{ lp_gpg_packages_extra }}"
      vars:
        my_delegate_to_localhost: true
      run_once: true
      when: lp_passwordstore_install|bool
    - name: "passwords: Passwordstore: Retrieve, create, or update userpass"
      include_role:
```

(continues on next page)

(continued from previous page)

```
        name: vbotka.ansible_lib
        tasks_from: al_pws_user_host.yml
      vars:
        al_pws_debug: "{{ lp_passwordstore_debug }}"
        al_pws_backup: "{{ lp_passwordstore_backup }}"
        al_pws_create: "{{ lp_passwordstore_create }}"
        al_pws_length: "{{ lp_passwordstore_length }}"
        al_pws_nosymbols: "{{ lp_passwordstore_nosymbols }}"
        al_pws_overwrite: "{{ lp_passwordstore_overwrite }}"
        al_pws_passwordstore: "{{ lp_passwordstore_passwordstore }}"
        al_pws_returnall: "{{ lp_passwordstore_returnall }}"
        al_pws_subkey: "{{ lp_passwordstore_subkey }}"
        al_pws_idempotent_password_hash: "{{ lp_passwordstore_idempotent_password_
↪hash }}"
        al_pws_query: "{{ lp_users }}"
      register: result
    - name: "passwords: Passwordstore: Create my_passwords"
      set_fact:
        my_passwords: "{{ my_passwords|default([]) +
                          [item|dict2items|
                          rejectattr('key', 'equalto', 'userpass')|
                          list|items2dict|
                          combine({'update_password': lp_password_update_password})] }
↪}"
      loop: "{{ al_pws_query_result }}"
      loop_control:
        label: "{{ item.name }}"
    - name: "passwords: Passwordstore: Debug my_passwords"
      debug:
        var: my_passwords
      when: lp_passwords_debug|bool
    - name: "passwords:  Passwordstore: Include users"
      include_tasks: users.yml
      vars:
        lp_users: "{{ my_passwords }}"
  rescue:
    - name: "passwords: Passwordstore: Debug fail"
      debug:
        var: result
      when: lp_passwords_debug_classified|bool
    - name: "passwords: Passwordstore: Fail"
      fail:
        msg: "[ERROR] Passwordstore failed."
      when: not lp_passwords_fail_gracefully|bool
  when: lp_passwordstore|bool
  tags: lp_passwords_passwordstore

# EOF
...
```

3.1.32 pm-utils.yml

Synopsis: Configure pm-utils.

Description of the task.

[pm-utils.yml]

```
---
# linux-postinstall pm-utils

# TODO:
# 1) add variables: lp_pm_powerd, lp_pm_configd
# 2) add templates: pm-powerd.j2, pm-configd.j2
# 3) add cases: resume, thaw, suspend, hibernate
# 4) install pm-utils

- name: "pm_utils: Configure /etc/pm/sleep.d"
  template:
    src: pm-sleepd.j2
    dest: "/etc/pm/sleep.d/{{ item.value.file }}"
    owner: root
    group: root
    mode: "0755"
    backup: "{{ lp_backup_conf }}"
  with_dict: "{{ lp_pm_sleepd|default({}) }}"
  when: item.value.file|length > 0

# EOF
...
```

3.1.33 postfix.yml

Synopsis: Configure postfix.

Description of the task.

[postfix.yml]

```
---
# linux-postinstall postfix

- name: "postfix: Debug"
  vars:
    msg: |
      ansible_os_family [{{ ansible_os_family }}]
      lp_postfix_service [{{ lp_postfix_service }}]
      lp_postfix_enable [{{ lp_postfix_enable }}]
      lp_postfix_main_conf
      {{ lp_postfix_main_conf|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_postfix_debug|bool
  tags: lp_postfix_debug

- name: "postfix: Configure /etc/postfix/main.cf"
  lineinfile:
    dest: /etc/postfix/main.cf
    regexp: '^\s*{{ item.key }}\s*=\s*(.*)$'
    line: "{{ item.key }} = {{ item.value }}"
    create: true
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_postfix_main_conf }}"
```

(continues on next page)

(continued from previous page)

```
  notify: reload postfix
  tags: lp_postfix_conf

- name: "postfix: Enable and start postfix"
  systemd:
    name: "{{ lp_postfix_service }}"
    enabled: true
    state: started
  when: lp_postfix_enable|bool
  tags: lp_postfix_service

- name: "postfix: Disable and stop postfix"
  systemd:
    name: "{{ lp_postfix_service }}"
    enabled: false
    state: stopped
  when: not lp_postfix_enable|bool
  tags: lp_postfix_service

# EOF
...
```

3.1.34 reboot.yml

Synopsis: Configure reboot.

Description of the task.

[reboot.yml]

```
---
# linux-postinstall reboot

- name: "reboot Debug"
  vars:
    msg: |
      lp_reboot_force [{{ lp_reboot_force }}]
      lp_reboot_required_ignore [{{ lp_reboot_required_ignore }}]
      lp_reboot_required_file [{{ lp_reboot_required_file }}]
      lp_reboot_command [{{ lp_reboot_command }}]
      lp_reboot_wait_connect_timeout [{{ lp_reboot_wait_connect_timeout }}]
      lp_reboot_wait_sleep [{{ lp_reboot_wait_sleep }}]
      lp_reboot_wait_delay [{{ lp_reboot_wait_delay }}]
      lp_reboot_wait_timeout [{{ lp_reboot_wait_timeout }}]
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_reboot_debug|bool

- name: "reboot: Debian test {{ lp_reboot_required_file }}"
  block:
    - name: "reboot: Stat {{ lp_reboot_required_file }}"
      stat:
        path: "{{ lp_reboot_required_file }}"
      register: reboot_required_file_status
    - name: "reboot: Set reboot_required"
      set_fact:
```

(continues on next page)

(continued from previous page)

```
        reboot_required: "{{ reboot_required_file_status.exists|
                             default(false) }}"
  when: ansible_os_family == "Debian"

- name: "reboot: RedHat test {{ lp_reboot_required_command }}"
  block:
    - name: "reboot: Run {{ lp_reboot_required_command }}"
      command: "{{ lp_reboot_required_command }}"
      register: reboot_required_cmd_status
    - name: "reboot: Set reboot_required"
      set_fact:
        reboot_required: "{{ (reboot_required_cmd_status.rc != 0)|
                              ternary(true, false) }}"
  when: ansible_os_family == "RedHat"

- name: "reboot: Debug reboot_required"
  debug:
    var: reboot_required
  when: lp_reboot_debug|bool

- name: "reboot: Reboot and wait for connection"
  reboot:
    connect_timeout: "{{ lp_reboot_wait_connect_timeout }}"
    post_reboot_delay: "{{ lp_reboot_wait_delay }}"
    reboot_timeout: "{{ lp_reboot_wait_timeout }}"
  when: (reboot_required|default(false) and
        (not lp_reboot_required_ignore)) or
        lp_reboot_force|bool

# - name: "reboot: Reboot and wait for connection"
#   block:
#     - name: "reboot: Reboot"  # noqa 305
#       shell: "{{ lp_reboot_command }}"
#       async: 1
#       poll: 0
#     - name: "reboot: Wait for connection"
#       wait_for_connection:
#         connect_timeout: "{{ lp_reboot_wait_connect_timeout }}"
#         sleep: "{{ lp_reboot_wait_sleep }}"
#         delay: "{{ lp_reboot_wait_delay }}"
#         timeout: "{{ lp_reboot_wait_timeout }}"
#   when: (reboot_required|default(false) and
#         (not lp_reboot_required_ignore)) or lp_reboot_force

# EOF
...
```

3.1.35 repos.yml

Synopsis: Configure repos.

Description of the task.

[repos.yml]

```
---
# linux-postinstall repos

- name: "repos: Debug"
  vars:
    msg: |
      lp_repos_keys
      {{ lp_repos_keys|to_nice_yaml }}
      lp_repos
      {{ lp_repos|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_repos_debug|bool
  tags: lp_repos_debug

- name: "repos: Manage repo signing keys"
  apt_key:
    data: "{{ item.data|default(omit) }}"
    file: "{{ item.file|default(omit) }}"
    id: "{{ item.id|default(omit) }}"
    keyring: "{{ item.keyring|default(omit) }}"
    keyserver: "{{ item.keyserver|default(omit) }}"
    state: "{{ item.state|default(omit) }}"
    url: "{{ item.url|default(omit) }}"
    validate_certs: "{{ item.validate_certs|default(omit) }}"
  loop: "{{ lp_repos_keys }}"
  register: result
  retries: "{{ lp_install_retries }}"
  until: result is succeeded
  delay: "{{ lp_install_delay }}"
  tags: lp_repos_keys_manage

- name: "repos: Manage repositories"
  apt_repository:
    codename: "{{ item.codename|default(omit) }}"
    filename: "{{ item.filename|default(omit) }}"
    mode: "{{ item.mode|default(omit) }}"
    repo: "{{ item.repo|mandatory }}"
    state: "{{ item.state|default(omit) }}"
    update_cache: "{{ item.update_cache|default(omit) }}"
    validate_certs: "{{ item.validate_certs|default(omit) }}"
  loop: "{{ lp_repos }}"
  tags: lp_repos_manage

# EOF
...
```

3.1.36 resolvconf.yml

Synopsis: Configure resolvconf.

Description of the task.

[resolvconf.yml]

```yaml
---
# linux-postinstall resolvconf

- name: "resolvconf: Debug"
  vars:
    msg: |
      lp_resolvconf_service [{{ lp_resolvconf_service }}]
      lp_resolvconf_enable [{{ lp_resolvconf_enable }}]
      lp_package_state [{{ lp_package_state }}]
      lp_resolvconf_packages
      {{ lp_resolvconf_packages|to_nice_yaml }}
      lp_resolvconf_confd_head_path [{{ lp_resolvconf_confd_head_path }}]
      lp_resolvconf_conf_owner [{{ lp_resolvconf_conf_owner }}]
      lp_resolvconf_conf_group [{{ lp_resolvconf_conf_group }}]
      lp_resolvconf_conf_mode [{{ lp_resolvconf_conf_mode }}]
      lp_resolvconf_confd_head
      {{ lp_resolvconf_confd_head|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_resolvconf_debug|bool
  tags: lp_resolvconf_debug

- name: "resolvconf: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_resolvconf_packages }}"
  tags: lp_resolvconf_packages

- name: "resolvconf: Configure {{ lp_resolvconf_confd_head_path }}"
  template:
    src: resolvconf-confd-head.j2
    dest: "{{ lp_resolvconf_confd_head_path }}"
    owner: "{{ lp_resolvconf_conf_owner }}"
    group: "{{ lp_resolvconf_conf_group }}"
    mode: "{{ lp_resolvconf_conf_mode }}"
    backup: "{{ lp_backup_conf }}"
  notify: restart resolvconf
  tags: lp_resolvconf_confd_head

- name: "resolvconf: Enable and start resolvconf"
  systemd:
    name: "{{ lp_resolvconf_service }}"
    enabled: true
    state: started
  when: lp_resolvconf_enable|bool
  tags: lp_resolvconf_service

- name: "resolvconf: Disable and stop resolvconf"
  systemd:
    name: "{{ lp_resolvconf_service }}"
    enabled: false
    state: stopped
  when: not lp_resolvconf_enable|bool
  tags: lp_resolvconf_service

# EOF
...
```

3.1.37 service.yml

Synopsis: Configure service.

Description of the task.

[service.yml]

```yaml
---
# linux-postinstall service

- name: "service: Set my_service_name_vars"
  set_fact:
    my_service_name_vars: "{{ my_service_name_vars|default([]) +
                              [{item: lookup('vars', 'lp_' ~ item ~ '_service')}] }}"
  loop: "{{ lp_service_enable }}"
  when: lookup('vars', 'lp_' ~ item, default='false')
  tags: lp_service_debug

- name: "service: Set my_service_enable_vars"
  set_fact:
    my_service_enable_vars: "{{ my_service_enable_vars|default([]) +
                                [{item: lookup('vars', 'lp_' ~ item ~ '_enable')}] }}"
  loop: "{{ lp_service_enable }}"
  when: lookup('vars', 'lp_' ~ item, default='false')
  tags: lp_service_debug

- name: "service: Debug"
  debug:
    msg: "{{ my_msg.split('\n')[:-1] }}"
  vars:
    my_msg: |
      lp_service
      {{ lp_service|to_nice_yaml }}
      lp_service_enable
      {{ lp_service_enable|to_nice_yaml }}
      my_service_name_vars
      {{ my_service_name_vars|default([])|to_nice_yaml }}
      my_service_enable_vars
      {{ my_service_enable_vars|default([])|to_nice_yaml }}
  when: lp_service_debug|bool
  tags: lp_service_debug

- name: "service: Automaticaly enable or disable services managed by this role"
  service:
    name: "{{ lookup('vars', 'lp_' ~ item ~ '_service') }}"
    enabled: "{{ lookup('vars', 'lp_' ~ item ~ '_enable') }}"
  loop: "{{ lp_service_enable }}"
  when: lookup('vars', 'lp_' ~ item, default='false')
  tags: lp_service_auto

- name: "service: General managent of services"
  service:
    name: "{{ item.name }}"
    state: "{{ item.state|default(omit) }}"
    enabled: "{{ item.enabled|default(omit) }}"
    arguments: "{{ item.arguments|default(omit) }}"
    pattern: "{{ item.pattern|default(omit) }}"
    runlevel: "{{ item.runlevel|default(omit) }}"
```

(continues on next page)

(continued from previous page)

```
    sleep: "{{ item.sleep|default(omit) }}"
    use: "{{ item.use|default('omit') }}"
  loop: "{{ lp_service }}"
  when: (item.state is defined) or
        (item.enabled is defined)
  tags: lp_service_general

# TODO: Mask a service. Do not allow any servce to activate a masked
# service. See tasks/wpagui.yml

# EOF
...
```

3.1.38 smart.yml

Synopsis: Configure smart.

Description of the task.

[smart.yml]

```
---
# linux-postinstall smart

- name: "smart: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_smart_packages }}"
  tags: lp_smart_packages

- name: "smart: Configure {{ lp_smart_conf_file }}. Do not scan for devices"
  lineinfile:
    state: absent
    dest: "{{ lp_smart_conf_file }}"
    regexp: '^\s*DEVICESCAN\s*(.*)$'
    owner: "{{ lp_smart_conf_owner }}"
    group: "{{ lp_smart_conf_group }}"
    mode: "{{ lp_smart_conf_mode }}"
    create: true
    backup: "{{ lp_backup_conf }}"
  when: not lp_smart_devicescan|bool
  notify: reload smart
  tags: lp_smart_conf

- name: "smart: Configure devices in {{ lp_smart_conf_file }}"
  lineinfile:
    dest: "{{ lp_smart_conf_file }}"
    regexp: "{{ item.regexp }}"
    line: "{{ item.line }}"
    owner: "{{ lp_smart_conf_owner }}"
    group: "{{ lp_smart_conf_group }}"
    mode: "{{ lp_smart_conf_mode }}"
    create: true
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_smart_devices }}"
  notify: reload smart
  tags: lp_smart_conf
```

(continues on next page)

(continued from previous page)

```

- name: "smart: Start and enable smart"
  service:
    name: "{{ lp_smart_service }}"
    state: started
    enabled: true
  register: result
  when: lp_smart_enable|bool
  tags: lp_smart_service

- name: "smart: Debug service"
  debug:
    var: result
  when: lp_smart_debug|bool

- name: "smart: Stop and disable smart"
  service:
    name: "{{ lp_smart_service }}"
    state: stopped
    enabled: false
  register: result
  when: not lp_smart_enable|bool
  tags: lp_smart_service

- name: "smart: Debug service"
  debug:
    var: result
  when: lp_smart_debug|bool

# EOF
...
```

3.1.39 speechd.yml

Synopsis: Configure speechd.

Description of the task.

[speechd.yml]

```
---
# linux-postinstall speechd

- name: "speechd: Debug"
  debug:
    msg: "lp_speechd_enable [{{ lp_speechd_enable }}]"
  when: lp_speechd_debug|bool

- name: "speechd: Enable and start speech-dispatcher"
  systemd:
    name: "{{ lp_speechd_service }}"
    enabled: true
    state: started
  when: lp_speechd_enable|bool

- name: "speechd: Stop and disable speech-dispatcher"
```

(continues on next page)

(continued from previous page)

```yaml
  systemd:
    name: "{{ lp_speechd_service }}"
    enabled: false
    state: stopped
  when: not lp_speechd_enable|bool

# EOF
...
```

3.1.40 sshd.yml

Synopsis: Configure sshd.

Description of the task.

[sshd.yml]

```yaml
---
# linux-postinstall sshd

- name: "sshd: Debug"
  vars:
    msg: |
      lp_sshd_config
      {{ lp_sshd_config|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_sshd_debug|bool
  tags: lp_sshd_debug

- name: "sshd: Configure /etc/ssh/sshd_config"
  lineinfile:
    dest: /etc/ssh/sshd_config
    regexp: "^\\s*{{ item.key }}\\s*(.*)$"
    insertbefore: "^{{ '#' }}{{ item.key }}\\s*(.*)$"
    line: "{{ item.key }} {{ item.value }}"
    backup: "{{ lp_backup_conf }}"
    validate: "{{ lp_sshd_path }} -t -f %s"
  loop: "{{ lp_sshd_config }}"
  notify: reload sshd
  tags: lp_sshd_config

- name: "sshd: Enable and start sshd"
  systemd:
    name: "{{ lp_sshd_service }}"
    enabled: true
    state: started
  when: lp_sshd_enable|bool
  tags: lp_sshd_service

- name: "sshd: Disable and stop sshd"
  systemd:
    name: "{{ lp_sshd_service }}"
    enabled: false
    state: stopped
  when: not lp_sshd_enable|bool
```

(continues on next page)

(continued from previous page)

```yaml
  tags: lp_sshd_service

# EOF
...
```

3.1.41 ssh.yml

Synopsis: Configure ssh.

Description of the task.

[ssh.yml]

```yaml
---
# linux-postinstall ssh

- name: "ssh: Debug"
  vars:
    msg: |
      lp_ssh_config
      {{ lp_ssh_config|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_ssh_debug|bool
  tags: lp_ssh_debug

- name: "ssh: Configure /etc/ssh/ssh_config"
  template:
    src: ssh_config.j2
    dest: /etc/ssh/ssh_config
    backup: "{{ lp_backup_conf }}"
  tags: lp_ssh_conf

# EOF
...
```

3.1.42 sudoers.yml

Synopsis: Configure sudoers.

Description of the task.

[sudoers.yml]

```yaml
---
# linux-postinstall sudoers

- name: "sudoers: Debug"
  vars:
    msg: |
      lp_sudoers_conf
      {{ lp_sudoers_conf|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_sudoers_debug|bool
```

(continues on next page)

(continued from previous page)

```
  tags: lp_sudoers_debug

- name: "sudoers: Configure /etc/sudoers"
  lineinfile:
    path: /etc/sudoers
    line: "{{ item.line }}"
    state: "{{ item.state|default('present') }}"
    create: true
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_sudoers_conf }}"
  tags: lp_sudoers_conf

- name: "sudoers: Configure /etc/sudoers.d/01"
  lineinfile:
    path: /etc/sudoers.d/01
    line: "{{ item }}"
    owner: "{{ lp_sudoers_owner }}"
    group: "{{ lp_sudoers_group }}"
    mode: "{{ lp_sudoers_mode }}"
    create: true
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_sudoers_01 }}"
  tags: lp_sudoers_dconf

# EOF
...
```

3.1.43 swap.yml

Synopsis: Configure swap.

Description of the task.

[swap.yml]

```
---
# linux-postinstall swap

- name: "swap: Debug"
  vars:
    msg: |
      lp_swap [{{ lp_swap }}]
      lp_swap_enable [{{ lp_swap_enable }}]
      lp_swap_file [{{ lp_swap_file|default("UNDEFINED") }}]
      lp_swap_size [{{ lp_swap_size|default("UNDEFINED") }}]
      lp_swap_stsize [{{ lp_swap_stsize|default("UNDEFINED") }}]
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_swap_debug|bool
  tags: lp_swap_debug

- name: "swap: Create swapfile {{ lp_swap_file }}"
  shell: sh -c 'if [ ! -e {{ lp_swap_file }} ]; then printf "create"; fi'
  register: command_result
  changed_when: command_result.stdout == "create"
  notify: create and mount swap file
```

(continues on next page)

(continued from previous page)

```
  tags: lp_swap_swapfile

- name: "swap: Change swapfile {{ lp_swap_file }}"
  shell: >
    sh -c
    'if [ -e {{ lp_swap_file }} ] &&
    [ "`stat --format '%s' {{ lp_swap_file }}`" -ne "{{ lp_swap_stsize }}" ];
    then printf "change";
    fi'
  register: command_result
  changed_when: command_result.stdout == "change"
  notify: change and mount swap file
  tags: lp_swap_swapfile

- name: "swap: Create swap entry in /etc/fstab"
  mount:
    name: "none"
    src: "{{ lp_swap_file }}"
    fstype: swap
    opts: sw
    passno: "0"
    dump: "0"
    state: present
    backup: "{{ lp_backup_conf }}"
  when: lp_swap_enable|bool
  tags: lp_swap_fstab

- name: "swap: Remove swap entry from /etc/fstab"
  mount:
    name: "none"
    src: "{{ lp_swap_file }}"
    fstype: swap
    opts: sw
    passno: 0
    dump: 0
    state: absent
    backup: "{{ lp_backup_conf }}"
  notify: remove swap file
  when:
    - not lp_swap_enable|bool
    - lp_swap_file is defined
  tags: lp_swap_swapfile

# EOF
...
```

3.1.44 sysctl.yml

Synopsis: Configure sysctl.

Description of the task.

[sysctl.yml]

```
---
# linux-postinstall sysctl
```

(continues on next page)

(continued from previous page)

```yaml

- name: "sysctl: Debug"
  vars:
    msg: |
      lp_sysctl_vars
      {{ lp_sysctl_vars|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_sysctl_debug|bool
  tags: lp_sysctl_debug

- name: "sysctl: Configure /etc/sysctl.conf"
  lineinfile:
    dest: /etc/sysctl.conf
    regexp: '^\s*{{ item.var }}\s*=(.*)$'
    line: "{{ item.var }} = {{ item.value }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_sysctl_vars }}"
  notify: load sysctl settings

# EOF
...
```

3.1.45 timesyncd.yml

Synopsis: Configure timesyncd.

Description of the task.

[timesyncd.yml]

```yaml
---
# linux-postinstall timesyncd

- name: "timesyncd: Debug"
  vars:
    msg: |
      lp_timesyncd [{{ lp_timesyncd }}]
      lp_timesyncd_NTP [{{ lp_timesyncd_NTP }}]
      lp_timesyncd_FallbackNTP [{{ lp_timesyncd_FallbackNTP }}]
      lp_timesyncd_RootDistanceMaxSec [{{ lp_timesyncd_RootDistanceMaxSec }}]
      lp_timesyncd_PollIntervalMinSec [{{ lp_timesyncd_PollIntervalMinSec }}]
      lp_timesyncd_PollIntervalMaxSec [{{ lp_timesyncd_PollIntervalMaxSec }}]
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_timesyncd_debug|bool
  tags: lp_timesyncd_debug

- name: "timesyncd: Configure /etc/systemd/timesyncd.conf"
  template:
    src: timesyncd.conf.j2
    dest: /etc/systemd/timesyncd.conf
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
```

(continues on next page)

(continued from previous page)

```
  notify: restart timesyncd
  tags: lp_timesyncd_conf

- name: "timesyncd: Enable and start timesyncd"
  service:
    name: "{{ lp_timesyncd_service }}"
    state: started
    enabled: true
  when: lp_timesyncd_enable|bool
  tags: lp_timesyncd_service

- name: "timesyncd: Disable and stop timesyncd"
  service:
    name: "{{ lp_timesyncd_service }}"
    state: stopped
    enabled: false
  when: not lp_timesyncd_enable|bool
  tags: lp_timesyncd_service

# Notes on CentOS
# * systemd compiled without timesyncd service in CentOS 7 ?
# * use ntpd or chrony only ?
# https://unix.stackexchange.com/questions/286708/
# centos-7-2-minimal-time-synchronization-timedated-and-or-ntpd-chrony
# https://www.freedesktop.org/wiki/Software/systemd/timedated/

# EOF
...
```

3.1.46 timezone.yml

Synopsis: Configure timezone.

Description of the task.

[timezone.yml]

```
---
# linux-postinstall timezone

- name: "timezone: Debug"
  debug:
    msg: "lp_timezone_zoneinfo [{{ lp_timezone_zoneinfo }}]"
  when: lp_timezone_debug|bool
  tags: lp_timezone_debug

- name: "timezone: Set timezone {{ lp_timezone_zoneinfo }}"
  timezone:
    name: "{{ lp_timezone_zoneinfo|default('UTC') }}"
  tags: lp_timezone_set

# EOF
...
```

3.1.47 tlp.yml

Synopsis: Configure tlp.

Description of the task.

[tlp.yml]

```yaml
---
# linux-postinstall tlp

- name: "tlp: Debug"
  vars:
    msg: |
      lp_tlp_enable [{{ lp_tlp_enable }}]
      lp_tlp_thinkpad [{{ lp_tlp_thinkpad }}]
      lp_tlp_packages
      {{ lp_tlp_packages|to_nice_yaml }}
      lp_tlp_packages_tp
      {{ lp_tlp_packages_tp|to_nice_yaml }}
      lp_tlp_config_file [{{ lp_tlp_config_file }}]
      lp_tlp_config
      {{ lp_tlp_config|to_nice_yaml }}
      lp_tlp_services
      {{ lp_tlp_services|to_nice_yaml }}
      lp_tlp_restart_service [{{ lp_tlp_restart_service }}]
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_tlp_debug|bool
  tags: lp_tlp_debug

- name: "tlp: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_tlp_packages }}"
  tags: lp_tlp_packages

- name: "tlp: Install packages for ThinkPad"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_tlp_packages_tp }}"
  when: lp_tlp_thinkpad|bool
  tags: lp_tlp_packages

- name: "tlp: Configure {{ lp_tlp_config_file }}"
  lineinfile:
    dest: "{{ lp_tlp_config_file }}"
    regexp: '^\s*{{ item.key }}\s*=\s*(.*)$'
    line: "{{ item.key }}={{ item.value }}"
    create: true
  loop: "{{ lp_tlp_config }}"
  notify: restart tlp
  tags: lp_tlp_conf

- name: "tlp: Start and enable tlp"
  systemd:
    name: "{{ item }}"
    state: started
    enabled: true
  loop: "{{ lp_tlp_services }}"
  when: lp_tlp_enable|bool
```

(continues on next page)

(continued from previous page)

```yaml
  tags: lp_tlp_service

- name: "tlp: Stop and disable tlp"
  systemd:
    name: "{{ item }}"
    state: stopped
    enabled: false
  loop: "{{ lp_tlp_services }}"
  when: not lp_tlp_enable|bool
  tags: lp_tlp_service

# EOF
...
```

3.1.48 udev.yml

Synopsis: Configure udev.

Description of the task.

[udev.yml]

```yaml
---
# linux-postinstall udev

- name: "udev: Debug"
  vars:
    msg: |
      lp_udev_rules_dir [{{ lp_udev_rules_dir }}]
      lp_udev_rules_template [{{ lp_udev_rules_template }}]
      lp_udev_rules
      {{ lp_udev_rules|to_nice_yaml }}

      lp_udev_persistent_net_template [{{ lp_udev_persistent_net_template }}]
      lp_udev_persistent_net_rules_file [{{ lp_udev_persistent_net_rules_file }}]
      lp_udev_persistent_net_rules
      {{ lp_udev_persistent_net_rules|to_nice_yaml }}

      lp_udev_hci_name_rules_file [{{ lp_udev_hci_name_rules_file }}]
      lp_udev_hci_name_rules
      {{ lp_udev_hci_name_rules|to_nice_yaml }}

      lp_udev_hci_run_rules_file [{{ lp_udev_hci_run_rules_file }}]
      lp_udev_hci_run_rules
      {{ lp_udev_hci_run_rules|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_udev_debug|bool
  tags: lp_udev_debug

# udev rules
- name: "udev: Configure {{ lp_udev_rules_dir }}"
  template:
    src: "{{ lp_udev_rules_template }}"
    dest: "{{ lp_udev_rules_dir }}/{{ item.key }}"
    owner: root
```

(continues on next page)

(continued from previous page)

```
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_udev_rules|dict2items }}"
  notify: reload udev
  tags: lp_udev_rules

# persistent_net
- name: "udev: Configure {{ lp_udev_rules_dir }}/
                         {{ lp_udev_persistent_net_rules_file }}"
  template:
    src: "{{ lp_udev_persistent_net_template }}"
    dest: "{{ lp_udev_rules_dir }}/{{ lp_udev_persistent_net_rules_file }}"
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_udev_persistent_net_rules }}"
  notify: reload udev
  tags: lp_udev_persistentnet

# hci name
- name: "udev: Configure {{ lp_udev_rules_dir }}/
                         {{ lp_udev_hci_name_rules_file }}"
  template:
    src: hci-name.rules.j2
    dest: "{{ lp_udev_rules_dir }}/{{ lp_udev_hci_name_rules_file }}"
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_udev_hci_name_rules }}"
  notify: reload udev
  tags: lp_udev_hciname

# hci run
- name: "udev: Configure {{ lp_udev_rules_dir }}/
                         {{ lp_udev_hci_run_rules_file }}"
  template:
    src: hci-run.rules.j2
    dest: "{{ lp_udev_rules_dir }}/{{ lp_udev_hci_run_rules_file }}"
    owner: root
    group: root
    mode: "0644"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_udev_hci_run_rules }}"
  notify: reload udev
  tags: lp_udev_hcirun

# Service
- name: "udev: Start and enable udev"
  service:
    name: "{{ lp_udev_service }}"
    state: started
    enabled: true
  when: lp_udev_enable|bool
  tags: lp_udev_service
```

(continues on next page)

(continued from previous page)

```yaml

- name: "udev: Stop and disable udev"
  service:
    name: "{{ lp_udev_service }}"
    state: stopped
    enabled: false
  when: not lp_udev_enable|bool
  tags: lp_udev_service

# EOF
...
```

3.1.49 ufw.yml

Synopsis: Configure ufw.

Description of the task.

[ufw.yml]

```yaml
---
# linux-postinstall ufw

#   Notes
#
# * Aliases of parameters in ufw module not implemented in task
#   "Configure ufw".
# * It's not necessary to reload ufs after configuration has
#   changed. Module ufw automatically updates the rules.
# * Best practice: First time 'lp_ufw_reset: true'; configure and enable
#   ufs (configuration item {state: 'enabled'} reloads firewall and
#   enables firewall on boot); 'lp_ufw_enable: true' start and enable ufw
#   service.
# * Configuration on the fly: configure and enable ufs.
# * The last configuration item should be {state: 'enabled'}.
# * See: man ufw.

- name: "ufw: Debug"
  vars:
    msg: |
      lp_ufw_enable [{{ lp_ufw_enable }}]
      lp_ufw_reset [{{ lp_ufw_reset }}]
      lp_ufw_reload [{{ lp_ufw_reload }}]
      lp_ufw_packages
      {{ lp_ufw_packages|to_nice_yaml }}
      lp_ufw_conf
      {{ lp_ufw_conf|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_ufw_debug|bool
  tags: lp_ufw_debug

- name: "ufw: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_ufw_packages }}"
  tags: lp_ufw_packages
```

(continues on next page)

(continued from previous page)

```

- name: "ufw: Disable and reset firewall to installation defaults"
  ufw:
    state: reset
  when: lp_ufw_reset|bool
  tags: lp_ufw_reset

- name: "ufw: Reload firewall"
  ufw:
    state: reloaded
  when: lp_ufw_reload|bool
  tags: lp_ufw_reload

- name: "ufw: Configure ufw"
  ufw:
    comment: "{{ item.comment|default(omit) }}"
    default: "{{ item.default|default(omit) }}"
    delete: "{{ item.delete|default(omit) }}"
    direction: "{{ item.direction|default(omit) }}"
    from_ip: "{{ item.from_ip|default(omit) }}"
    from_port: "{{ item.from_port|default(omit) }}"
    insert: "{{ item.insert|default(omit) }}"
    insert_relative_to: "{{ item.insert_relative_to|default(omit) }}"
    interface: "{{ item.interface|default(omit) }}"
    log: "{{ item.log|default(omit) }}"
    logging: "{{ item.logging|default(omit) }}"
    name: "{{ item.name|default(omit) }}"
    proto: "{{ item.proto|default(omit) }}"
    route: "{{ item.route|default(omit) }}"
    rule: "{{ item.rule|default(omit) }}"
    state: "{{ item.state|default(omit) }}"
    to_ip: "{{ item.to_ip|default(omit) }}"
    to_port: "{{ item.to_port|default(omit) }}"
  loop: "{{ lp_ufw_conf }}"
  tags: lp_ufw_conf

- name: "ufw: Start and enable ufw"
  service:
    name: "{{ lp_ufw_service }}"
    state: started
    enabled: true
  register: result
  when: lp_ufw_enable|bool
  tags: lp_ufw_service

- name: "ufw: Debug enabled service"
  debug:
    var: result
  when:
    - lp_ufw_enable|bool
    - lp_ufw_debug|bool
  tags: lp_ufw_service

- name: "ufw: Stop and disable ufw"
  service:
    name: "{{ lp_ufw_service }}"
    state: stopped
```

(continues on next page)

(continued from previous page)

```yaml
    enabled: false
  register: result
  when: not lp_ufw_enable|bool
  tags: lp_ufw_service

- name: "ufw: Debug disabled service"
  debug:
    var: result
  when:
    - not lp_ufw_enable|bool
    - lp_ufw_debug|bool
  tags: lp_ufw_service

# EOF
...
```

3.1.50 users.yml

Synopsis: Configure users.

Description of the task.

[users.yml]

```yaml
---
# linux-postinstall users

- name: "users: Debug"
  vars:
    msg: |
      lp_users
      {{ lp_users|default(['UNDEFINED'])|to_nice_yaml }}
      lp_users_groups
      {{ lp_users_groups|default(['UNDEFINED'])|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_users_debug|bool
  tags: lp_users_debug

- name: "users: Manage user accounts"
  user:
    name: "{{ item.name }}"
    authorization: "{{ item.authorization|default(omit) }}"
    comment: "{{ item.comment|default(omit) }}"
    create_home: "{{ item.create_home|default(omit) }}"
    expires: "{{ item.expires|default(omit) }}"
    force: "{{ item.force|default(omit) }}"
    generate_ssh_key: "{{ item.generate_ssh_key|default(omit) }}"
    group: "{{ item.group|default(omit) }}"
    hidden: "{{ item.hidden|default(omit) }}"
    home: "{{ item.home|default(omit) }}"
    local: "{{ item.local|default(omit) }}"
    login_class: "{{ item.login_class|default(omit) }}"
    move_home: "{{ item.move_home|default(omit) }}"
    non_unique: "{{ item.non_unique|default(omit) }}"
    password: "{{ item.password|default(omit) }}"
```

(continues on next page)

(continued from previous page)

```
    password_lock: "{{ item.password_lock|default(omit) }}"
    profile: "{{ item.profile|default(omit) }}"
    remove: "{{ item.remove|default(omit) }}"
    role: "{{ item.role|default(omit) }}"
    seuser: "{{  item.seuser|default(omit) }}"
    shell: "{{ item.shell|default(omit) }}"
    skeleton: "{{ item.skeleton|default(omit) }}"
    ssh_key_bits: "{{ item.ssh_key_bits|default(omit) }}"
    ssh_key_comment: "{{ item.ssh_key_comment|default(omit) }}"
    ssh_key_file: "{{ item.ssh_key_file|default(omit) }}"
    ssh_key_passphrase: "{{ item.ssh_key_passphrase|default(omit) }}"
    ssh_key_type: "{{ item.ssh_key_type|default(omit) }}"
    state: "{{ item.state|default(omit) }}"
    system: "{{ item.system|default(omit) }}"
    uid: "{{ item.uid|default(omit) }}"
    update_password: "{{ item.update_password|default(omit) }}"
  loop: "{{ lp_users|default([]) }}"
  loop_control:
    label: "{{ item.name }}"
  tags: lp_users_accounts

- name: "users: Add users to additional groups"
  user:
    name: "{{ item.name }}"
    groups: "{{ item.groups }}"
    append: "{{ item.append|default(true) }}"
  loop: "{{ lp_users_groups|default([]) }}"
  tags: lp_users_groups

#  EOF
...
```

3.1.51 vars.yml

Synopsis: Configure vars.

Description of the task.

[vars.yml]

```
---
# linux-postinstall vars

- name: "vars: Include firstfound default vars"
  when: lp_vars_distro == "firstfound"
  include_tasks: vars-firstfound.yml

- name: "vars: Include incremental default vars"
  when: lp_vars_distro == "incremental"
  include_tasks: vars-firstfound.yml

- name: "vars: Include firstfound default vars for various flavors"
  when: lp_flavors_enable|bool
  include_tasks: sub/vars-flavors.yml

# TODO "vars: Include incremental default vars for various flavors"
```

(continues on next page)

(continued from previous page)

```

# EOF
...
```

3.1.52 vars-firstfound.yml

Synopsis: Configure OS specific default vars. Method first_found.

Description of the task.

[vars-firstfound.yml]

```yaml
---
# linux-postinstall vars-firstfound

- name: "vars-firstfound: Include default vars for
              [{{ ansible_distribution_release }},
               {{ ansible_distribution }},
               {{ ansible_os_family }}]"
  include_vars: "{{ lookup('first_found', params) }}"
  register: result
  vars:
    params:
      files:
        - "{{ ansible_distribution }}-{{ ansible_distribution_release }}.yml"
        - "{{ ansible_distribution }}.yml"
        - "{{ ansible_os_family }}.yml"
        - default.yml
        - defaults.yml
      paths:
        - "{{ role_path }}/vars/defaults"
# [TODO]
#     skip: "{{ lp_vars_distro_firstfound_skip|bool }}"
# skip doesn't work with first_found lookup #43833
# https://github.com/ansible/ansible/issues/43833
# workaround: Create empty defaults.yml

- name: "vars-firstfound: Debug include default vars from"
  debug:
    var: result.ansible_included_var_files
  when: lp_debug|bool

- name: "vars-firstfound: Include custom vars for
              [{{ ansible_distribution_release }},
               {{ ansible_distribution }},
               {{ ansible_os_family }}]"
  register: result
  include_vars: "{{ lookup('first_found', params) }}"
  vars:
    params:
      files:
        - "{{ ansible_distribution }}-{{ ansible_distribution_release }}.yml"
        - "{{ ansible_distribution }}.yml"
        - "{{ ansible_os_family }}.yml"
        - default.yml
        - defaults.yml
```

(continues on next page)

(continued from previous page)

```
      paths:
        - "{{ role_path }}/vars"
# [TODO]
#     skip: "{{ lp_vars_distro_firstfound_skip|bool }}"
# skip doesn't work with first_found lookup #43833
# https://github.com/ansible/ansible/issues/43833
# workaround: Create empty defaults.yml

- name: "vars-firstfound: Debug include custom vars from"
  debug:
    var: result.ansible_included_var_files
  when: lp_debug|bool

# EOF
...
```

3.1.53 vars-incremental.yml

Synopsis: Configure OS specific default vars. Method incremental.

Description of the task.

[vars-incremental.yml]

```
---
# linux-postinstall vars-incremental

- name: "vars-incemental: Include default vars for
              [{{ ansible_os_family }},
               {{ ansible_distribution }},
               {{ ansible_distribution_release }}]"
  include_vars: "{{ item }}"
  register: result
  loop:
    - "{{ my_path }}/defaults.yml"
    - "{{ my_path }}/default.yml"
    - "{{ my_path }}/{{ ansible_os_family }}.yml"
    - "{{ my_path }}/{{ ansible_distribution }}.yml"
    - "{{ my_path }}/{{ ansible_distribution }}-{{ ansible_distribution_release }}.yml
↪"
  when: item is exists
  vars:
    my_path: "{{ role_path }}/vars/defaults.incr"

- name: "vars-incemental: Debug include default vars"
  debug:
    var: result
  when: lp_debug|bool

- name: "vars-incemental: Include custom vars for
              [{{ ansible_os_family }},
               {{ ansible_distribution }},
               {{ ansible_distribution_release }}]"
  include_vars: "{{ item }}"
  register: result
  loop:
```

(continues on next page)

(continued from previous page)

```
    - "{{ my_path }}/defaults.yml"
    - "{{ my_path }}/default.yml"
    - "{{ my_path }}/{{ ansible_os_family }}.yml"
    - "{{ my_path }}/{{ ansible_distribution }}.yml"
    - "{{ my_path }}/{{ ansible_distribution }}-{{ ansible_distribution_release }}.yml
↪"
  when: item is exists
  vars:
    my_path: "{{ role_path }}/vars"

- name: "vars-incemental: Debug include custom vars"
  debug:
    var: result
  when: lp_debug|bool

# EOF
...
```

3.1.54 virtualbox.yml

Synopsis: Configure virtualbox.

Description of the task.

[virtualbox.yml]

```
---
# linux-postinstall virtualbox

- name: "virtualbox: Debug"
  vars:
    msg: |
      ansible_lsb.description [{{ ansible_lsb.codename }}]
      lp_virtualbox [{{ lp_virtualbox }}]
      lp_virtualbox_ignore_errors [{{ lp_virtualbox_ignore_errors }}]
      lp_virtualbox_keys [{{ lp_virtualbox_keys }}]
      lp_virtualbox_repos [{{ lp_virtualbox_repos }}]
      lp_virtualbox_install
      {{ lp_virtualbox_install|to_nice_yaml }}
      lp_virtualbox_services
      {{ lp_virtualbox_services|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_virtualbox_debug|bool
  tags: lp_virtualbox_debug

# TODO: assert lp_virtualbox_modules are loaded
#       when: lp_virtualbox|bool

- name: "virtualbox: Add signing key of VirtualBox"
  apt_key:
    url: "{{ item }}"
    state: present
  loop: "{{ lp_virtualbox_keys }}"
  register: result
  retries: "{{ lp_install_retries }}"
```

(continues on next page)

(continued from previous page)

```yaml
  until: result is succeeded
  delay: "{{ lp_install_delay }}"
  ignore_errors: "{{ lp_virtualbox_ignore_errors }}"
  tags: lp_virtualbox_keys

- name: "virtualbox: Add repository of VirtualBox"
  apt_repository:
    repo: "{{ item }}"
    state: present
  loop: "{{ lp_virtualbox_repos }}"
  ignore_errors: "{{ lp_virtualbox_ignore_errors }}"
  tags: lp_virtualbox_repos

- name: "virtualbox: Install VirtualBox packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_virtualbox_packages }}"
  ignore_errors: "{{ lp_virtualbox_ignore_errors }}"
  tags: lp_virtualbox_pkg

- name: "virtualbox: Enable and start services"
  service:
    name: "{{ item }}"
    state: started
    enabled: true
  loop: "{{ lp_virtualbox_services }}"
  when: lp_virtualbox_enable|bool
  tags: lp_virtualbox_services

- name: "virtualbox: Disable and stop services"
  service:
    name: "{{ item }}"
    state: stopped
    enabled: false
  loop: "{{ lp_virtualbox_services }}"
  when: not lp_virtualbox_enable|bool
  tags: lp_virtualbox_services

# EOF
...
```

3.1.55 wpagui.yml

Synopsis: Configure wpagui.

Description of the task.

[wpagui.yml]

```yaml
---
# linux-postinstall wpa_gui
# Install wpa_gui and disable NetworkManager

- name: "wpagui: Debug"
  vars:
    msg: |
      lp_wpagui_packages
```

(continues on next page)

(continued from previous page)

```
      {{ lp_wpagui_packages|to_nice_yaml }}
      lp_wpagui_systemd
      {{ lp_wpagui_systemd|to_nice_yaml }}
      lp_wpagui_service
      {{ lp_wpagui_service|to_nice_yaml }}
      lp_wpagui_service_mask
      {{ lp_wpagui_service_mask|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_wpagui_debug|bool
  tags: lp_wpagui_debug

- name: "wpagui: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_wpagui_packages }}"
  tags: lp_wpagui_packages

- name: "wpagui: Disable NM /etc/init/network-manager.override"
  template:
    src: network-manager-override.j2
    dest: /etc/init/network-manager.override
    owner: root
    group: root
    mode: "0644"
  tags: lp_wpagui_disableNM

- name: "wpagui: Configure managed=false
                in /etc/NetworkManager/NetworkManager.conf"
  lineinfile:
    dest: /etc/NetworkManager/NetworkManager.conf
    regexp: '^\s*managed\s*=\s*(.*)$'
    line: "managed=false"
  tags: lp_wpagui_disableNM

# NetworkManager.service will be stopped and disabled in the next task
# - name: "wpagui: Stop and disable NM"
#   systemd:
#     name: "{{ item }}"
#     state: stopped
#     enabled: false
#   loop: "{{ lp_wpagui_systemd }}"
#   tags: lp_wpagui_disableNM

- name: "wpagui: Stop and disable all NM services"
  service:
    name: "{{ item }}"
    state: stopped
    enabled: false
  loop: "{{ lp_wpagui_service }}"
  tags: lp_wpagui_disableNM

- name: "wpagui: Mask NM services"
  command: "systemctl mask {{ item }}"
  args:
    warn: false
  loop: "{{ lp_wpagui_service_mask }}"
  changed_when: false
```

(continues on next page)

(continued from previous page)

```
  tags: lp_wpagui_mask_NM
  #
  # False Positives: Skipping Rules
  # https://github.com/ansible/ansible-lint#false-positives-skipping-rules
  # noqa 303 does not work
  #
  # ansible-lint "systemctl used in place of systemd module" No
  # systemctl / systemd module. #48848
  # https://github.com/ansible/ansible/issues/48848

# EOF
...
```

3.1.56 wpasupplicant.yml

Synopsis: Configure wpasupplicant.

Description of the task.

[wpasupplicant.yml]

```
---
# linux-postinstall wpasupplicant

- name: "wpasupplicant: Debug"
  vars:
    msg: |
      lp_package_state [{{ lp_package_state }}]
      lp_wpasupplicant_packages
      {{ lp_wpasupplicant_packages|to_nice_yaml }}
      lp_wpasupplicant_conf_only [{{ lp_wpasupplicant_conf_only }}]
      lp_wpasupplicant_conf_dir [{{ lp_wpasupplicant_conf_dir }}]
      lp_wpasupplicant_conf_file [{{ lp_wpasupplicant_conf_file }}]
      lp_wpasupplicant_conf_owner [{{ lp_wpasupplicant_conf_owner }}]
      lp_wpasupplicant_conf_group [{{ lp_wpasupplicant_conf_group }}]
      lp_wpasupplicant_conf_mode [{{ lp_wpasupplicant_conf_mode }}]
      lp_wpasupplicant_conf_ctrl_interface
      [{{ lp_wpasupplicant_conf_ctrl_interface }}]
      lp_wpasupplicant_conf_global
      {{ lp_wpasupplicant_conf_global|to_yaml }}
      lp_wpa_action_script [{{ lp_wpa_action_script }}]
      lp_wpa_action_script_dir [{{ lp_wpa_action_script_dir }}]
      lp_wpa_action_script_file [{{ lp_wpa_action_script_file }}]
      lp_wpa_action_script_owner [{{ lp_wpa_action_script_owner }}]
      lp_wpa_action_script_group [{{ lp_wpa_action_script_group }}]
      lp_wpa_action_script_mode [{{ lp_wpa_action_script_mode }}]
      lp_wpa_action_script_dhclient [{{ lp_wpa_action_script_dhclient }}]
      lp_wpa_action_script_pidfile [{{ lp_wpa_action_script_pidfile }}]
      lp_wpa_action_script_options_connect [{{ lp_wpa_action_script_options_connect }}
↪]
      lp_wpa_action_script_options_disconnect [{{ lp_wpa_action_script_options_
↪disconnect }}]
      lp_wpa_action_script_logfile [{{ lp_wpa_action_script_logfile }}]

      {% if lp_wpasupplicant_debug_classified|bool %}
      lp_wpasupplicant_conf
```

(continues on next page)

(continued from previous page)

```
      {{ lp_wpasupplicant_conf|to_yaml }}
      {% endif %}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_wpasupplicant_debug|bool
  tags: lp_wpasupplicant_debug

- name: "wpasupplicant: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_wpasupplicant_packages }}"
  tags: lp_wpasupplicant_packages

- name: "wpasupplicant: Configure {{ lp_wpasupplicant_conf_dir }}/{{
                                    lp_wpasupplicant_conf_file }}.DEV"
  template:
    src: wpa_supplicant.conf.j2
    dest: "{{ lp_wpasupplicant_conf_dir }}/{{
              lp_wpasupplicant_conf_file }}.{{
              item.dev }}"
    owner: "{{ lp_wpasupplicant_conf_owner }}"
    group: "{{ lp_wpasupplicant_conf_group }}"
    mode: "{{ lp_wpasupplicant_conf_mode }}"
    backup: "{{ lp_backup_conf }}"
  register: lp_wpasupplicant_conf_changes
  notify: reconfigure wpa_supplicant
  loop: "{{ lp_wpasupplicant_conf }}"
  loop_control:
    label: "{{ item.dev }}"
  no_log: "{{ not lp_wpasupplicant_debug_classified }}"
  tags: lp_wpasupplicant_conf

- name: "wpasupplicant: Debug: registered lp_wpasupplicant_conf_changes"
  debug:
    var: lp_wpasupplicant_conf_changes
  when: lp_wpasupplicant_debug_classified|bool

# - name: "wpasupplicant: Debug: wpa_cli reconfigure commands"
#  debug:
#    msg: >
#      'sh -c "[ -S {{ lp_wpasupplicant_conf_ctrl_interface }}/\
#                    {{ item.item.dev }} ] &&
#      wpa_cli -p {{ lp_wpasupplicant_conf_ctrl_interface }} \
#              -i {{ item.item.dev }} reconfigure"'
#  loop: "{{ lp_wpasupplicant_conf_changes.results }}"
#  when:
#    - item.changed
#    - lp_wpasupplicant_debug

- name: "wpasupplicant: Create dir {{ lp_wpa_action_script_dir }}"
  file:
    state: directory
    path: "{{ lp_wpa_action_script_dir }}"
    owner: "{{ lp_wpa_action_script_owner }}"
    group: "{{ lp_wpa_action_script_group }}"
  when: lp_wpa_action_script|bool
  tags: lp_wpa_action_script_dir

```

(continues on next page)

(continued from previous page)

```
- name: "wpasupplicant: Create script {{ lp_wpa_action_script_file }}"
  template:
    src: wpa_action.sh.j2
    dest: "{{ lp_wpa_action_script_file }}"
    owner: "{{ lp_wpa_action_script_owner }}"
    group: "{{ lp_wpa_action_script_group }}"
    mode: "{{ lp_wpa_action_script_mode }}"
    backup: "{{ lp_backup_conf }}"
  when: lp_wpa_action_script|bool
  tags: lp_wpa_action_script_file

# EOF
...
```

3.1.57 xen.yml

Synopsis: Configure xen.

Description of the task.

[xen.yml]

```
---
# linux-postinstall xen

- name: "xen: Debug"
  vars:
    msg: |
      lp_xen_packages
      {{ lp_xen_packages|to_nice_yaml }}
      lp_xen_dom0_mem
      {{ lp_xen_dom0_mem|to_nice_yaml }}
      lp_xen_default_grub_conf
      {{ lp_xen_default_grub_conf|to_nice_yaml }}
      lp_xen_global
      {{ lp_xen_global|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_xen_debug|bool
  tags: lp_xen_debug

- name: "xen: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_xen_packages }}"
  tags: lp_xen_packages

- name: "xen: Configure /etc/default/grub"
  lineinfile:
    dest: /etc/default/grub
    regexp: '^\s*{{ item.key }}\s*=\s*(.*)$'
    line: "{{ item.key }}={{ item.value }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_xen_default_grub_conf }}"
  notify: update grub
  tags: lp_xen_default_grub

```

(continues on next page)

(continued from previous page)

```yaml
- name: "xen: Configure /etc/xen/xl.conf"
  lineinfile:
    dest: /etc/xen/xl.conf
    regexp: '^\s*{{ item.var }}\s*=\s*(.*)$'
    line: "{{ item.var }}={{ item.value }}"
    create: true
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_xen_global }}"
  tags: lp_xen_global

# EOF
...
```

3.1.58 xorg.yml

Synopsis: Configure xorg.

Description of the task.

[xorg.yml]

```yaml
---
# linux-postinstall xen

- name: "xorg: Debug"
  vars:
    msg: |
      lp_xorg_conf
      {{ lp_xorg_conf|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_xorg_debug|bool
  tags: lp_xorg_debug

- name: "xorg: Configure {{ lp_xorg_conf_dir }}"
  template:
    src: xorg.conf.j2
    dest: "{{ lp_xorg_conf_dir }}/{{ item.file }}"
    backup: "{{ lp_backup_conf }}"
  loop: "{{ lp_xorg_conf }}"
  tags: lp_xorg_conf

# EOF
...
```

3.1.59 zeitgeist.yml

Synopsis: Configure zeitgeist.

Description of the task.

[zeitgeist.yml]

```
---
# linux-postinstall zeitgeist

# One-way atm
- name: Remove zeitgeist
  apt:
    state: absent
    name: zeitgeist
    purge: true
  when:
    - not lp_zeitgeist|bool
    - ansible_os_family == "Debian"

- name: Remove zeitgeist-*
  apt:
    state: absent
    name: zeitgeist-*
    purge: true
  when:
    - not lp_zeitgeist|bool
    - ansible_os_family == "Debian"

# - name: Disable zeitgeist
#   service:
#     name: zeitgeist
#     state: stopped
#     enabled: no
#   when: not lp_zeitgeist
# "Could not find the requested service zeitgeist"

# for i in zeitgeist-fts zeitgeist; do
# systemctl --user disable $i;
# systemctl --user stop $i;
# systemctl --user mask $i;
# done

# EOF
...
```

3.1.60 zfs.yml

Synopsis: Configure zfs.

Description of the task.

[zfs.yml]

```
---
# linux-postinstall zfs

- name: "zfs: Debug"
  vars:
    msg: |
      lp_zfs_install [{{ lp_zfs_install }}]
      lp_zfs_packages
      {{ lp_zfs_packages|to_nice_yaml }}
```

(continues on next page)

(continued from previous page)

```
      lp_zfs_services
      {{ lp_zfs_services|default([])|to_yaml }}
      lp_zfs_manage
      {{ lp_zfs_manage|to_yaml }}
      lp_zfs_mountpoints
      {{ lp_zfs_mountpoints|to_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_zfs_debug|bool
  tags: lp_zfs_debug

- name: "zfs: Install packages"
  include_tasks: fn/install-package.yml
  loop: "{{ lp_zfs_packages }}"
  when: lp_zfs_install|bool
  tags: lp_zfs_packages

- name: "zfs: Manage zfs services"
  systemd:
    name: "{{ item.name }}"
    enabled: "{{ item.enabled|default(true) }}"
    state: "{{ item.state|default('started') }}"
  loop: "{{ lp_zfs_services|default([]) }}"
  tags: lp_zfs_services

- name: "zfs: Manage zfs"
  zfs:
    name: "{{ item.name }}"
    state: "{{ item.state }}"
    origin: "{{ item.origin|default(omit) }}"
    extra_zfs_properties: "{{ item.extra_zfs_properties|default(omit) }}"
  loop: "{{ lp_zfs_manage }}"
  tags: lp_zfs_manage

- name: "zfs: Set mode and ownership of zfs mountpoints"
  file:
    state: directory
    path: "{{ item.mountpoint }}"
    owner: "{{ item.owner|default(omit) }}"
    group: "{{ item.group|default(omit) }}"
    mode: "{{ item.mode|default(omit) }}"
  loop: "{{ lp_zfs_mountpoints }}"
  tags: lp_zfs_mountpoints

# EOF
...
```

3.1.61 sub/vars-flavors.yml

Synopsis: Configure flavor specific variables.

Description of the task.

[sub/vars-flavors.yml]

```yaml
---
# linux-postinstall vars: vars-flavors

# Create dir lp_flavors_dir. Loop lp_flavors and get stat of
# release_file(s). If release_file exists include tasks specific to this
# flavor.

- name: 'sub: vars-flavors: Debug'
  vars:
    msg: |
      lp_flavors_dir [{{ lp_flavors_dir }}]
      lp_flavors_dir_owner [{{ lp_flavors_dir_owner }}]
      lp_flavors_dir_group [{{ lp_flavors_dir_group }}]
      lp_flavors_dir_mode [{{ lp_flavors_dir_mode }}]
      lp_flavors
      {{ lp_flavors|to_nice_yaml }}
  debug:
    msg: "{{ msg.split('\n')[:-1] }}"
  when: lp_debug|bool

- name: 'sub: vars-flavors: Create {{ lp_flavors_dir }}'
  file:
    state: directory
    path: '{{ lp_flavors_dir }}'
    owner: '{{ lp_flavors_dir_owner }}'
    group: '{{ lp_flavors_dir_group }}'
    mode: '{{ lp_flavors_dir_mode }}'
  delegate_to: localhost
  run_once: true

- name: 'sub: vars-flavors: Detect flavor'
  stat:
    path: '{{ item.value.release_file }}'
  loop: '{{ lp_flavors|dict2items }}'
  register: result

- name: 'sub: vars-flavors: Debug result'
  when: lp_debug|bool
  debug:
    msg: "{{ result.results|json_query('[?stat.exists].item') }}"

- name: 'sub: vars-flavors: Include tasks for flavor'
  include_tasks: "{{ 'sub/vars-flavors-' ~ outer_item.key ~ '.yml' }}"
  loop: "{{ result.results|json_query('[?stat.exists].item') }}"
  loop_control:
    loop_var: outer_item

# EOF
...
```

3.1.62 sub/vars-flavors-common.yml

Synopsis: Configure common flavor specific variables.

Description of the task.

[sub/vars-flavors-common.yml]

```yaml
---
# linux-postinstall vars: vars-flavors-common

# Fetch my_release_file from the remote host and store the file in
# lp_flavors_dir. Read release_attr from the fetched file and include
# vars that correspond the flavor, release and HW.

# my_release_file
- name: 'sub: vars-flavors-common: Set my_release_file'
  set_fact:
    my_release_file: '{{ outer_item.value.release_file }}'
- name: 'sub: vars-flavors-common: Debug my_release_file'
  when: lp_debug|bool
  debug:
    var: my_release_file

# my_flavor
- name: 'sub: vars-flavors-common: Set my_flavor'
  set_fact:
    my_flavor: '{{ outer_item.key }}'
- name: 'sub: vars-flavors-common: Debug my_flavor'
  when: lp_debug|bool
  debug:
    var: my_flavor

# my_release_file_fetch
- name: 'sub: vars-flavors-common: Set my_release_file_fetch'
  set_fact:
    my_release_file_fetch: "{{ lp_flavors_dir ~ '/' ~
                               inventory_hostname ~ '-' ~
                               my_flavor }}"
- name: 'sub: vars-flavors-common: Fetch {{ my_release_file }} to
                                         {{ my_release_file_fetch }}'
  fetch:
    flat: true
    src: '{{ my_release_file }}'
    dest: '{{ my_release_file_fetch }}'

# my_release_keys
- name: 'sub: vars-flavors-common: Clear my_release_keys'
  set_fact:
    my_release_keys: []
- name: 'sub: vars-flavors-common: Set my_release_keys'
  set_fact:
    my_release_keys: "{{ my_release_keys|
                         default([]) + [item.split('=').0|trim] }}"
  loop: "{{ lookup('file', my_release_file_fetch).splitlines() }}"
  when: item is match('^(\s*[a-zA-Z0-9_]+\s*)=(.*)$')
- name: 'sub: vars-flavors-common: Debug my_release_keys'
  when: lp_debug|bool
  debug:
    var: my_release_keys

# my_release_dict
- name: 'sub: vars-flavors-common: Clear my_release_dict'
  set_fact:
    my_release_dict: {}
```

(continues on next page)

(continued from previous page)

```
- name: 'sub: vars-flavors-common: Set my_release_dict attributes'
  set_fact:
    my_release_dict: "{{ my_release_dict|
                         combine({item: lookup('ini', item ~
                                                          ' type=properties file=' ~
                                                          my_release_file_fetch)}) }}"
  loop: '{{ my_release_keys }}'
- name: 'sub: vars-flavors-common: Debug my_release_dict'
  when: lp_debug|bool
  debug:
    var: my_release_dict

# my_release
- name: 'sub: vars-flavors-common: Add flavor to my_release'
  set_fact:
    my_release: '{{ my_release|
                    default({})|
                    combine({my_flavor: my_release_dict}) }}'
- name: 'sub: vars-flavors-common: Debug my_release'
  when: lp_debug|bool
  debug:
    var: my_release

# my_labels
- name: 'sub: vars-flavors-common: Set my_labels'
  set_fact:
    my_labels: "{{ lp_flavors[my_flavor].file_labels|
                   map('extract', my_release[my_flavor])|
                   list }}"
- name: 'sub: vars-flavors-common: Debug my_labels'
  when: lp_debug|bool
  debug:
    var: my_labels

# Include default vars for flavor
- name: 'sub: vars-flavors-common: Include default vars for flavor
              [{{ my_labels.1 }},
               {{ my_labels.0 }},
               {{ my_flavor }}]'
  include_vars: "{{ lookup('first_found', params) }}"
  vars:
    params:
      files:
        - '{{ my_flavor }}_{{ my_labels.0 }}_{{ my_labels.1 }}.yml'
        - '{{ my_flavor }}_{{ my_labels.0 }}.yml'
        - '{{ my_flavor }}.yml'
        - default.yml
        - defaults.yml
      paths:
        - '{{ role_path }}/vars/flavors'
# [TODO]
#     skip: "{{ lp_vars_distro_firstfound_skip|bool }}"
# skip doesn't work with first_found lookup #43833
# https://github.com/ansible/ansible/issues/43833
# workaround: Create empty defaults.yml

# EOF
```

(continues on next page)

(continued from previous page)

```
...
```

CHAPTER

FOUR

COPYRIGHT

CHAPTER

FIVE

LEGAL NOTICE

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX