
ansible-role-linux-postinstall Documentation

Release 1.14.3

Vladimir Botka

Jul 26, 2020

TABLE OF CONTENTS:

1	Quick start guide	3
2	User's guide	5
2.1	Introduction	6
2.2	Installation	6
2.3	Playbook	6
2.4	Debug	7
2.5	Tags	7
2.6	Tasks	8
2.7	Variables	24
2.8	Best practice	33
3	Annotated source code	35
3.1	Tasks	37
4	Copyright	105
5	Legal Notice	107
6	Indices and tables	109
	Index	111

This [role](#) and the documentation is work in progress. Feel free to [share your feedback](#) and [report issues](#). Contributions are welcome.

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 -l 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)

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

- Configure passwords

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

- Install packages and enable autoremove

```
1 shell> cat host_vars/srv.example.com/lp-packages.yml
2 lp_packages_autoremove: true
3 lp_packages_install:
4   - ansible
5   - ansible-lint
6   - ansible-tower-cli
```

- Configure services

```
1 shell> cat host_vars/srv.example.com/lp-service.yml
2 lp_service_debug: true
3 lp_service:
4   - {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 differences 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.

USER'S GUIDE**Table of Contents**

- *User's guide*
 - *Introduction*
 - *Installation*
 - *Playbook*
 - *Debug*
 - *Tags*
 - *Tasks*
 - * *Netplan*
 - *Synopsis*
 - *Examples*
 - * *Passwords*
 - *Synopsis*
 - *Passwordstore*
 - *Examples*
 - * *ZFS*
 - *Synopsis*
 - *Examples*
 - *Variables*
 - * *Default variables*
 - * *Default OS specific variables*
 - * *Custom OS specific variables*
 - * *Flavors specific variables (WIP)*
 - *Best practice*
 - * *Recommended configuration after the installation of OS*
 - * *Flavors*

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.linear_postinstall
```

and install it

```
shell> ansible-galaxy install vbotka.linear_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.linear_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_rfc,
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-*.yaml` and check the syntax

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

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

```
shell> ansible-playbook linux-postinstall.yaml -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.yaml -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.yaml
- hosts: test_01
  become: true
  roles:
    - vbotka.linux_postinstall
```

Create `host_vars/test_01/lp-netplan.yaml`

```
shell> cat host_vars/test_01/lp-netplan.yaml
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 parameters of the Ansible module `user` will be omitted 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 parameters of the Ansible module `user` will be omitted 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
1rLy0eVpJiTzpj-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.linear-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.linear-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.linear_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.linear_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.linear_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.linear_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.linear_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`]

```
1  ---
2  # linux_postinstall defaults
3
4  # -----
5  # Put distro and flavor specific vars and overrides to the vars
6  # directory.
7
8  lp_vars_distro: firstfound
9  lp_vars_distro_firstfound_skip: true # issue #43833
10 lp_vars_flavors: firstfound
11
12 lp_hostname: ""
13 lp_fqdn: ""
14
15 lp_debug: false
```

(continues on next page)

(continued from previous page)

```

16 lp_backup_conf: false
17
18 lp_install_retries: 5
19 lp_install_delay: 2
20
21 lp_repos_debug: false
22 lp_repos: []
23 lp_repos_keys: []
24
25 lp_packages_auto: false
26 lp_packages_debug: false
27 lp_packages_install: []
28 lp_packages_remove: []
29 lp_packages_selections_preinstall: []
30 lp_packages_selections_postinstall: []
31 lp_packages_autoremove: false
32
33 lp_package_state: present           # apt and yum support "latest"
34 lp_package_state_remove: absent     # see "remove" vs. "purge"
35 lp_package_install_dryrun: false
36
37 lp_modules_debug: false
38 lp_modules: []
39 lp_modules_blacklist: []
40 lp_modules_blacklist_path: /etc/modprobe.d
41 lp_modules_options: []
42 lp_modules_options_path: /etc/modprobe.d
43
44 lp_sysctl_debug: false
45 lp_sysctl_vars: []
46
47 lp_udev: true
48 lp_udev_debug: false
49 lp_udev_enable: true
50 lp_udev_rules_dir: /etc/udev/rules.d
51 lp_udev_rules_template: udev-rules.j2
52 lp_udev_rules: {}
53 lp_udev_persistent_net_template: persistent-net.rules.j2
54 lp_udev_persistent_net_rules_file: 70-persistent-net.rules
55 lp_udev_persistent_net_rules: []
56 lp_udev_hci_name_rules_file: 71-hci-name.rules
57 lp_udev_hci_name_rules: []
58 lp_udev_hci_run_rules_file: 72-hci-run.rules
59 lp_udev_hci_run_rules: []
60
61 lp_wpagui: false
62 lp_wpagui_debug: false
63 lp_wpagui_nm_override: manual
64
65 lp_iptables: false
66 lp_iptables_type: default
67 lp_iptables_wan_if: eth0
68 lp_iptables_lan_if: eth1
69 lp_iptables_lan: 10.1.0.0/24
70 lp_iptables_INPUT_if: []
71 lp_iptables_INPUT_net: []
72

```

(continues on next page)

(continued from previous page)

```

73 lp_users_debug: false
74 lp_users: []
75 lp_users_groups: []
76
77 lp_passwords: false
78 lp_passwords_debug: false
79 lp_passwords_debug_classified: false
80 lp_passwords_fail_gracefully: false
81 lp_password_update_password: always
82 lp_passwordstore: false
83 lp_passwordstore_debug: false
84 lp_passwordstore_install: true
85 lp_passwordstore_backup: false
86 lp_passwordstore_create: false
87 lp_passwordstore_length: 16
88 lp_passwordstore_nosymbols: false
89 lp_passwordstore_overwrite: false
90 lp_passwordstore_passwordstore: ~/.password-store
91 lp_passwordstore_returnall: false
92 lp_passwordstore_subkey: password
93 lp_passwordstore_idempotent_password_hash: true
94
95 lp_aliases: false
96 lp_aliases_config: []
97
98 lp_authorized_key: []
99
100 lp_hosts_debug: false
101 lp_hosts: []
102
103 lp_grub: true
104 lp_grub_debug: false
105 lp_grub_default: []
106
107 lp_kvm: false
108 lp_kvm_debug: false
109
110 lp_xen: false
111 lp_xen_debug: false
112 lp_xen_dom0_mem: 512M
113 lp_xen_dom0_mem_max: 512M
114 lp_xen_XEN_OVERRIDE_GRUB_DEFAULT: 0
115 lp_xen_default_grub_conf:
116   - key: GRUB_CMDLINE_XEN_DEFAULT
117     value: "\"dom0_mem={{ lp_xen_dom0_mem }}\", max:{{ lp_xen_dom0_mem_max }}\"\"
118   - key: XEN_OVERRIDE_GRUB_DEFAULT
119     value: "{{ lp_xen_XEN_OVERRIDE_GRUB_DEFAULT }}"
120 lp_xen_global: []
121
122 lp_latex: false
123 lp_latex_download_timeout: 20
124 lp_latex_macros: []
125 lp_latex_get_url_ignore_errors: false
126
127 lp_auto_upgrades: false
128 lp_auto_upgrades_enable: false
129

```

(continues on next page)

(continued from previous page)

```

130 lp_auto_upgrades_Update_Package_Lists: 0
131 lp_auto_upgrades_Unattended_Upgrade: 0
132
133 lp_pm: false
134 lp_pm_sleepd: {}
135
136 lp_ssh: false
137 lp_ssh_debug: false
138 lp_ssh_config: []
139
140 lp_sshd: false
141 lp_sshd_debug: false
142 lp_sshd_enable: false
143 lp_sshd_config: []
144
145 lp_bluetooth: false
146 lp_bluetooth_debug: false
147 lp_bluetooth_enable: false
148 lp_bluetooth_main_conf: []
149
150 lp_xorg_debug: false
151 lp_xorg_conf_dir: /usr/share/X11/xorg.conf.d
152 lp_xorg_conf: []
153
154 lp_cron_tab: []
155 lp_cron_var: []
156
157 lp_modemmanager: true
158 lp_modemmanager_enable: true
159 lp_modemmanager_override: ""
160
161 lp_gpsd: false
162 lp_gpsd_enable: false
163 lp_gpsd_START_DAEMON: "true"
164 lp_gpsd_USBAUTO: "true"
165 lp_gpsd_DEVICES: /dev/rfcomm0
166 lp_gpsd_GPSD_OPTIONS: -b
167 lp_gpsd_bt_rfcomm: []
168
169 lp_postfix: true
170 lp_postfix_debug: false
171 lp_postfix_enable: true
172 lp_postfix_main_conf: []
173
174 lp_smart: false
175 lp_smart_debug: false
176 lp_smart_enable: false
177 lp_smart_packages:
178     - smartmontools
179 lp_smart_service: smartmontools
180 lp_smart_devicescan: false
181 lp_smart_conf_file: /etc/smartd.conf
182 lp_smart_conf_owner: root
183 lp_smart_conf_group: root
184 lp_smart_conf_mode: "0644"
185 lp_smart_devices: []
186

```

(continues on next page)

(continued from previous page)

```

187 lp_virtualbox: false
188 lp_virtualbox_debug: false
189 lp_virtualbox_enable: false
190 lp_virtualbox_ignore_errors: false
191 lp_virtualbox_version: 5.2
192 lp_virtualbox_keys:
193   - https://www.virtualbox.org/download/oracle_vbox_2016.asc
194   - https://www.virtualbox.org/download/oracle_vbox.asc
195 lp_virtualbox_packages:
196   - "virtualbox-{{ lp_virtualbox_version }}"
197
198 lp_zeitgeist: true
199
200 lp_lid: false
201 lp_lid_logind_conf: /etc/systemd/logind.conf
202 lp_lid_logind_conf_vars: []
203 lp_lid_upower_conf: /etc/UPower/UPower.conf
204 lp_lid_upower_conf_vars: []
205
206 lp_acpi: false
207 lp_acpi_dir: /etc/acpi
208 lp_acpi_owner: root
209 lp_acpi_group: root
210 lp_acpi_event_mode: "0644"
211 lp_acpi_action_mode: "0755"
212 lp_acpi_events: {}
213 lp_acpi_actions: {}
214
215 lp_speechd: true
216 lp_speechd_debug: false
217 lp_speechd_enable: true
218
219 lp_sudoers_debug: false
220 lp_sudoers_owner: root
221 lp_sudoers_group: root
222 lp_sudoers_mode: "0440"
223 lp_sudoers_conf:
224   - {line: "#includedir /etc/sudoers.d", state: "present"}
225 lp_sudoers_01: []
226
227 lp_nfsd: false
228 lp_nfsd_enable: false
229 lp_nfsd_exports: []
230
231 lp_netplan: false
232 lp_netplan_root: /etc/netplan
233 lp_netplan_owner: root
234 lp_netplan_group: root
235 lp_netplan_version: 2
236 lp_netplan_renderer: NetworkManager
237 lp_netplan_conf: []
238
239 lp_apparmor: true
240 lp_apparmor_disable: []
241 lp_apparmor_complain: []
242 lp_apparmor_enforce: []
243

```

(continues on next page)

(continued from previous page)

```

244 lp_swap: false
245 lp_swap_debug: false
246 lp_swap_enable: false
247
248 lp_timezone: false
249 lp_timezone_debug: false
250 lp_timezone_zoneinfo: UTC
251
252 lp_timesyncd: true
253 lp_timesyncd_debug: false
254 lp_timesyncd_enable: true
255 lp_timesyncd_NTP: ""
256 lp_timesyncd_FallbackNTP: ntp.ubuntu.com
257 lp_timesyncd_RootDistanceMaxSec: 5
258 lp_timesyncd_PollIntervalMinSec: 32
259 lp_timesyncd_PollIntervalMaxSec: 2048
260
261 lp_gpg: false
262 lp_gpg_debug: false
263 lp_gpg_install: true
264 lp_gpg_packages_extra: []
265 lp_gpg_conf_default: []
266 lp_gpg_conf: []
267 lp_gpg_agent_conf_default: []
268 lp_gpg_agent_conf: []
269 lp_gpg_dirmngr_conf_default: []
270 lp_gpg_dirmngr_conf: []
271
272 lp_wpasupplicant: true
273 lp_wpasupplicant_debug: false
274 lp_wpasupplicant_debug_classified: false
275 lp_wpasupplicant_conf_only: false
276 lp_wpasupplicant_conf_owner: root
277 lp_wpasupplicant_conf_group: root
278 lp_wpasupplicant_conf_mode: "0600"
279 lp_wpasupplicant_conf_dir: /etc/wpa_supplicant
280 lp_wpasupplicant_conf_file: wpa_supplicant.conf
281 lp_wpasupplicant_conf_global:
282   - {key: ctrl_interface, value: "{{ lp_wpasupplicant_conf_ctrl_interface }}"}}
283 lp_wpasupplicant_conf: []
284 lp_wpa_action_script: false
285 lp_wpa_action_script_dir: /root/bin
286 lp_wpa_action_script_file: "{{ lp_wpa_action_script_dir }}/wpa_action.sh"
287 lp_wpa_action_script_owner: root
288 lp_wpa_action_script_group: root
289 lp_wpa_action_script_mode: "0770"
290 lp_wpa_action_script_dhclient: "{{ lp_dhclient }}"
291 lp_wpa_action_script_pidfile: /var/run/dhclient.${ifname}.pid
292 lp_wpa_action_script_options_connect: "-4 -nw -pf $pidfile -v"
293 lp_wpa_action_script_options_disconnect: "-4 -r -pf $pidfile -v"
294 lp_wpa_action_script_logfile: "/tmp/wpa_action.${ifname}"
295
296 lp_logrotate_conf_file: /etc/logrotate.conf
297 lp_logrotate_conf_dir: /etc/logrotate.d
298 lp_logrotate_conf_lines:
299   - {line: "include /etc/logrotate.d", state: "present"}
300 lp_logrotate_conf_blocks: []

```

(continues on next page)

(continued from previous page)

```

301 lp_logrotate_conf: []
302
303 lp_tlp: false
304 lp_tlp_debug: false
305 lp_tlp_enable: false
306 lp_tlp_thinkpad: false
307 lp_tlp_config: []
308
309 lp_autofs: false
310 lp_autofs_enable: false
311 lp_autofs_conf_file: /etc/autofs.conf
312 lp_autofs_conf: []
313 lp_autofs_master_conf_file: /etc/auto.master
314 lp_autofs_master_conf: []
315 lp_autofs_misc_conf_file: /etc/auto.misc
316 lp_autofs_misc_conf: []
317
318 lp_libvirt: false
319 lp_libvirt_debug: false
320 lp_libvirt_guests_enable: false
321 lp_libvirt_libvirtd_enable: false
322 lp_libvirt_conf: {}
323
324 lp_zfs: false
325 lp_zfs_install: true
326 lp_zfs_debug: false
327 lp_zfs_manage: []
328 lp_zfs_mountpoints: []
329
330 lp_service_debug: false
331 lp_service: []
332 lp_service_enable:
333     - udev
334     - auto_upgrades
335     - sshd
336     - bluetooth
337     - gpsd
338     - postfix
339     - smart
340     - speechd
341     - timesyncd
342     - autofs
343     - libvirt_libvirtd
344     - libvirt_guests
345
346 lp_ufw: true
347 lp_ufw_enable: true
348 lp_ufw_debug: false
349 lp_ufw_reset: false
350 lp_ufw_reload: false
351 lp_ufw_service: ufw
352 lp_ufw_conf:
353     - {state: enabled, policy: allow}
354     - {logging: "on"}
355
356 lp_debsums: false
357 lp_debsums_debug: false

```

(continues on next page)

(continued from previous page)

```

358 lp_debsums_default_file: /etc/default/debsums
359 lp_debsums_default_conf:
360   - {key: CRON_CHECK, value: never}
361 lp_debsums_ignore_file: /etc/debsums-ignore
362 lp_debsums_ignore_conf: []
363
364 lp_fstab_entries: []
365
366 lp_resolvconf: false
367 lp_resolvconf_enable: false
368 lp_resolvconf_debug: false
369 lp_resolvconf_confd_head: []
370
371 lp_reboot: false
372 lp_reboot_debug: false
373 lp_reboot_force: false
374 lp_reboot_required_ignore: true
375 lp_reboot_required_file: /var/run/reboot-required
376 lp_reboot_required_command: /sbin/needs-restarting -r
377 lp_reboot_command: "sleep 5 && shutdown -r now"
378 lp_reboot_wait_connect_timeout: 20
379 lp_reboot_wait_sleep: 5
380 lp_reboot_wait_delay: 5
381 lp_reboot_wait_timeout: 300
382
383 # Include default vars for various flavors. For example put vars into
384 # one of the files below. First found will be included.
385 #
386 #   vars/flavors/armbian-<VERSION>-<BOARD>.yaml
387 #   vars/flavors/armbian-<VERSION>.yaml
388 #   vars/flavors/armbian.yaml
389 #   vars/defaults.yaml
390 #
391 # 1) File with service tasks task/sub/vars-flavors-<flavor>.yaml is
392 #    needed when new flavor is added to lp_flavors. See
393 #    tasks/sub/vars-flavors-common.yaml
394 # 2) For precedence of vars see tasks/vars.yaml
395
396 lp_flavors_enable: true
397 lp_flavors_dir: "{ inventory_dir ~ '/flavors' }"
398 lp_flavors_dir_owner: admin
399 lp_flavors_dir_group: adm
400 lp_flavors_dir_mode: "0775"
401 lp_flavors:
402   lsb:
403     release_file: /etc/lsb-release
404     file_labels: [DISTRIB_ID, DISTRIB_CODENAME]
405   os:
406     release_file: /etc/os-release
407     file_labels: [ID, UBUNTU_CODENAME]
408   armbian:
409     release_file: /etc/armbian-release
410     file_labels: [VERSION, BOARD]
411
412 # userland paths
413 lp_dhclient: /sbin/dhclient
414

```

(continues on next page)

(continued from previous page)

```

415 # TODO:
416 # * lp_virtualbox_services lp_virtualbox_enable
417 # * lp_tlp_services lp_tlp_enable
418 # * lp_nfsd_services lp_nfsd_enable
419
420 # EOF
421 ...

```

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>

ANNOTATED SOURCE CODE

Table of Contents

- *Annotated source code*
 - *Tasks*
 - * *main.yml*
 - * *acpi.yml*
 - * *aliases.yml*
 - * *apparmor.yml*
 - * *authorized_keys.yml*
 - * *autofs.yml*
 - * *auto_upgrades.yml*
 - * *bluetooth.yml*
 - * *cron.yml*
 - * *debsums.yml*
 - * *debug.yml*
 - * *fstab.yml*
 - * *gpg.yml*
 - * *gpsd.yml*
 - * *grub.yml*
 - * *hostname.yml*
 - * *hosts.yml*
 - * *iptables.yml*
 - * *kvm.yml*
 - * *latex.yml*
 - * *libvirt-conf.yml*
 - * *libvirt.yml*
 - * *lid.yml*

- * *logrotate.yml*
- * *modemmanager.yml*
- * *modules.yml*
- * *netplan.yml*
- * *nfsd.yml*
- * *packages-auto.yml*
- * *packages.yml*
- * *passwords.yml*
- * *pm-utils.yml*
- * *postfix.yml*
- * *reboot.yml*
- * *repos.yml*
- * *resolvconf.yml*
- * *service.yml*
- * *smart.yml*
- * *speechd.yml*
- * *sshd.yml*
- * *ssh.yml*
- * *sudoers.yml*
- * *swap.yml*
- * *sysctl.yml*
- * *timesyncd.yml*
- * *timezone.yml*
- * *tlp.yml*
- * *udev.yml*
- * *ufw.yml*
- * *users.yml*
- * *vars.yml*
- * *vars-firstfound.yml*
- * *vars-incremental.yml*
- * *virtualbox.yml*
- * *wpagui.yml*
- * *wpasupplicant.yml*
- * *xen.yml*
- * *xorg.yml*


```

* zeitgeist.yml
* zfs.yml
* sub/vars-flavors.yml
* sub/vars-flavors-common.yml

```

3.1 Tasks

3.1.1 main.yml

Synopsis: Tasks of the playbook.

Description of the task.

[main.yml]

```

1  ---
2  # tasks linux-postinstall
3
4  - import_tasks: vars.yml
5    tags: [lp_vars, always]
6
7  - import_tasks: debug.yml
8    when: lp_debug|bool
9    tags: [lp_debug, always]
10
11 - import_tasks: swap.yml
12   when: ((ansible_os_family == "RedHat") or
13         (ansible_os_family == "Debian")) and lp_swap|bool
14   tags: lp_swap
15
16 - import_tasks: modules.yml
17   when: (ansible_os_family == "RedHat") or
18         (ansible_os_family == "Debian")
19   tags: lp_modules
20
21 - import_tasks: udev.yml
22   when: ((ansible_os_family == "RedHat") or
23         (ansible_os_family == "Debian")) and lp_udev|bool
24   tags: lp_udev
25
26 - import_tasks: fstab.yml
27   when: ((ansible_os_family == "RedHat") or
28         (ansible_os_family == "Debian"))
29   tags: lp_fstab
30
31 - import_tasks: netplan.yml
32   when: (ansible_os_family == "Debian") and lp_netplan|bool
33   tags: lp_netplan
34
35 - import_tasks: timezone.yml
36   when: ((ansible_os_family == "RedHat") or
37         (ansible_os_family == "Debian")) and lp_timezone|bool
38   tags: lp_timezone

```

(continues on next page)

(continued from previous page)

```
39 - import_tasks: timesyncd.yml
40   when: (ansible_os_family == "Debian") and lp_timesyncd|bool
41   tags: lp_timesyncd
42
43
44 - import_tasks: repos.yml
45   when: ansible_os_family == "Debian"
46   tags: lp_repos
47
48 - import_tasks: packages.yml
49   when: (ansible_os_family == "RedHat") or
50         (ansible_os_family == "Debian")
51   tags: lp_packages
52
53 - import_tasks: auto_upgrades.yml
54   when: (ansible_os_family == "Debian") and lp_auto_upgrades|bool
55   tags: lp_auto_upgrades
56
57 - import_tasks: sysctl.yml
58   when: (ansible_os_family == "RedHat") or
59         (ansible_os_family == "Debian")
60   tags: lp_sysctl
61
62 - import_tasks: zfs.yml
63   when: (ansible_os_family == "Debian") and lp_zfs|bool
64   tags: lp_zfs
65
66 - import_tasks: hostname.yml
67   when: (ansible_os_family == "RedHat") or
68         (ansible_os_family == "Debian")
69   tags: lp_hostname
70
71 - import_tasks: hosts.yml
72   when: (ansible_os_family == "RedHat") or
73         (ansible_os_family == "Debian")
74   tags: lp_hosts
75
76 - import_tasks: iptables.yml
77   when: (ansible_os_family == "Debian") and lp_iptables|bool
78   tags: lp_iptables
79
80 - import_tasks: grub.yml
81   when: (ansible_os_family == "Debian") and lp_grub|bool
82   tags: lp_grub
83   # https://unix.stackexchange.com/questions/152222/
84   # equivalent-of-update-grub-for-rhel-fedora-centos-systems
85
86 - import_tasks: users.yml
87   when: (ansible_os_family == "RedHat" ) or
88         (ansible_os_family == "Debian" )
89   tags: lp_users
90
91 - import_tasks: gpg.yml
92   when: (ansible_os_family == "Debian") and lp_gpg|bool
93   tags: lp_gpg
94
95 - import_tasks: passwords.yml
```

(continues on next page)

(continued from previous page)

```

96  when: (ansible_os_family == "RedHat" ) or
97         (ansible_os_family == "Debian" ) and lp_passwords|bool
98  tags: lp_passwords
99
100 - import_tasks: sudoers.yml
101   when: (ansible_os_family == "RedHat" ) or
102          (ansible_os_family == "Debian" )
103   tags: lp_sudoers
104
105 - import_tasks: authorized_keys.yml
106   when: (ansible_os_family == "RedHat" ) or
107          (ansible_os_family == "Debian" )
108   tags: lp_authorized_keys
109
110 - import_tasks: aliases.yml
111   when: ((ansible_os_family == "RedHat" ) or
112          (ansible_os_family == "Debian" )) and lp_aliases|bool
113   tags: lp_aliases
114
115 - import_tasks: pm-utils.yml
116   when: (ansible_os_family == "Debian") and lp_pm|bool
117   tags: lp_pm
118
119 - import_tasks: ssh.yml
120   when: ((ansible_os_family == "RedHat" ) or
121          (ansible_os_family == "Debian" )) and lp_ssh|bool
122   tags: lp_ssh
123
124 - import_tasks: sshd.yml
125   when: ((ansible_os_family == "RedHat" ) or
126          (ansible_os_family == "Debian" )) and lp_sshd|bool
127   tags: lp_sshd
128
129 - import_tasks: bluetooth.yml
130   when: (ansible_os_family == "Debian") and lp_bluetooth|bool
131   tags: lp_bluetooth
132
133 - import_tasks: xorg.yml
134   when: ansible_os_family == "Debian"
135   tags: lp_xorg
136
137 - import_tasks: cron.yml
138   when: (ansible_os_family == "RedHat" ) or
139          (ansible_os_family == "Debian" )
140   tags: lp_cron
141
142 - import_tasks: modemmanager.yml
143   when: (ansible_os_family == "Debian") and lp_modemmanager|bool
144   tags: lp_modemmanager
145
146 - import_tasks: gpsd.yml
147   when: (ansible_os_family == "Debian") and lp_gpsd|bool
148   tags: lp_gpsd
149
150 - import_tasks: postfix.yml
151   when: ((ansible_os_family == "RedHat" ) or
152          (ansible_os_family == "Debian" )) and lp_postfix|bool

```

(continues on next page)

(continued from previous page)

```
153 tags: lp_postfix
154
155 - import_tasks: smart.yml
156   when: (ansible_os_family == "Debian") and lp_smart|bool
157   tags: lp_smart
158
159 - import_tasks: apparmor.yml
160   when: (ansible_os_family == "Debian") and lp_apparmor|bool
161   tags: lp_apparmor
162
163 - meta: flush_handlers
164
165 - import_tasks: zeitgeist.yml
166   when: ansible_os_family == "Debian"
167   tags: lp_zeitgeist
168
169 - import_tasks: lid.yml
170   when: (ansible_os_family == "Debian") and lp_lid|bool
171   tags: lp_lid
172
173 - import_tasks: acpi.yml
174   when: (ansible_os_family == "Debian") and lp_acpi|bool
175   tags: lp_acpi
176
177 - import_tasks: speechd.yml
178   when: (ansible_os_family == "Debian") and lp_speechd|bool
179   tags: lp_speechd
180
181 - import_tasks: nfsd.yml
182   when: (ansible_os_family == "Debian") and lp_nfsd|bool
183   tags: lp_nfsd
184
185 - meta: flush_handlers
186
187 - import_tasks: latex.yml
188   when: (ansible_os_family == "Debian") and lp_latex|bool
189   tags: lp_latex
190
191 - import_tasks: kvm.yml
192   when: (ansible_os_family == "Debian") and lp_kvm|bool
193   tags: lp_kvm
194
195 - import_tasks: xen.yml
196   when: (ansible_os_family == "Debian") and lp_xen|bool
197   tags: lp_xen
198
199 - import_tasks: virtualbox.yml
200   when: (ansible_os_family == "Debian") and lp_virtualbox|bool
201   tags: lp_virtualbox
202
203 - import_tasks: wpagui.yml
204   when: (ansible_os_family == "Debian") and lp_wpagui|bool
205   tags: lp_wpagui
206
207 - import_tasks: wpasupplicant.yml
208   when: ((ansible_os_family == "RedHat" ) or
209         (ansible_os_family == "Debian" )) and lp_wpasupplicant|bool
```

(continues on next page)

(continued from previous page)

```

210     tags: lp_wpasupplicant
211
212 - import_tasks: logrotate.yml
213   when: (ansible_os_family == "RedHat" ) or
214         (ansible_os_family == "Debian" )
215   tags: lp_logrotate
216
217 - import_tasks: tlp.yml
218   when: (ansible_os_family == "Debian") and lp_tlp|bool
219   tags: lp_tlp
220
221 - import_tasks: autofs.yml
222   when: (ansible_os_family == "Debian") and lp_autofs|bool
223   tags: lp_autofs
224
225 - import_tasks: libvirt.yml
226   when: (ansible_os_family == "Debian") and lp_libvirt|bool
227   tags: lp_libvirt
228
229 - import_tasks: ufw.yml
230   when: (ansible_os_family == "Debian") and lp_ufw|bool
231   tags: lp_ufw
232
233 - import_tasks: debsums.yml
234   when: (ansible_os_family == "Debian") and lp_debsums|bool
235   tags: lp_debsums
236
237 - meta: flush_handlers
238
239 - import_tasks: service.yml
240   tags: lp_service
241
242 - import_tasks: resolvconf.yml
243   when: (ansible_os_family == "Debian") and lp_resolvconf|bool
244   tags: lp_resolvconf
245
246 - import_tasks: reboot.yml
247   when: ((ansible_os_family == "RedHat" ) or
248         (ansible_os_family == "Debian")) and lp_reboot|bool
249   tags: lp_reboot
250
251 # EOF
252 ...

```

3.1.2 acpi.yml

Synopsis: Configure acpi.

Description of the task.

[acpi.yml]

```

1 ---
2 # linux-postinstall acpi
3
4 - name: "acpi: Configure {{ lp_acpi_dir }}/events"

```

(continues on next page)

(continued from previous page)

```

5  template:
6      src: "{{ item.value.template }}"
7      dest: "{{ lp_acpi_dir }}/events/{{ item.value.file }}"
8      owner: "{{ lp_acpi_owner }}"
9      group: "{{ lp_acpi_group }}"
10     mode: "{{ lp_acpi_event_mode }}"
11     backup: "{{ lp_backup_conf }}"
12     loop: "{{ lp_acpi_events|dict2items }}"
13     tags: lp_acpi_events
14
15 - name: "acpi: Create actions in {{ lp_acpi_dir }}"
16     template:
17         src: "{{ item.value.template }}"
18         dest: "{{ lp_acpi_dir }}/{{ item.value.file }}"
19         owner: "{{ lp_acpi_owner }}"
20         group: "{{ lp_acpi_group }}"
21         mode: "{{ lp_acpi_action_mode }}"
22         backup: "{{ lp_backup_conf }}"
23         loop: "{{ lp_acpi_actions|dict2items }}"
24         tags: lp_acpi_actions
25
26 # EOF
27 ...

```

3.1.3 aliases.yml

Synopsis: Configure aliases.

Description of the task.

[aliases.yml]

```

1  ---
2  # linux-postinstall aliases
3
4  - name: "aliases: Configure /etc/aliases"
5      template:
6          src: aliases.j2
7          dest: /etc/aliases
8          owner: root
9          group: root
10         mode: "0644"
11         backup: "{{ lp_backup_conf }}"
12         notify: newaliases
13
14 # EOF
15 ...

```

3.1.4 apparmor.yml

Synopsis: Configure apparmor.

Description of the task.

[apparmor.yml]

```

1 ---
2 # linux-postinstall apparmor
3
4 - name: "apparmor: Install packages"
5   include_tasks: fn/install-package.yml
6   loop: "{{ lp_apparmor_packages }}"
7   tags: lp_apparmor_packages
8
9 - name: "apparmor: Create list of profiles"
10  block:
11    - name: "apparmor: List profiles"
12      shell: "aa-status --json | jq .profiles | jq to_entries"
13      register: result
14      changed_when: false
15    - name: "apparmor: Create list of enforced profiles"
16      set_fact:
17        lp_apparmor_profiles_enforce: "{{ lp_apparmor_profiles_enforce|default([]) +
18      ↪ [ item.key ] }}"
19        loop: "{{ result.stdout|default([]) }}"
20        when: item.value == 'enforce'
21    - name: "apparmor: Create list of complained profiles"
22      set_fact:
23        lp_apparmor_profiles_complain: "{{ lp_apparmor_profiles_complain|default([])
24      ↪ + [ item.key ] }}"
25        loop: "{{ result.stdout|default([]) }}"
26        when: item.value == 'complain'
27    - name: "apparmor: Debug: List enforced profiles"
28      debug:
29        var: lp_apparmor_profiles_enforce
30        when: lp_debug
31    - name: "apparmor: Debug: List complained profiles"
32      debug:
33        var: lp_apparmor_profiles_complain
34        when: lp_debug
35  tags: lp_apparmor_profiles
36
37 - name: "apparmor: Disable profiles"
38   command: aa-disable "{{ item }}"
39   loop: "{{ lp_apparmor_disable }}"
40   when: item in lp_apparmor_profiles_enforce|default([]) or
41         item in lp_apparmor_profiles_complain|default([])
42   tags: lp_apparmor_disable
43
44 - name: "apparmor: Enforce profiles"
45   command: aa-enforce "{{ item }}"
46   loop: "{{ lp_apparmor_enforce }}"
47   when: item not in lp_apparmor_profiles_enforce|default([])
48   tags: lp_apparmor_enforce
49
50 - name: "apparmor: Complain profiles"
51   command: aa-complain "{{ item }}"
52   loop: "{{ lp_apparmor_complain }}"
53   when: item not in lp_apparmor_profiles_complain|default([])
54   tags: lp_apparmor_enforce
55
56 - name: "apparmor: Start and enable apparmor"
57   service:

```

(continues on next page)

(continued from previous page)

```

56     name: apparmor
57     state: started
58     enabled: true
59     when: lp_apparmor|bool
60     tags: lp_apparmor_service
61
62 - name: "apparmor: Stop and disable apparmor"
63   service:
64     name: apparmor
65     state: stopped
66     enabled: false
67     when: not lp_apparmor|bool
68     tags: lp_apparmor_service
69
70 # EOF
71 ...

```

3.1.5 authorized_keys.yml

Synopsis: Configure authorized_keys.

Description of the task.

[authorized_keys.yml]

```

1 ---
2 # linux-postinstall authorized_keys
3
4 - name: "authorized_key: Configure authorized_keys"
5   authorized_key:
6     user: "{{ item.user }}"
7     key: "{{ item.key }}"
8     manage_dir: true
9     loop: "{{ lp_authorized_keys }}"
10
11 # EOF
12 ...

```

3.1.6 autofs.yml

Synopsis: Configure autofs.

Description of the task.

[autofs.yml]

```

1 ---
2 # linux-postinstall autofs
3
4 - name: "autofs: Debug"
5   debug:
6     msg: "lp_autofs_enable [ {{ lp_autofs_enable }} ]"
7     when: lp_debug|bool
8
9 - name: "autofs: Configure {{ lp_autofs_conf_file }}"

```

(continues on next page)

(continued from previous page)

```

10  lineinfile:
11      dest: "{{ lp_autofs_conf_file }}"
12      regexp: "^{{ item.key }}\\s*=\\s*(.*)$"
13      line: "{{ item.key }} = {{ item.value }}"
14      backup: "{{ lp_backup_conf }}"
15      loop: "{{ lp_autofs_conf }}"
16      notify: reload autofs
17
18  - name: "autofs: Configure {{ lp_autofs_master_conf_file }}"
19      lineinfile:
20          dest: "{{ lp_autofs_master_conf_file }}"
21          regexp: "^{{ item.key }}\\s*(.*)$"
22          line: "{{ item.key }} {{ item.value }}"
23          backup: "{{ lp_backup_conf }}"
24          loop: "{{ lp_autofs_master_conf }}"
25          notify: reload autofs
26
27  - name: "autofs: Configure {{ lp_autofs_misc_conf_file }}"
28      lineinfile:
29          dest: "{{ lp_autofs_misc_conf_file }}"
30          regexp: "^{{ item.key }}\\s*(.*)$"
31          line: "{{ item.key }} {{ item.value }}"
32          backup: "{{ lp_backup_conf }}"
33          loop: "{{ lp_autofs_misc_conf }}"
34          notify: reload autofs
35
36  - name: "autofs: Enable and start autofs"
37      systemd:
38          name: "{{ lp_autofs_service }}"
39          enabled: true
40          state: started
41          when: lp_autofs_enable|bool
42
43  - name: "autofs: Stop and disable autofs"
44      systemd:
45          name: "{{ lp_autofs_service }}"
46          enabled: false
47          state: stopped
48          when: not lp_autofs_enable|bool
49
50  # EOF
51  ...

```

3.1.7 auto_upgrades.yml

Synopsis: Configure auto_upgrades.

Description of the task.

[auto_upgrades.yml]

```

1  ---
2  # linux-postinstall auto_upgrades
3
4  - name: "auto_upgrades: Configure /etc/apt/apt.conf.d/20auto-upgrades"
5      template:

```

(continues on next page)

(continued from previous page)

```

6     src: auto-upgrades.j2
7     dest: /etc/apt/apt.conf.d/20auto-upgrades
8     owner: root
9     group: root
10    mode: "0644"
11    backup: "{{ lp_backup_conf }}"
12
13  - name: "auto_upgrades: Disable and stop unattended-upgrades"
14    systemd:
15      name: "{{ lp_auto_upgrades_service }}"
16      state: stopped
17      enabled: false
18      when: not lp_auto_upgrades_enable|bool
19
20  - name: "auto_upgrades: Enable and start unattended-upgrades"
21    systemd:
22      name: "{{ lp_auto_upgrades_service }}"
23      state: started
24      enabled: true
25      when: lp_auto_upgrades_enable|bool
26
27  # EOF
28  ...

```

3.1.8 bluetooth.yml

Synopsis: Configure bluetooth.

Description of the task.

[bluetooth.yml]

```

1  ---
2  # linux-postinstall bluetooth
3
4  - name: "bluetooth: Debug"
5    debug:
6      msg: "lp_bluetooth_enable [{{ lp_bluetooth_enable }}]"
7      when: lp_bluetooth_debug|bool
8      tags: lp_bluetooth_debug
9
10  - name: "bluetooth: Configure /etc/bluetooth/main.conf"
11    lineinfile:
12      dest: /etc/bluetooth/main.conf
13      regexp: "^{{ item.key }}\\s*=(.*)$"
14      insertbefore: "^{{ '#' }}{{ item.key }}\\s*=(.*)$"
15      line: "{{ item.key }} = {{ item.value }}"
16      backup: "{{ lp_backup_conf }}"
17      loop: "{{ lp_bluetooth_main_conf }}"
18      notify: restart bluetooth
19      tags: lp_bluetooth_conf
20
21  - name: "bluetooth: Enable and start bluetooth"
22    systemd:
23      name: "{{ lp_bluetooth_service }}"
24      enabled: true

```

(continues on next page)

(continued from previous page)

```

25     state: started
26 when: lp_bluetooth_enable|bool
27 tags: lp_bluetooth_enable
28
29 - name: "bluetooth: Stop and disable bluetooth"
30   systemd:
31     name: "{{ lp_bluetooth_service }}"
32     enabled: false
33     state: stopped
34 when: not lp_bluetooth_enable|bool
35 tags: lp_bluetooth_disable
36
37 # EOF
38 ...

```

3.1.9 cron.yml

Synopsis: Configure cron.

Description of the task.

[cron.yml]

```

1 ---
2 # linux-postinstall cron
3
4 - name: "cron: Configure cron variables"
5   cronvar:
6     name: "{{ item.name }}"
7     value: "{{ item.value }}"
8     user: "{{ item.user }}"
9     loop: "{{ lp_cron_var }}"
10    tags: lp_cron_var
11
12 - name: "cron: Configure cron"
13   cron:
14     state: "{{ item.state }}"
15     user: "{{ item.user }}"
16     name: "{{ item.name }}"
17     minute: "{{ item.minute }}"
18     hour: "{{ item.hour }}"
19     day: "{{ item.day }}"
20     month: "{{ item.month }}"
21     weekday: "{{ item.weekday }}"
22     job: "{{ item.command }}"
23     loop: "{{ lp_cron_tab }}"
24     tags: lp_cron_tab
25
26 # EOF
27 ...

```

3.1.10 debsums.yml

Synopsis: Configure debsums.

Description of the task.

[debsums.yml]

```

1  ---
2  # linux-postinstall debsums
3
4  - name: "debsums: Debug"
5    vars:
6      msg: |
7        lp_debsums_default_file [{{ lp_debsums_default_file }}]
8        lp_debsums_default_conf
9        {{ lp_debsums_default_conf|to_yaml }}
10       lp_debsums_ignore_file [{{ lp_debsums_ignore_file }}]
11       lp_debsums_ignore_conf
12       {{ lp_debsums_ignore_conf|to_nice_yaml }}
13    debug:
14      msg: "{{ msg.split('\n')[:-1] }}"
15    when: lp_debsums_debug|bool
16    tags: lp_debsums_debug
17
18  - name: "debsums: Install packages"
19    include_tasks: fn/install-package.yml
20    loop: "{{ lp_debsums_packages }}"
21    tags: lp_debsums_packages
22
23  - name: "debsums: Configure {{ lp_debsums_default_file }}"
24    lineinfile:
25      dest: "{{ lp_debsums_default_file }}"
26      state: "{{ item.state|default(omit) }}"
27      regexp: '^s*{{ item.key }}s*=(.*)$'
28      line: "{{ item.key }}={{ item.value }}"
29      backup: "{{ lp_backup_conf }}"
30      create: true
31    loop: "{{ lp_debsums_default_conf }}"
32    tags: lp_debsums_default_conf
33
34  - name: "debsums: Configure {{ lp_debsums_ignore_file }}"
35    lineinfile:
36      dest: "{{ lp_debsums_ignore_file }}"
37      state: "{{ item.state|default(omit) }}"
38      line: "{{ item }}"
39      backup: "{{ lp_backup_conf }}"
40      create: true
41    loop: "{{ lp_debsums_ignore_conf }}"
42    tags: lp_debsums_ignore_conf
43
44  # EOF
45  ...

```

3.1.11 debug.yml

Synopsis: Configure debug.

Description of the task.

[debug.yml]

```

1  ---
2  # Hint: Get readable output with stdout_callback = yaml
3
4  - name: "Debug"
5    vars:
6      msg: |
7        ansible_architecture [{{ ansible_architecture }}]
8        ansible_os_family [{{ ansible_os_family }}]
9        ansible_distribution [{{ ansible_distribution }}]
10       ansible_distribution_major_version [{{ ansible_distribution_major_version }}]
11       ansible_distribution_version [{{ ansible_distribution_version }}]
12       ansible_distribution_release [{{ ansible_distribution_release }}]
13       ansible_python_version [{{ ansible_python_version }}]
14
15       lp_vars_distro [{{ lp_vars_distro }}]
16       lp_vars_distro_firstfound_skip [{{ lp_vars_distro_firstfound_skip }}]
17       lp_vars_flavors [{{ lp_vars_flavors }}]
18       lp_flavors_enable [{{ lp_flavors_enable }}]
19       {{ my_release|default([])|to_nice_yaml }}
20
21       lp_backup_conf [{{ lp_backup_conf }}]
22
23       lp_acpi [{{ lp_acpi }}]
24       lp_aliases [{{ lp_aliases }}]
25       lp_apparmor [{{ lp_apparmor }}]
26       lp_auto_upgrades [{{ lp_auto_upgrades }}] lp_auto_upgrades_enable [{{ lp_auto_
↪ upgrades_enable }}]
27       lp_autofs [{{ lp_autofs }}] lp_autofs_enable [{{ lp_autofs_enable }}]
28       lp_bluetooth [{{ lp_bluetooth }}] lp_bluetooth_enable [{{ lp_bluetooth_enable }}]
↪ ]
29       lp_debsums [{{ lp_debsums }}]
30       lp_gpg [{{ lp_gpg }}]
31       lp_gpsd [{{ lp_gpsd }}] lp_gpsd_enable [{{ lp_gpsd_enable }}]
32       lp_grub [{{ lp_grub }}]
33       lp_iptables [{{ lp_iptables }}]
34       lp_kvm [{{ lp_kvm }}]
35       lp_latex [{{ lp_latex }}]
36       lp_libvirt [{{ lp_libvirt }}]
37       lp_libvirt_guests_enable [{{ lp_libvirt_guests_enable }}]
38       lp_libvirt_libvirtd_enable [{{ lp_libvirt_libvirtd_enable }}]
39       lp_lid [{{ lp_lid }}]
40       lp_modemmanager [{{ lp_modemmanager }}] lp_modemmanager_enable [{{ lp_
↪ modemmanager_enable }}]
41       lp_netplan [{{ lp_netplan }}]
42       lp_nfsd [{{ lp_nfsd }}] lp_nfsd_enable [{{ lp_nfsd_enable }}]
43       lp_packages_autoremove [{{ lp_packages_autoremove }}]
44       lp_passwords [{{ lp_passwords }}]
45       lp_pm [{{ lp_pm }}]
46       lp_postfix [{{ lp_postfix }}] lp_postfix_enable [{{ lp_postfix_enable }}]
47       lp_reboot [{{ lp_reboot }}]
48       lp_resolvconf [{{ lp_resolvconf }}]
49       lp_smart [{{ lp_smart }}] lp_smart_enable [{{ lp_smart_enable }}]
50       lp_speechd [{{ lp_speechd }}] lp_speechd_enable [{{ lp_speechd_enable }}]
51       lp_ssh [{{ lp_ssh }}]
52       lp_sshd [{{ lp_sshd }}] lp_sshd_enable [{{ lp_sshd_enable }}]
53       lp_swap [{{ lp_swap }}] lp_swap_enable [{{ lp_swap_enable }}]
54       lp_timesyncd [{{ lp_timesyncd }}] lp_timesyncd_enable [{{ lp_timesyncd_enable }}]
↪ ]

```

(continues on next page)

(continued from previous page)

```

55     lp_timezone [{{ lp_timezone }}]
56     lp_tlp [{{ lp_tlp }}] lp_tlp_enable [{{ lp_tlp_enable }}]
57     lp_ufw [{{ lp_ufw }}]
58     lp_virtualbox [{{ lp_virtualbox }}] lp_virtualbox_enable [{{ lp_virtualbox_
↪enable }}]
59     lp_wpagui [{{ lp_wpagui }}]
60     lp_wpasupplicant [{{ lp_wpasupplicant }}]
61     lp_xen [{{ lp_xen }}]
62     lp_zeitgeist [{{ lp_zeitgeist }}]
63     lp_zfs [{{ lp_zfs }}]
64
65     lp_service
66     {{ lp_service|to_yaml }}
67     lp_package_state [{{ lp_package_state }}]
68     lp_package_state_remove [{{ lp_package_state_remove }}]
69
70     debug:
71         msg: "{{ msg.split('\n')[:-1] }}"
72
73     # EOF
74     ...

```

3.1.12 fstab.yml

Synopsis: Configure fstab.

Description of the task.

[fstab.yml]

```

1  ---
2  # linux-postinstall fstab
3
4  - name: "fstab: Configure fstab entries"
5    mount:
6      name: "{{ item.name }}"
7      state: "{{ item.state|default('mounted') }}"
8      src: "{{ item.src|default(omit) }}"
9      fstype: "{{ item.fstype|default(omit) }}"
10     opts: "{{ item.opts|default(omit) }}"
11     dump: "{{ item.dump|default(omit) }}"
12     passno: "{{ item.passno|default(omit) }}"
13     backup: "{{ lp_backup_conf }}"
14     loop: "{{ lp_fstab_entries }}"
15
16     # EOF
17     ...

```

3.1.13 gpg.yml

Synopsis: Configure gpg.

Description of the task.

[gpg.yml]

```

1  ---
2  # linux-postinstall gpg
3
4  - name: "gpg: Debug"
5    vars:
6      msg: |
7        lp_gpg_install [{{ lp_gpg_install }}]
8        lp_gpg_packages
9        {{ lp_gpg_packages|to_nice_yaml }}
10       lp_gpg_packages_extra
11       {{ lp_gpg_packages_extra|to_nice_yaml }}
12
13       lp_gpg_conf_default
14       {{ lp_gpg_conf_default|to_yaml }}
15       lp_gpg_conf
16       {{ lp_gpg_conf|to_yaml }}
17
18       lp_gpg_agent_conf_default
19       {{ lp_gpg_agent_conf_default|to_yaml }}
20       lp_gpg_agent_conf
21       {{ lp_gpg_agent_conf|to_yaml }}
22
23       lp_gpg_dirmngr_conf_default
24       {{ lp_gpg_dirmngr_conf_default|to_yaml }}
25       lp_gpg_dirmngr_conf
26       {{ lp_gpg_dirmngr_conf|to_yaml }}
27     debug:
28       msg: "{{ msg.split('\n')[:-1] }}"
29     when: lp_gpg_debug|bool
30     tags: lp_gpg_debug
31
32  - name: "gpg: Install packages"
33    include_tasks: fn/install-package.yml
34    loop:
35      - "{{ lp_gpg_packages }}"
36      - "{{ lp_gpg_packages_extra }}"
37    when: lp_gpg_install|bool
38    tags: lp_gpg_packages
39
40  - name: "gpg: Configure gpg.conf"
41    template:
42      src: gpg.conf.j2
43      dest: "{{ item.dest|default('/home/' ~ item.owner ~ '/.gnupg/gpg.conf') }}"
44      owner: "{{ item.owner }}"
45      group: "{{ item.owner }}"
46      mode: "0600"
47      backup: "{{ lp_backup_conf }}"
48    loop: "{{ lp_gpg_conf }}"
49    loop_control:
50      label: "{{ item.owner }}"
51    tags: lp_gpg_conf
52
53  - name: "gpg: Configure gpg-agent.conf"
54    template:
55      src: gpg-agent.conf.j2
56      dest: "{{ item.dest|default('/home/' ~ item.owner ~ '/.gnupg/gpg-agent.conf') }}"
57      owner: "{{ item.owner }}"

```

(continues on next page)

(continued from previous page)

```

58     group: "{{ item.owner }}"
59     mode: "0600"
60     backup: "{{ lp_backup_conf }}"
61     loop: "{{ lp_gpg_agent_conf }}"
62     loop_control:
63       label: "{{ item.owner }}"
64     register: lp_gpg_agent_conf_changes
65     notify: gpgconf kill gpg-agent
66     tags: lp_gpg_agent_conf
67
68 - name: "gpg: Configure dirmngr.conf"
69   template:
70     src: gpg-dirmngr.conf.j2
71     dest: "{{ item.dest|default('/home/' ~ item.owner ~ '/.gnupg/dirmngr.conf') }}"
72     owner: "{{ item.owner }}"
73     group: "{{ item.owner }}"
74     mode: "0600"
75     backup: "{{ lp_backup_conf }}"
76     loop: "{{ lp_gpg_dirmngr_conf }}"
77     loop_control:
78       label: "{{ item.owner }}"
79     register: lp_gpg_dirmngr_conf_changes
80     notify: gpgconf kill dirmngr
81     tags: lp_gpg_dirmngr_conf
82
83 # TODO: import keys
84
85 # EOF
86 ...

```

3.1.14 gpsd.yml

Synopsis: Configure gpsd.

Description of the task.

[gpsd.yml]

```

1 ---
2 # linux-postinstall gpsd
3
4 - name: "gpsd: Install packages for gpsd"
5   include_tasks: fn/install-package.yml
6   loop: "{{ lp_gpsd_packages }}"
7   tags: lp_gpsd_packages
8
9 - name: "gpsd: Add user gpsd to group dialout"
10  user:
11    name: gpsd
12    groups: dialout
13    append: true
14    tags: lp_gpsd_group
15
16 - name: "gpsd: Configure /etc/bluetooth/rfcomm.conf"
17   blockinfile:
18     dest: /etc/bluetooth/rfcomm.conf

```

(continues on next page)

(continued from previous page)

```

19  marker: "# {mark} ANSIBLE MANAGED BLOCK rfcomm{{ item.rfcomm }}"
20  insertafter: EOF
21  owner: root
22  group: root
23  mode: "0644"
24  backup: "{{ lp_backup_conf }}"
25  block: |
26      rfcomm{{ item.rfcomm }} {
27          bind {{ item.bind }}
28          device {{ item.device }}
29          channel {{ item.channel }}
30          comment "{{ item.comment }}"
31      }
32  loop: "{{ lp_gpsd_bt_rfcomm }}"
33  notify: restart bluetooth
34  tags: lp_gpsd_bt_rfc
35
36 - name: "gpsd: Configure /etc/default/gpsd"
37   template:
38       src: gpsd.j2
39       dest: /etc/default/gpsd
40       owner: root
41       group: root
42       mode: "0644"
43       backup: "{{ lp_backup_conf }}"
44   notify: restart gpsd
45   tags: lp_gpsd_config
46
47 - name: "gpsd: Stop and disable gpsd"
48   systemd:
49       name: "{{ lp_gpsd_service }}"
50       state: stopped
51       enabled: false
52   when: not lp_gpsd_enable|bool
53   tags: lp_gpsd_service
54
55 - name: "gpsd: Start and enable gpsd"
56   systemd:
57       name: "{{ lp_gpsd_service }}"
58       state: started
59       enabled: true
60   when: lp_gpsd_enable|bool
61   tags: lp_gpsd_service
62
63 # EOF
64 ...

```

3.1.15 grub.yml

Synopsis: Configure grub.

Description of the task.

[grub.yml]

```

1 ---
2
3 - name: "grub: Debug"
4   vars:
5     msg: |
6       lp_grub_default
7       {{ lp_grub_default|to_yaml }}
8   debug:
9     msg: "{{ msg.split('\n')[:-1] }}"
10  when: lp_grub_debug|bool
11  tags: lp_grub_debug
12
13 - name: "grub: Configure /etc/default/grub"
14   lineinfile:
15     dest: /etc/default/grub
16     regexp: '^s*{{ item.var }}\s*=(.*)$'
17     line: "{{ item.var }}={{ item.value }}"
18     backup: "{{ lp_backup_conf|bool }}"
19     loop: "{{ lp_grub_default }}"
20     notify: update grub
21     tags: lp_grub_conf
22
23 # EOF
24 ...

```

3.1.16 hostname.yml

Synopsis: Configure hostname.

Description of the task.

[hostname.yml]

```

1 ---
2 # linux-postinstall hostname
3
4 # TODO:
5 # 1) SET/DONT_SET hostname via DHCP
6 # /etc/dhcp/dhclient.conf
7 # #send host-name = gethostname();
8 # request host-name = "myhostname";
9 # https://askubuntu.com/questions/104918/how-to-get-the-hostname-from-a-dhcp-server
10 # http://blog.schlomo.schapiro.org/2013/11/setting-hostname-from-dhcp-in-debian.html
11 # https://askubuntu.com/questions/757423/how-to-force-dhcp-client-to-allow-a-self-
12   ↳ defined-domain-name
13
14 - name: "hostname: Configure hostname in /etc/hostname"
15   template:
16     src: hostname.j2
17     dest: /etc/hostname
18     owner: root
19     group: root
20     mode: "0644"
21     backup: "{{ lp_backup_conf }}"
22   when:
23     - lp_hostname|length > 0

```

(continues on next page)

(continued from previous page)

```

23     - ansible_os_family == "Debian"
24     # notify: set hostname
25
26     - name: "hostname: Configure hostname"
27       hostname:
28         name: "{{ lp_hostname }}"
29         when: lp_hostname|length > 0
30
31     # EOF
32     ...

```

3.1.17 hosts.yml

Synopsis: Configure hosts.

Description of the task.

[hosts.yml]

```

1  ---
2  # linux-postinstall hosts
3
4  - name: "hosts: Debug"
5    vars:
6      msg: |
7        lp_hosts_default_override
8        {{ lp_hosts_default_override|default('UNDEFINED')|to_yaml }}
9        lp_hosts_default
10       {{ lp_hosts_default|to_yaml }}
11       lp_hosts
12       {{ lp_hosts|to_yaml }}
13    debug:
14      msg: "{{ msg.split('\n')[:-1] }}"
15    when: lp_hosts_debug|bool
16    tags: lp_hosts_debug
17
18  - name: "hosts: Configure hosts in /etc/hosts"
19    template:
20      src: hosts.j2
21      dest: /etc/hosts
22      owner: root
23      group: root
24      mode: "0644"
25      backup: "{{ lp_backup_conf }}"
26    tags: lp_hosts_conf
27
28  # EOF
29  ...

```

3.1.18 iptables.yml

Synopsis: Configure iptables.

Description of the task.

[iptables.yml]

```

1 ---
2 # linux-postinstall iptables
3
4 - name: "iptables: Create /etc/network/if-pre-up.d/iptables"
5   template:
6     src: iptables-restore.j2
7     dest: /etc/network/if-pre-up.d/iptables
8     owner: root
9     group: root
10    mode: "0755"
11
12 - name: "iptables: Create /etc/network/iptables
13     using {{ lp_iptables_type }}-iptables.j2"
14   template:
15     src: "{{ lp_iptables_type }}-iptables.j2"
16     dest: /etc/network/iptables
17     owner: root
18     group: root
19     mode: "0644"
20    notify: reload iptables
21
22 # EOF
23 ...

```

3.1.19 kvm.yml

Synopsis: Configure kvm.

Description of the task.

[kvm.yml]

```

1 ---
2 # linux-postinstall kvm
3
4 - name: "kvm: Debug"
5   vars:
6     msg: |
7       lp_kvm_packages
8       {{ lp_kvm_packages|to_nice_yaml }}
9   debug:
10    msg: "{{ msg.split('\n')[:-1] }}"
11  when: lp_kvm_debug|bool
12  tags: lp_kvm_debug
13
14 - name: "kvm: Install packages"
15   include_tasks: fn/install-package.yml
16   loop: "{{ lp_kvm_packages }}"
17   tags: lp_kvm_packages
18
19 # EOF
20 ...

```

3.1.20 latex.yml

Synopsis: Configure latex.

Description of the task.

[latex.yml]

```

1  ---
2  # linux-postinstall LaTeX
3
4  - name: "latex: Install packages"
5    include_tasks: fn/install-package.yml
6    loop: "{{ lp_latex_packages }}"
7    tags: lp_latex_packages
8
9  - name: "latex: Create directory /usr/share/texmf/tex/latex"
10    file:
11      state: directory
12      path: /usr/share/texmf/tex/latex
13    tags: lp_latex_dir
14
15  - name: "latex: Create directories for macros"
16    file:
17      state: directory
18      path: "{{ item.dest }}"
19    loop: "{{ lp_latex_macros }}"
20    tags: lp_latex_macros
21
22  - name: "latex: Download macros"
23    get_url:
24      url: "{{ item.url }}"
25      dest: "{{ item.dest }}"
26      timeout: "{{ lp_latex_download_timeout }}"
27    loop: "{{ lp_latex_macros }}"
28    ignore_errors: "{{ lp_latex_get_url_ignore_errors }}"
29    changed_when: false
30    tags: lp_latex_labels
31
32  # get_url: check mode reports changes with force enabled
33  # https://github.com/ansible/ansible/issues/25418
34
35  # TODO:
36  # 1) Compile and register labels.sty
37  # cd /usr/share/texmf/tex/latex/labels/
38  # latex labels.ins
39  # texhash /usr/share/texmf
40
41  # EOF
42  ...

```

3.1.21 libvirt-conf.yml

Synopsis: Configure libvirt-conf.

Description of the task.

[libvirt-conf.yml]

```

1  ---
2  # linux-postinstall libvirt-conf

```

(continues on next page)

(continued from previous page)

```

3
4 - name: "libvirt-conf: Debug"
5   debug:
6     msg: "{{ item }}"
7   when: lp_libvirt_debug|bool
8
9 - name: "libvirt-conf: Configure {{ lp_libvirt_conf_dir }}/{{ item.key }}"
10  lineinfile:
11    dest: "{{ lp_libvirt_conf_dir }}/{{ item.key }}"
12    regexp: '^{{ conf[0] }}(\s|=)(.*)$'
13    line: "{{ conf[0] }} = {{ conf[1] }}"
14    state: "{{ conf[2] | default('present') }}"
15    backup: "{{ lp_backup_conf }}"
16    create: true
17  loop: "{{ item.value }}"
18  loop_control:
19    loop_var: conf
20  notify:
21    - reload libvirtd
22    - reload libvirt_guests
23
24 # EOF
25 ...

```

3.1.22 libvirt.yml

Synopsis: Configure libvirt.

Description of the task.

[libvirt.yml]

```

1 ---
2 # linux-postinstall libvirt
3
4 - name: "libvirt Debug"
5   vars:
6     msg: |
7       lp_libvirt_guests_enable [{{ lp_libvirt_guests_enable }}]
8       lp_libvirt_libvirtd_enable [{{ lp_libvirt_libvirtd_enable }}]
9       lp_libvirt_conf_owner [{{ lp_libvirt_conf_owner }}]
10      lp_libvirt_conf_group [{{ lp_libvirt_conf_group }}]
11      lp_libvirt_conf_mode [{{ lp_libvirt_conf_mode }}]
12      lp_libvirt_conf_dir [{{ lp_libvirt_conf_dir }}]
13      lp_libvirt_packages
14      {{ lp_libvirt_packages|to_nice_yaml }}
15      lp_libvirt_conf
16      {{ lp_libvirt_conf|to_yaml }}
17   debug:
18     msg: "{{ msg.split('\n')[:-1] }}"
19   when: lp_libvirt_debug|bool
20   tags: lp_libvirt_debug
21
22 - name: "libvirt: Install packages"
23   include_tasks: fn/install-package.yml
24   loop: "{{ lp_libvirt_packages }}"

```

(continues on next page)

(continued from previous page)

```

25 tags: lp_libvirt_pkg
26
27 - name: "libvirt: Configure"
28   include_tasks: libvirt-conf.yml
29   loop: "{{ lp_libvirt_conf|dict2items }}"
30   tags: lp_libvirt_conf
31
32 - name: "libvirt: Start and enable {{ lp_libvirt_libvirtd_service }}"
33   service:
34     name: "{{ lp_libvirt_libvirtd_service }}"
35     state: started
36     enabled: true
37   when: lp_libvirt_libvirtd_enable|bool
38   tags: lp_libvirt_libvirtd_service
39
40 - name: "libvirt: Stop and disable {{ lp_libvirt_libvirtd_service }}"
41   service:
42     name: "{{ lp_libvirt_libvirtd_service }}"
43     state: stopped
44     enabled: false
45   when: not lp_libvirt_libvirtd_enable|bool
46   tags: lp_libvirt_libvirtd_service
47
48 - name: "libvirt: Start and enable {{ lp_libvirt_guests_service }}"
49   service:
50     name: "{{ lp_libvirt_guests_service }}"
51     state: started
52     enabled: true
53   when: lp_libvirt_guests_enable|bool
54   tags: lp_libvirt_guests_service
55
56 - name: "libvirt: Stop and disable {{ lp_libvirt_guests_service }}"
57   service:
58     name: "{{ lp_libvirt_guests_service }}"
59     state: stopped
60     enabled: false
61   when: not lp_libvirt_guests_enable|bool
62   tags: lp_libvirt_guests_service
63
64 # EOF
65 ...

```

3.1.23 lid.yml

Synopsis: Configure lid.

Description of the task.

[lid.yml]

```

1 ---
2 # linux-postinstall lid
3
4 - name: "lid: Configure {{ lp_lid_logind_conf }}"
5   lineinfile:
6     dest: "{{ lp_lid_logind_conf }}"

```

(continues on next page)

(continued from previous page)

```

7     regexp: '^\\s*{{ item.var }}\\s*=\\s*(.*)$'
8     line: "{{ item.var }}={{ item.value }}"
9     backup: "{{ lp_backup_conf }}"
10    loop: "{{ lp_lid_logind_conf_vars }}"
11    notify: logind message reboot
12
13 - name: "lid: Configure {{ lp_lid_upower_conf }}"
14   lineinfile:
15     dest: "{{ lp_lid_upower_conf }}"
16     regexp: '^\\s*{{ item.var }}\\s*=\\s*(.*)$'
17     line: "{{ item.var }}={{ item.value }}"
18     backup: "{{ lp_backup_conf }}"
19     loop: "{{ lp_lid_upower_conf_vars }}"
20
21 # EOF
22 ...

```

3.1.24 logrotate.yml

Synopsis: Configure logrotate.

Description of the task.

[logrotate.yml]

```

1 ---
2 # linux-postinstall logrotate
3
4 - name: "logrotate: Install packages for logrotate"
5   include_tasks: fn/install-package.yml
6   loop: "{{ lp_logrotate_packages }}"
7
8 - name: "logrotate: Configure blocks in {{ lp_logrotate_conf_file }}"
9   blockinfile:
10    path: "{{ lp_logrotate_conf_file }}"
11    mark: "{{ item.mark }}"
12    block: "{{ item.block }}"
13    state: "{{ item.state }}"
14    backup: "{{ lp_backup_conf }}"
15    loop: "{{ lp_logrotate_conf_blocks }}"
16
17 - name: "logrotate: Configure lines in {{ lp_logrotate_conf_file }}"
18   lineinfile:
19    path: "{{ lp_logrotate_conf_file }}"
20    line: "{{ item.line }}"
21    state: "{{ item.state }}"
22    backup: "{{ lp_backup_conf }}"
23    loop: "{{ lp_logrotate_conf_lines }}"
24
25 - name: "logrotate: Configure {{ lp_logrotate_conf_dir }}"
26   blockinfile:
27    path: "{{ lp_logrotate_conf_dir }}/{{ item.path }}"
28    block: "{{ item.conf }}"
29    create: true
30    backup: "{{ lp_backup_conf }}"
31    loop: "{{ lp_logrotate_confd }}"

```

(continues on next page)

(continued from previous page)

```
32 # EOF
33 ...
34
```

3.1.25 modemmanager.yml

Synopsis: Configure modemmanager.

Description of the task.

[modemmanager.yml]

```
1 ---
2 # linux-postinstall ModemManager
3
4 - name: "modem_manager: Configure /etc/init/modemmanager.override"
5   template:
6     src: modem-manager-override.j2
7     dest: /etc/init/modemmanager.override
8     owner: root
9     group: root
10    mode: "0644"
11
12 - name: "modem_manager: Stop and disable ModemManager"
13   service:
14     name: ModemManager
15     state: stopped
16     enabled: false
17   when: not lp_modemmanager_enable|bool
18
19 - name: "modem_manager: Start and enable ModemManager"
20   service:
21     name: ModemManager
22     state: started
23     enabled: true
24   when: lp_modemmanager_enable|bool
25
26 # EOF
27 ...
```

3.1.26 modules.yml

Synopsis: Configure modules.

Description of the task.

[modules.yml]

```
1 ---
2 # linux-postinstall modules
3
4 - name: "modules: Debug"
5   vars:
6     msg: |
7       lp_modules_conf [{{ lp_modules_conf }}]
```

(continues on next page)

(continued from previous page)

```

8     lp_modules
9     {{ lp_modules|to_yaml }}
10    lp_modules_options_path [ {{ lp_modules_options_path }}]
11    lp_modules_options
12    {{ lp_modules_options|to_nice_yaml }}
13    lp_modules_blacklist_path [{{ lp_modules_blacklist_path }}]
14    lp_modules_blacklist
15    {{ lp_modules_blacklist|to_nice_yaml }}
16  debug:
17    msg: "{{ msg.split('\n')[:-1] }}"
18  when: lp_modules_debug|bool
19
20 - name: "modules: modprobe modules"
21  modprobe:
22    name: "{{ item.name }}"
23    params: "{{ item.params }}"
24    state: "{{ item.state|default('present') }}"
25    loop: "{{ lp_modules }}"
26
27  # Debian
28 - name: "modules: Configure {{ lp_modules_conf }} in Debian"
29  lineinfile:
30    dest: "{{ lp_modules_conf }}"
31    regexp: '^s*{{ item.name }}s*(.*)$'
32    line: "{{ item.name }} {{ item.params }}"
33    backup: "{{ lp_backup_conf }}"
34    loop: "{{ lp_modules }}"
35    when:
36      - ansible_os_family == "Debian"
37      - item.state|default("present") == "present"
38
39  # RedHat
40 - name: "modules: Configure {{ lp_modules_conf }} in RedHat"
41  lineinfile:
42    dest: "{{ lp_modules_conf }}"
43    regexp: '^s*modprobe\s+{{ item.name }}s*(.*)$'
44    line: "modprobe {{ item.name }} {{ item.params }}"
45    backup: "{{ lp_backup_conf }}"
46    loop: "{{ lp_modules }}"
47    when:
48      - ansible_os_family == "RedHat"
49      - item.state|default("present") == "present"
50
51 - name: "modules: Blacklist modules in {{ lp_modules_blacklist_path }}"
52  template:
53    src: blacklist-module.j2
54    dest: "{{ lp_modules_blacklist_path }}/blacklist-{{ item }}.conf"
55    backup: "{{ lp_backup_conf }}"
56    loop: "{{ lp_modules_blacklist }}"
57  notify: update initramfs
58
59 - name: "modules: Set modules options in {{ lp_modules_options_path }}"
60  template:
61    src: options-module.j2
62    dest: "{{ lp_modules_options_path }}/{{ item.module }}.conf"
63    backup: "{{ lp_backup_conf }}"
64    loop: "{{ lp_modules_options }}"

```

(continues on next page)

(continued from previous page)

```
65     notify: update initramfs
66
67     # EOF
68     ...
```

3.1.27 netplan.yml

Synopsis: Configure netplan.

Description of the task.

[netplan.yml]

```
1  ---
2  # linux-postinstall netplan
3
4  # Configure 01-network-manager-all.yaml only if it already exists
5  - name: "netplan: Stat {{ lp_netplan_default }}"
6    stat:
7      path: "{{ lp_netplan_root }}/{{ lp_netplan_default }}"
8      register: result
9
10 - name: "netplan: Configure {{ lp_netplan_root }}/{{ lp_netplan_default }}"
11   template:
12     src: netplan-default.j2
13     dest: "{{ lp_netplan_root }}/{{ lp_netplan_default }}"
14     owner: "{{ lp_netplan_owner }}"
15     group: "{{ lp_netplan_group }}"
16     mode: "{{ lp_netplan_mode }}"
17     backup: "{{ lp_backup_conf }}"
18     notify: netplan apply
19     when: result.stat.exists|default(false)
20
21 - name: "netplan: Configure files in {{ lp_netplan_root }}"
22   template:
23     src: netplan-conf.j2
24     dest: "{{ lp_netplan_root }}/{{ item.file }}"
25     owner: "{{ item.owner | default(lp_netplan_owner) }}"
26     group: "{{ item.group | default(lp_netplan_group) }}"
27     mode: "{{ item.mode | default(lp_netplan_mode) }}"
28     backup: "{{ lp_backup_conf }}"
29     loop: "{{ lp_netplan_conf }}"
30     notify: netplan apply
31
32   # EOF
33   ...
```

3.1.28 nfsd.yml

Synopsis: Configure nfsd.

Description of the task.

[nfsd.yml]

```

1 ---
2 # linux-postinstall nfsd
3
4 - name: "nfsd: Install packages"
5   include_tasks: fn/install-package.yml
6   loop: "{{ lp_nfsd_packages }}"
7   tags: lp_nfsd_packages
8
9 - name: "nfsd: Configure exports"
10  template:
11    src: exports.j2
12    dest: /etc/exports
13    owner: root
14    group: root
15    mode: "0644"
16  notify: reload nfsd
17  tags: lp_nfsd_exports
18
19 - name: "nfsd: Enable and start nfsd services"
20  systemd:
21    name: "{{ item }}"
22    enabled: true
23    state: started
24  loop: "{{ lp_nfsd_services }}"
25  when:
26    - lp_nfsd_enable|bool
27    - lp_nfsd_services|length > 0
28  tags: lp_nfsd_service
29
30 - name: "nfsd: Stop and disable nfsd services"
31  systemd:
32    name: "{{ item }}"
33    enabled: false
34    state: stopped
35  loop: "{{ lp_nfsd_services }}"
36  when:
37    - not lp_nfsd_enable|bool
38    - lp_nfsd_services|length > 0
39  tags: lp_nfsd_service
40
41 # EOF
42 ...

```

3.1.29 packages-auto.yml

Synopsis: Configure packages-auto.

Description of the task.

[packages-auto.yml]

```

1 ---
2 # linux-postinstall packages-auto
3
4 - name: "packages-auto: Init variable local_pkg_lists"
5   set_fact:

```

(continues on next page)

(continued from previous page)

```

6     local_pkg_lists: []
7     local_pkg_list: []
8     tags: lp_packages_auto
9
10 - name: "packages-auto: List variables ^lp_.*_packages$"
11   set_fact:
12     local_pkg_lists: "{{ local_pkg_lists +
13                          [{'install': item.split('_')[0] + '_' + item.split('_')[1],
14                           'packages': item}] }}"
15   loop: "{{ hostvars[inventory_hostname].keys() |
16             select('match', '^lp_.*_packages$') |
17             list }}"
18   tags: lp_packages_auto
19
20 - name: "packages-auto: Debug local_pkg_lists"
21   debug:
22     msg: "[{{ lookup('vars', item.install, default='false')|bool }}]
23          packages {{ lookup('vars', item.packages) }}"
24   loop: "{{ local_pkg_lists }}"
25   when: lp_packages_debug|bool
26   tags: lp_packages_auto
27
28 - name: "packages-auto: Create local_pkg_list"
29   set_fact:
30     local_pkg_list: "{{ local_pkg_list + lookup('vars', item.packages) }}"
31   loop: "{{ local_pkg_lists }}"
32   when: lookup('vars', item.install, default='False')|bool
33   tags: lp_packages_auto
34
35 - name: "packages-auto: Debug local_pkg_list"
36   debug:
37     var: local_pkg_list
38   when: lp_packages_debug|bool
39   tags: lp_packages_auto
40
41 - name: "packages-auto: Install packages"
42   include_tasks: fn/install-package.yml
43   loop: "{{ local_pkg_list }}"
44   tags: lp_packages_auto
45
46 # EOF
47 ...

```

3.1.30 packages.yml

Synopsis: Configure packages.

Description of the task.

[packages.yml]

```

1 ---
2 # linux-postinstall packages
3
4 - name: "packages: Debug"
5   vars:

```

(continues on next page)

(continued from previous page)

```

6     msg: |
7         ansible_os_family [{{ ansible_os_family }}]
8         lp_packages_auto [{{ lp_packages_auto }}]
9         lp_packages_autoremove [{{ lp_packages_autoremove }}]
10        lp_packages_selections_preinstall
11        {{ lp_packages_selections_preinstall|to_nice_yaml }}
12        lp_packages_install
13        {{ lp_packages_install|to_nice_yaml }}
14        lp_packages_remove
15        {{ lp_packages_remove|to_nice_yaml }}
16        lp_packages_selections_postinstall
17        {{ lp_packages_selections_postinstall|to_nice_yaml }}
18    debug:
19        msg: "{{ msg.split('\n')[:-1] }}"
20    when: lp_packages_debug|bool
21    tags: lp_packages_debug
22
23 - name: "packages: Configure package selections before Install/Remove"
24   dpkg_selections:
25       name: "{{ item.name }}"
26       selection: "{{ item.selection }}"
27   loop: "{{ lp_packages_selections_preinstall }}"
28   when: ansible_os_family == "Debian"
29   tags: lp_packages_selections_preinstall
30
31 - name: "packages: Install packages listed in variables lp*_packages"
32   include_tasks: packages-auto.yml
33   when: lp_packages_auto|bool
34   tags: lp_packages_auto
35
36 - name: "packages: Install packages"
37   include_tasks: fn/install-package.yml
38   loop: "{{ lp_packages_install }}"
39   tags: lp_packages_install
40
41 - name: "packages: Remove packages"
42   include_tasks: fn/remove-package.yml
43   loop: "{{ lp_packages_remove }}"
44   tags: lp_packages_remove
45
46 - name: "packages: Configure package selections after Install/Remove"
47   dpkg_selections:
48       name: "{{ item.name }}"
49       selection: "{{ item.selection }}"
50   loop: "{{ lp_packages_selections_postinstall }}"
51   when: ansible_os_family == "Debian"
52   tags: lp_packages_selections_postinstall
53
54 # EOF
55 ...

```

3.1.31 passwords.yml

Synopsis: Configure passwords.

Description of the task.

[passwords.yml]

```

1  ---
2  # linux-postinstall passwords
3
4  - name: "passwords: Debug"
5    vars:
6      msg: |
7        lp_passwords_fail_gracefully [{{ lp_passwords_fail_gracefully }}]
8        lp_password_update_password [{{ lp_password_update_password }}]
9        lp_users
10       {% if lp_passwords_debug_classified|bool %}
11       {{ lp_users|default([])|to_nice_yaml }}
12       {% else %}
13       {% for user in lp_users|default([]) %}
14       - userpass: *****
15       {% for k,v in user.items() %}
16       {% if k not in ['userpass'] %}
17         {{ k }}: {{ v }}
18       {% endif %}
19       {% endfor %}
20       {% endfor %}
21       {% endif %}
22
23       lp_passwordstore [{{ lp_passwordstore }}]
24       lp_passwordstore_install [{{ lp_passwordstore_install }}]
25       lp_passwordstore_debug [{{ lp_passwordstore_debug }}]
26       lp_passwordstore_backup [{{ lp_passwordstore_backup }}]
27       lp_passwordstore_create [{{ lp_passwordstore_create }}]
28       lp_passwordstore_length [{{ lp_passwordstore_length }}]
29       lp_passwordstore_nosymbols [{{ lp_passwordstore_nosymbols }}]
30       lp_passwordstore_overwrite [{{ lp_passwordstore_overwrite }}]
31       lp_passwordstore_passwordstore [{{ lp_passwordstore_passwordstore }}]
32       lp_passwordstore_returnall [{{ lp_passwordstore_returnall }}]
33       lp_passwordstore_subkey [{{ lp_passwordstore_subkey }}]
34       lp_passwordstore_idempotent_password_hash [{{ lp_passwordstore_idempotent_
35       ↪password_hash }}]
36       lp_passwordstore_packages
37       {{ lp_passwordstore_packages|to_nice_yaml }}
38    debug:
39      msg: "{{ msg.split('\n')[:-1] }}"
40    when: lp_passwords_debug|bool
41    tags: lp_passwords_debug
42
43  - name: "passwords: Passwordstore"
44    block:
45      - name: "passwords: Passwordstore: Install packages"
46        include_tasks: fn/install-package.yml
47        loop:
48          - "{{ lp_passwordstore_packages }}"
49          - "{{ lp_gpg_packages }}"
50          - "{{ lp_gpg_packages_extra }}"
51        vars:
52          my_delegate_to_localhost: true
53          run_once: true
54          when: lp_passwordstore_install|bool
55      - name: "passwords: Passwordstore: Retrieve, create, or update userpass"
56        include_role:

```

(continues on next page)

(continued from previous page)

```

56     name: vbotka.ansible_lib
57     tasks_from: al_pws_user_host.yml
58     vars:
59         al_pws_debug: "{{ lp_passwordstore_debug }}"
60         al_pws_backup: "{{ lp_passwordstore_backup }}"
61         al_pws_create: "{{ lp_passwordstore_create }}"
62         al_pws_length: "{{ lp_passwordstore_length }}"
63         al_pws_nosymbols: "{{ lp_passwordstore_nosymbols }}"
64         al_pws_overwrite: "{{ lp_passwordstore_overwrite }}"
65         al_pws_passwordstore: "{{ lp_passwordstore_passwordstore }}"
66         al_pws_returnall: "{{ lp_passwordstore_returnall }}"
67         al_pws_subkey: "{{ lp_passwordstore_subkey }}"
68         al_pws_idempotent_password_hash: "{{ lp_passwordstore_idempotent_password_
↪ hash }}"
69         al_pws_query: "{{ lp_users }}"
70     register: result
71     - name: "passwords: Passwordstore: Create my_passwords"
72     set_fact:
73         my_passwords: "{{ my_passwords|default([]) +
74                             [item|dict2items|
75                             rejectattr('key', 'equalto', 'userpass')|
76                             list|items2dict|
77                             combine({'update_password': lp_password_update_password})] ] }
↪ }}"
78     loop: "{{ al_pws_query_result }}"
79     loop_control:
80         label: "{{ item.name }}"
81     - name: "passwords: Passwordstore: Debug my_passwords"
82     debug:
83         var: my_passwords
84     when: lp_passwords_debug|bool
85     - name: "passwords: Passwordstore: Include users"
86     include_tasks: users.yml
87     vars:
88         lp_users: "{{ my_passwords }}"
89     rescue:
90     - name: "passwords: Passwordstore: Debug fail"
91     debug:
92         var: result
93     when: lp_passwords_debug_classified|bool
94     - name: "passwords: Passwordstore: Fail"
95     fail:
96         msg: "[ERROR] Passwordstore failed."
97     when: not lp_passwords_fail_gracefully|bool
98     when: lp_passwordstore|bool
99     tags: lp_passwords_passwordstore
100
101 # EOF
102 ...

```

3.1.32 pm-utils.yml

Synopsis: Configure pm-utils.

Description of the task.

[pm-utils.yml]

```

1 ---
2 # linux-postinstall pm-utils
3
4 # TODO:
5 # 1) add variables: lp_pm_powerd, lp_pm_configd
6 # 2) add templates: pm-powerd.j2, pm-configd.j2
7 # 3) add cases: resume, thaw, suspend, hibernate
8 # 4) install pm-utils
9
10 - name: "pm_utils: Configure /etc/pm/sleep.d"
11   template:
12     src: pm-sleepd.j2
13     dest: "/etc/pm/sleep.d/{{ item.value.file }}"
14     owner: root
15     group: root
16     mode: "0755"
17     backup: "{{ lp_backup_conf }}"
18   with_dict: "{{ lp_pm_sleepd|default({}) }}"
19   when: item.value.file|length > 0
20
21 # EOF
22 ...

```

3.1.33 postfix.yml

Synopsis: Configure postfix.

Description of the task.

[postfix.yml]

```

1 ---
2 # linux-postinstall postfix
3
4 - name: "postfix: Debug"
5   vars:
6     msg: |
7       ansible_os_family [{{ ansible_os_family }}]
8       lp_postfix_service [{{ lp_postfix_service }}]
9       lp_postfix_enable [{{ lp_postfix_enable }}]
10      lp_postfix_main_conf
11      {{ lp_postfix_main_conf|to_yaml }}
12   debug:
13     msg: "{{ msg.split('\n')[:-1] }}"
14   when: lp_postfix_debug|bool
15   tags: lp_postfix_debug
16
17 - name: "postfix: Configure /etc/postfix/main.cf"
18   lineinfile:
19     dest: /etc/postfix/main.cf
20     regexp: '^s*{{ item.key }}s*=\s*(.*)$'
21     line: "{{ item.key }} = {{ item.value }}"
22     create: true
23     backup: "{{ lp_backup_conf }}"
24     loop: "{{ lp_postfix_main_conf }}"

```

(continues on next page)

(continued from previous page)

```

25  notify: reload postfix
26  tags: lp_postfix_conf
27
28  - name: "postfix: Enable and start postfix"
29    systemd:
30      name: "{{ lp_postfix_service }}"
31      enabled: true
32      state: started
33    when: lp_postfix_enable|bool
34    tags: lp_postfix_service
35
36  - name: "postfix: Disable and stop postfix"
37    systemd:
38      name: "{{ lp_postfix_service }}"
39      enabled: false
40      state: stopped
41    when: not lp_postfix_enable|bool
42    tags: lp_postfix_service
43
44  # EOF
45  ...

```

3.1.34 reboot.yml

Synopsis: Configure reboot.

Description of the task.

[reboot.yml]

```

1  ---
2  # linux-postinstall reboot
3
4  - name: "reboot Debug"
5    vars:
6      msg: |
7        lp_reboot_force [{{ lp_reboot_force }}]
8        lp_reboot_required_ignore [{{ lp_reboot_required_ignore }}]
9        lp_reboot_required_file [{{ lp_reboot_required_file }}]
10       lp_reboot_command [{{ lp_reboot_command }}]
11       lp_reboot_wait_connect_timeout [{{ lp_reboot_wait_connect_timeout }}]
12       lp_reboot_wait_sleep [{{ lp_reboot_wait_sleep }}]
13       lp_reboot_wait_delay [{{ lp_reboot_wait_delay }}]
14       lp_reboot_wait_timeout [{{ lp_reboot_wait_timeout }}]
15    debug:
16      msg: "{{ msg.split('\n')[:-1] }}"
17    when: lp_reboot_debug|bool
18
19  - name: "reboot: Debian test {{ lp_reboot_required_file }}"
20    block:
21      - name: "reboot: Stat {{ lp_reboot_required_file }}"
22        stat:
23          path: "{{ lp_reboot_required_file }}"
24          register: reboot_required_file_status
25      - name: "reboot: Set reboot_required"
26        set_fact:

```

(continues on next page)

(continued from previous page)

```

27     reboot_required: "{{ reboot_required_file_status.exists |
28                          default(false) }}"
29   when: ansible_os_family == "Debian"
30
31 - name: "reboot: RedHat test {{ lp_reboot_required_command }}"
32   block:
33     - name: "reboot: Run {{ lp_reboot_required_command }}"
34       command: "{{ lp_reboot_required_command }}"
35       register: reboot_required_cmd_status
36     - name: "reboot: Set reboot_required"
37       set_fact:
38         reboot_required: "{{ (reboot_required_cmd_status.rc != 0) |
39                              ternary(true, false) }}"
40   when: ansible_os_family == "RedHat"
41
42 - name: "reboot: Debug reboot_required"
43   debug:
44     var: reboot_required
45   when: lp_reboot_debug|bool
46
47 - name: "reboot: Reboot and wait for connection"
48   reboot:
49     connect_timeout: "{{ lp_reboot_wait_connect_timeout }}"
50     post_reboot_delay: "{{ lp_reboot_wait_delay }}"
51     reboot_timeout: "{{ lp_reboot_wait_timeout }}"
52   when: (reboot_required|default(false) and
53          (not lp_reboot_required_ignore)) or
54          lp_reboot_force|bool
55
56 # - name: "reboot: Reboot and wait for connection"
57 #   block:
58 #     - name: "reboot: Reboot" # noqa 305
59 #       shell: "{{ lp_reboot_command }}"
60 #       async: 1
61 #       poll: 0
62 #     - name: "reboot: Wait for connection"
63 #       wait_for_connection:
64 #         connect_timeout: "{{ lp_reboot_wait_connect_timeout }}"
65 #         sleep: "{{ lp_reboot_wait_sleep }}"
66 #         delay: "{{ lp_reboot_wait_delay }}"
67 #         timeout: "{{ lp_reboot_wait_timeout }}"
68 #       when: (reboot_required|default(false) and
69 #              (not lp_reboot_required_ignore)) or lp_reboot_force
70
71 # EOF
72 ...

```

3.1.35 repos.yml

Synopsis: Configure repos.

Description of the task.

[repos.yml]

```

1  ---
2  # linux-postinstall repos
3
4  - name: "repos: Debug"
5    vars:
6      msg: |
7        lp_repos_keys
8        {{ lp_repos_keys|to_nice_yaml }}
9        lp_repos
10       {{ lp_repos|to_nice_yaml }}
11    debug:
12      msg: "{{ msg.split('\n')[:-1] }}"
13    when: lp_repos_debug|bool
14    tags: lp_repos_debug
15
16 - name: "repos: Manage repo signing keys"
17   apt_key:
18     data: "{{ item.data|default(omit) }}"
19     file: "{{ item.file|default(omit) }}"
20     id: "{{ item.id|default(omit) }}"
21     keyring: "{{ item.keyring|default(omit) }}"
22     keyserver: "{{ item.keyserver|default(omit) }}"
23     state: "{{ item.state|default(omit) }}"
24     url: "{{ item.url|default(omit) }}"
25     validate_certs: "{{ item.validate_certs|default(omit) }}"
26   loop: "{{ lp_repos_keys }}"
27   register: result
28   retries: "{{ lp_install_retries }}"
29   until: result is succeeded
30   delay: "{{ lp_install_delay }}"
31   tags: lp_repos_keys_manage
32
33 - name: "repos: Manage repositories"
34   apt_repository:
35     codename: "{{ item.codename|default(omit) }}"
36     filename: "{{ item.filename|default(omit) }}"
37     mode: "{{ item.mode|default(omit) }}"
38     repo: "{{ item.repo|mandatory }}"
39     state: "{{ item.state|default(omit) }}"
40     update_cache: "{{ item.update_cache|default(omit) }}"
41     validate_certs: "{{ item.validate_certs|default(omit) }}"
42   loop: "{{ lp_repos }}"
43   tags: lp_repos_manage
44
45 # EOF
46 ...

```

3.1.36 resolvconf.yml

Synopsis: Configure resolvconf.

Description of the task.

[[resolvconf.yml](#)]

```

1 ---
2 # linux-postinstall resolvconf
3
4 - name: "resolvconf: Debug"
5   vars:
6     msg: |
7       lp_resolvconf_service [{{ lp_resolvconf_service }}]
8       lp_resolvconf_enable [{{ lp_resolvconf_enable }}]
9       lp_package_state [{{ lp_package_state }}]
10      lp_resolvconf_packages
11      {{ lp_resolvconf_packages|to_nice_yaml }}
12      lp_resolvconf_conf_d_head_path [{{ lp_resolvconf_conf_d_head_path }}]
13      lp_resolvconf_conf_owner [{{ lp_resolvconf_conf_owner }}]
14      lp_resolvconf_conf_group [{{ lp_resolvconf_conf_group }}]
15      lp_resolvconf_conf_mode [{{ lp_resolvconf_conf_mode }}]
16      lp_resolvconf_conf_d_head
17      {{ lp_resolvconf_conf_d_head|to_yaml }}
18   debug:
19     msg: "{{ msg.split('\n')[:-1] }}"
20   when: lp_resolvconf_debug|bool
21   tags: lp_resolvconf_debug
22
23 - name: "resolvconf: Install packages"
24   include_tasks: fn/install-package.yml
25   loop: "{{ lp_resolvconf_packages }}"
26   tags: lp_resolvconf_packages
27
28 - name: "resolvconf: Configure {{ lp_resolvconf_conf_d_head_path }}"
29   template:
30     src: resolvconf-conf_d-head.j2
31     dest: "{{ lp_resolvconf_conf_d_head_path }}"
32     owner: "{{ lp_resolvconf_conf_owner }}"
33     group: "{{ lp_resolvconf_conf_group }}"
34     mode: "{{ lp_resolvconf_conf_mode }}"
35     backup: "{{ lp_backup_conf }}"
36   notify: restart resolvconf
37   tags: lp_resolvconf_conf_d_head
38
39 - name: "resolvconf: Enable and start resolvconf"
40   systemd:
41     name: "{{ lp_resolvconf_service }}"
42     enabled: true
43     state: started
44   when: lp_resolvconf_enable|bool
45   tags: lp_resolvconf_service
46
47 - name: "resolvconf: Disable and stop resolvconf"
48   systemd:
49     name: "{{ lp_resolvconf_service }}"
50     enabled: false
51     state: stopped
52   when: not lp_resolvconf_enable|bool
53   tags: lp_resolvconf_service
54
55 # EOF
56 ...

```

3.1.37 service.yml

Synopsis: Configure service.

Description of the task.

[service.yml]

```

1 ---
2 # linux-postinstall service
3
4 - name: "service: Set my_service_name_vars"
5   set_fact:
6     my_service_name_vars: "{{ my_service_name_vars|default([]) +
7                               [{item: lookup('vars', 'lp_' ~ item ~ '_service')}]} }}"
8   loop: "{{ lp_service_enable }}"
9   when: lookup('vars', 'lp_' ~ item, default='false')
10  tags: lp_service_debug
11
12 - name: "service: Set my_service_enable_vars"
13   set_fact:
14     my_service_enable_vars: "{{ my_service_enable_vars|default([]) +
15                                 [{item: lookup('vars', 'lp_' ~ item ~ '_enable')}]} }}"
16   loop: "{{ lp_service_enable }}"
17   when: lookup('vars', 'lp_' ~ item, default='false')
18   tags: lp_service_debug
19
20 - name: "service: Debug"
21   debug:
22     msg: "{{ my_msg.split('\n')[:-1] }}"
23   vars:
24     my_msg: |
25       lp_service
26       {{ lp_service|to_nice_yaml }}
27       lp_service_enable
28       {{ lp_service_enable|to_nice_yaml }}
29       my_service_name_vars
30       {{ my_service_name_vars|default([])|to_nice_yaml }}
31       my_service_enable_vars
32       {{ my_service_enable_vars|default([])|to_nice_yaml }}
33   when: lp_service_debug|bool
34   tags: lp_service_debug
35
36 - name: "service: Automatically enable or disable services managed by this role"
37   service:
38     name: "{{ lookup('vars', 'lp_' ~ item ~ '_service') }}"
39     enabled: "{{ lookup('vars', 'lp_' ~ item ~ '_enable') }}"
40     loop: "{{ lp_service_enable }}"
41     when: lookup('vars', 'lp_' ~ item, default='false')
42     tags: lp_service_auto
43
44 - name: "service: General managment of services"
45   service:
46     name: "{{ item.name }}"
47     state: "{{ item.state|default(omit) }}"
48     enabled: "{{ item.enabled|default(omit) }}"
49     arguments: "{{ item.arguments|default(omit) }}"
50     pattern: "{{ item.pattern|default(omit) }}"
51     runlevel: "{{ item.runlevel|default(omit) }}"

```

(continues on next page)

(continued from previous page)

```

52     sleep: "{{ item.sleep|default(omit) }}"
53     use:   "{{ item.use|default('omit') }}"
54     loop:  "{{ lp_service }}"
55     when:  (item.state is defined) or
56            (item.enabled is defined)
57     tags:  lp_service_general
58
59     # TODO: Mask a service. Do not allow any service to activate a masked
60     # service. See tasks/wpagui.yml
61
62     # EOF
63     ...

```

3.1.38 smart.yml

Synopsis: Configure smart.

Description of the task.

[smart.yml]

```

1  ---
2  # linux-postinstall smart
3
4  - name: "smart: Install packages"
5    include_tasks: fn/install-package.yml
6    loop:  "{{ lp_smart_packages }}"
7    tags:  lp_smart_packages
8
9  - name: "smart: Configure {{ lp_smart_conf_file }}. Do not scan for devices"
10    lineinfile:
11      state: absent
12      dest:  "{{ lp_smart_conf_file }}"
13      regexp: '^\\s*DEVICESCAN\\s*(.*)$'
14      owner:  "{{ lp_smart_conf_owner }}"
15      group:  "{{ lp_smart_conf_group }}"
16      mode:   "{{ lp_smart_conf_mode }}"
17      create: true
18      backup: "{{ lp_backup_conf }}"
19      when:   not lp_smart_devicescan|bool
20      notify: reload smart
21      tags:  lp_smart_conf
22
23  - name: "smart: Configure devices in {{ lp_smart_conf_file }}"
24    lineinfile:
25      dest:  "{{ lp_smart_conf_file }}"
26      regexp: "{{ item.regexp }}"
27      line:   "{{ item.line }}"
28      owner:  "{{ lp_smart_conf_owner }}"
29      group:  "{{ lp_smart_conf_group }}"
30      mode:   "{{ lp_smart_conf_mode }}"
31      create: true
32      backup: "{{ lp_backup_conf }}"
33      loop:   "{{ lp_smart_devices }}"
34      notify: reload smart
35      tags:  lp_smart_conf

```

(continues on next page)

(continued from previous page)

```

36
37 - name: "smart: Start and enable smart"
38   service:
39     name: "{{ lp_smart_service }}"
40     state: started
41     enabled: true
42   register: result
43   when: lp_smart_enable|bool
44   tags: lp_smart_service
45
46 - name: "smart: Debug service"
47   debug:
48     var: result
49   when: lp_smart_debug|bool
50
51 - name: "smart: Stop and disable smart"
52   service:
53     name: "{{ lp_smart_service }}"
54     state: stopped
55     enabled: false
56   register: result
57   when: not lp_smart_enable|bool
58   tags: lp_smart_service
59
60 - name: "smart: Debug service"
61   debug:
62     var: result
63   when: lp_smart_debug|bool
64
65 # EOF
66 ...

```

3.1.39 speechd.yml

Synopsis: Configure speechd.

Description of the task.

[speechd.yml]

```

1 ---
2 # linux-postinstall speechd
3
4 - name: "speechd: Debug"
5   debug:
6     msg: "lp_speechd_enable [{{ lp_speechd_enable }}"
7     when: lp_speechd_debug|bool
8
9 - name: "speechd: Enable and start speech-dispatcher"
10  systemd:
11    name: "{{ lp_speechd_service }}"
12    enabled: true
13    state: started
14    when: lp_speechd_enable|bool
15
16 - name: "speechd: Stop and disable speech-dispatcher"

```

(continues on next page)

(continued from previous page)

```

17  systemd:
18      name: "{{ lp_speechd_service }}"
19      enabled: false
20      state: stopped
21      when: not lp_speechd_enable|bool
22
23  # EOF
24  ...

```

3.1.40 sshd.yml

Synopsis: Configure sshd.

Description of the task.

[sshd.yml]

```

1  ---
2  # linux-postinstall sshd
3
4  - name: "sshd: Debug"
5    vars:
6      msg: |
7          lp_sshd_config
8          {{ lp_sshd_config|to_yaml }}
9    debug:
10      msg: "{{ msg.split('\n')[:-1] }}"
11    when: lp_sshd_debug|bool
12    tags: lp_sshd_debug
13
14  - name: "sshd: Configure /etc/ssh/sshd_config"
15    lineinfile:
16      dest: /etc/ssh/sshd_config
17      regexp: "^\\s*{{ item.key }}\\s*(.*)$"
18      insertbefore: "^{{ '#' }}{{ item.key }}\\s*(.*)$"
19      line: "{{ item.key }} {{ item.value }}"
20      backup: "{{ lp_backup_conf }}"
21      validate: "{{ lp_sshd_path }} -t -f %s"
22      loop: "{{ lp_sshd_config }}"
23      notify: reload sshd
24      tags: lp_sshd_config
25
26  - name: "sshd: Enable and start sshd"
27    systemd:
28      name: "{{ lp_sshd_service }}"
29      enabled: true
30      state: started
31      when: lp_sshd_enable|bool
32      tags: lp_sshd_service
33
34  - name: "sshd: Disable and stop sshd"
35    systemd:
36      name: "{{ lp_sshd_service }}"
37      enabled: false
38      state: stopped
39      when: not lp_sshd_enable|bool

```

(continues on next page)

(continued from previous page)

```

40 tags: lp_sshd_service
41
42 # EOF
43 ...

```

3.1.41 ssh.yml

Synopsis: Configure ssh.

Description of the task.

[ssh.yml]

```

1 ---
2 # linux-postinstall ssh
3
4 - name: "ssh: Debug"
5   vars:
6     msg: |
7       lp_ssh_config
8       {{ lp_ssh_config|to_yaml }}
9   debug:
10    msg: "{{ msg.split('\n')[:-1] }}"
11  when: lp_ssh_debug|bool
12  tags: lp_ssh_debug
13
14 - name: "ssh: Configure /etc/ssh/ssh_config"
15   template:
16     src: ssh_config.j2
17     dest: /etc/ssh/ssh_config
18     backup: "{{ lp_backup_conf }}"
19     tags: lp_ssh_conf
20
21 # EOF
22 ...

```

3.1.42 sudoers.yml

Synopsis: Configure sudoers.

Description of the task.

[sudoers.yml]

```

1 ---
2 # linux-postinstall sudoers
3
4 - name: "sudoers: Debug"
5   vars:
6     msg: |
7       lp_sudoers_conf
8       {{ lp_sudoers_conf|to_yaml }}
9   debug:
10    msg: "{{ msg.split('\n')[:-1] }}"
11  when: lp_sudoers_debug|bool

```

(continues on next page)

(continued from previous page)

```

12 tags: lp_sudoers_debug
13
14 - name: "sudoers: Configure /etc/sudoers"
15   lineinfile:
16     path: /etc/sudoers
17     line: "{{ item.line }}"
18     state: "{{ item.state|default('present') }}"
19     create: true
20     backup: "{{ lp_backup_conf }}"
21     loop: "{{ lp_sudoers_conf }}"
22     tags: lp_sudoers_conf
23
24 - name: "sudoers: Configure /etc/sudoers.d/01"
25   lineinfile:
26     path: /etc/sudoers.d/01
27     line: "{{ item }}"
28     owner: "{{ lp_sudoers_owner }}"
29     group: "{{ lp_sudoers_group }}"
30     mode: "{{ lp_sudoers_mode }}"
31     create: true
32     backup: "{{ lp_backup_conf }}"
33     loop: "{{ lp_sudoers_01 }}"
34     tags: lp_sudoers_dconf
35
36 # EOF
37 ...

```

3.1.43 swap.yml

Synopsis: Configure swap.

Description of the task.

[swap.yml]

```

1 ---
2 # linux-postinstall swap
3
4 - name: "swap: Debug"
5   vars:
6     msg: |
7       lp_swap [{{ lp_swap }}]
8       lp_swap_enable [{{ lp_swap_enable }}]
9       lp_swap_file [{{ lp_swap_file|default("UNDEFINED") }}]
10      lp_swap_size [{{ lp_swap_size|default("UNDEFINED") }}]
11      lp_swap_stsize [{{ lp_swap_stsize|default("UNDEFINED") }}]
12   debug:
13     msg: "{{ msg.split('\n')[:-1] }}"
14   when: lp_swap_debug|bool
15   tags: lp_swap_debug
16
17 - name: "swap: Create swapfile {{ lp_swap_file }}"
18   shell: sh -c 'if [ ! -e {{ lp_swap_file }} ]; then printf "create"; fi'
19   register: command_result
20   changed_when: command_result.stdout == "create"
21   notify: create and mount swap file

```

(continues on next page)

(continued from previous page)

```

22 tags: lp_swap_swapfile
23
24 - name: "swap: Change swapfile {{ lp_swap_file }}"
25   shell: >
26     sh -c
27     'if [ -e {{ lp_swap_file }} ] &&
28     [ "`stat --format '%s' {{ lp_swap_file }}" -ne "{{ lp_swap_stsize }}" ];
29     then printf "change";
30     fi'
31   register: command_result
32   changed_when: command_result.stdout == "change"
33   notify: change and mount swap file
34   tags: lp_swap_swapfile
35
36 - name: "swap: Create swap entry in /etc/fstab"
37   mount:
38     name: "none"
39     src: "{{ lp_swap_file }}"
40     fstype: swap
41     opts: sw
42     passno: "0"
43     dump: "0"
44     state: present
45     backup: "{{ lp_backup_conf }}"
46   when: lp_swap_enable|bool
47   tags: lp_swap_fstab
48
49 - name: "swap: Remove swap entry from /etc/fstab"
50   mount:
51     name: "none"
52     src: "{{ lp_swap_file }}"
53     fstype: swap
54     opts: sw
55     passno: 0
56     dump: 0
57     state: absent
58     backup: "{{ lp_backup_conf }}"
59   notify: remove swap file
60   when:
61     - not lp_swap_enable|bool
62     - lp_swap_file is defined
63   tags: lp_swap_swapfile
64
65 # EOF
66 ...

```

3.1.44 sysctl.yml

Synopsis: Configure sysctl.

Description of the task.

[sysctl.yml]

```

1 ---
2 # linux-postinstall sysctl

```

(continues on next page)

(continued from previous page)

```

3
4 - name: "sysctl: Debug"
5   vars:
6     msg: |
7       lp_sysctl_vars
8       {{ lp_sysctl_vars|to_yaml }}
9   debug:
10    msg: "{{ msg.split('\n')[:-1] }}"
11  when: lp_sysctl_debug|bool
12  tags: lp_sysctl_debug
13
14 - name: "sysctl: Configure /etc/sysctl.conf"
15   lineinfile:
16     dest: /etc/sysctl.conf
17     regexp: '^s*{{ item.var }}s*=(.*)$'
18     line: "{{ item.var }} = {{ item.value }}"
19     backup: "{{ lp_backup_conf }}"
20     loop: "{{ lp_sysctl_vars }}"
21     notify: load sysctl settings
22
23 # EOF
24 ...

```

3.1.45 timesyncd.yml

Synopsis: Configure timesyncd.

Description of the task.

[timesyncd.yml]

```

1 ---
2 # linux-postinstall timesyncd
3
4 - name: "timesyncd: Debug"
5   vars:
6     msg: |
7       lp_timesyncd [{{ lp_timesyncd }}]
8       lp_timesyncd_NTP [{{ lp_timesyncd_NTP }}]
9       lp_timesyncd_FallbackNTP [{{ lp_timesyncd_FallbackNTP }}]
10      lp_timesyncd_RootDistanceMaxSec [{{ lp_timesyncd_RootDistanceMaxSec }}]
11      lp_timesyncd_PollIntervalMinSec [{{ lp_timesyncd_PollIntervalMinSec }}]
12      lp_timesyncd_PollIntervalMaxSec [{{ lp_timesyncd_PollIntervalMaxSec }}]
13   debug:
14    msg: "{{ msg.split('\n')[:-1] }}"
15  when: lp_timesyncd_debug|bool
16  tags: lp_timesyncd_debug
17
18 - name: "timesyncd: Configure /etc/systemd/timesyncd.conf"
19   template:
20     src: timesyncd.conf.j2
21     dest: /etc/systemd/timesyncd.conf
22     owner: root
23     group: root
24     mode: "0644"
25     backup: "{{ lp_backup_conf }}"

```

(continues on next page)

(continued from previous page)

```

26  notify: restart timesyncd
27  tags: lp_timesyncd_conf
28
29  - name: "timesyncd: Enable and start timesyncd"
30    service:
31      name: "{{ lp_timesyncd_service }}"
32      state: started
33      enabled: true
34    when: lp_timesyncd_enable|bool
35    tags: lp_timesyncd_service
36
37  - name: "timesyncd: Disable and stop timesyncd"
38    service:
39      name: "{{ lp_timesyncd_service }}"
40      state: stopped
41      enabled: false
42    when: not lp_timesyncd_enable|bool
43    tags: lp_timesyncd_service
44
45  # Notes on CentOS
46  # * systemd compiled without timesyncd service in CentOS 7 ?
47  # * use ntpd or chrony ?
48  # https://unix.stackexchange.com/questions/286708/
49  # centos-7-2-minimal-time-synchronization-timedated-and-or-ntpdc-chrony
50  # https://www.freedesktop.org/wiki/Software/systemd/timedated/
51
52  # EOF
53  ...

```

3.1.46 timezone.yml

Synopsis: Configure timezone.

Description of the task.

[timezone.yml]

```

1  ---
2  # linux-postinstall timezone
3
4  - name: "timezone: Debug"
5    debug:
6      msg: "lp_timezone_zoneinfo [{{ lp_timezone_zoneinfo }}"
7    when: lp_timezone_debug|bool
8    tags: lp_timezone_debug
9
10 - name: "timezone: Set timezone {{ lp_timezone_zoneinfo }}"
11   timezone:
12     name: "{{ lp_timezone_zoneinfo|default('UTC') }}"
13   tags: lp_timezone_set
14
15  # EOF
16  ...

```

3.1.47 tlp.yml

Synopsis: Configure tlp.

Description of the task.

[tlp.yml]

```

1  ---
2  # linux-postinstall tlp
3
4  - name: "tlp: Debug"
5    vars:
6      msg: |
7        lp_tlp_enable [{{ lp_tlp_enable }}]
8        lp_tlp_thinkpad [{{ lp_tlp_thinkpad }}]
9        lp_tlp_packages
10       {{ lp_tlp_packages|to_nice_yaml }}
11       lp_tlp_packages_tp
12       {{ lp_tlp_packages_tp|to_nice_yaml }}
13       lp_tlp_config_file [{{ lp_tlp_config_file }}]
14       lp_tlp_config
15       {{ lp_tlp_config|to_nice_yaml }}
16       lp_tlp_services
17       {{ lp_tlp_services|to_nice_yaml }}
18       lp_tlp_restart_service [{{ lp_tlp_restart_service }}]
19     debug:
20       msg: "{{ msg.split('\n')[:-1] }}"
21     when: lp_tlp_debug|bool
22     tags: lp_tlp_debug
23
24  - name: "tlp: Install packages"
25    include_tasks: fn/install-package.yml
26    loop: "{{ lp_tlp_packages }}"
27    tags: lp_tlp_packages
28
29  - name: "tlp: Install packages for ThinkPad"
30    include_tasks: fn/install-package.yml
31    loop: "{{ lp_tlp_packages_tp }}"
32    when: lp_tlp_thinkpad|bool
33    tags: lp_tlp_packages
34
35  - name: "tlp: Configure {{ lp_tlp_config_file }}"
36    lineinfile:
37      dest: "{{ lp_tlp_config_file }}"
38      regexp: '^\\s*{{ item.key }}\\s*=\\s*(.*)$'
39      line: "{{ item.key }}={{ item.value }}"
40      create: true
41    loop: "{{ lp_tlp_config }}"
42    notify: restart tlp
43    tags: lp_tlp_conf
44
45  - name: "tlp: Start and enable tlp"
46    systemd:
47      name: "{{ item }}"
48      state: started
49      enabled: true
50    loop: "{{ lp_tlp_services }}"
51    when: lp_tlp_enable|bool

```

(continues on next page)

(continued from previous page)

```

52 tags: lp_tlp_service
53
54 - name: "tlp: Stop and disable tlp"
55   systemd:
56     name: "{{ item }}"
57     state: stopped
58     enabled: false
59   loop: "{{ lp_tlp_services }}"
60   when: not lp_tlp_enable|bool
61   tags: lp_tlp_service
62
63 # EOF
64 ...

```

3.1.48 udev.yml

Synopsis: Configure udev.

Description of the task.

[udev.yml]

```

1 ---
2 # linux-postinstall udev
3
4 - name: "udev: Debug"
5   vars:
6     msg: |
7       lp_udev_rules_dir [{{ lp_udev_rules_dir }}]
8       lp_udev_rules_template [{{ lp_udev_rules_template }}]
9       lp_udev_rules
10      {{ lp_udev_rules|to_nice_yaml }}
11
12      lp_udev_persistent_net_template [{{ lp_udev_persistent_net_template }}]
13      lp_udev_persistent_net_rules_file [{{ lp_udev_persistent_net_rules_file }}]
14      lp_udev_persistent_net_rules
15      {{ lp_udev_persistent_net_rules|to_nice_yaml }}
16
17      lp_udev_hci_name_rules_file [{{ lp_udev_hci_name_rules_file }}]
18      lp_udev_hci_name_rules
19      {{ lp_udev_hci_name_rules|to_nice_yaml }}
20
21      lp_udev_hci_run_rules_file [{{ lp_udev_hci_run_rules_file }}]
22      lp_udev_hci_run_rules
23      {{ lp_udev_hci_run_rules|to_nice_yaml }}
24   debug:
25     msg: "{{ msg.split('\n')[:-1] }}"
26   when: lp_udev_debug|bool
27   tags: lp_udev_debug
28
29 # udev rules
30 - name: "udev: Configure {{ lp_udev_rules_dir }}"
31   template:
32     src: "{{ lp_udev_rules_template }}"
33     dest: "{{ lp_udev_rules_dir }}/{{ item.key }}"
34     owner: root

```

(continues on next page)

(continued from previous page)

```

35     group: root
36     mode: "0644"
37     backup: "{{ lp_backup_conf }}"
38     loop: "{{ lp_udev_rules|dict2items }}"
39     notify: reload udev
40     tags: lp_udev_rules
41
42 # persistent_net
43 - name: "udev: Configure {{ lp_udev_rules_dir }}/"
44     {{ lp_udev_persistent_net_rules_file }}"
45     template:
46     src: "{{ lp_udev_persistent_net_template }}"
47     dest: "{{ lp_udev_rules_dir }}/{{ lp_udev_persistent_net_rules_file }}"
48     owner: root
49     group: root
50     mode: "0644"
51     backup: "{{ lp_backup_conf }}"
52     loop: "{{ lp_udev_persistent_net_rules }}"
53     notify: reload udev
54     tags: lp_udev_persistentnet
55
56 # hci name
57 - name: "udev: Configure {{ lp_udev_rules_dir }}/"
58     {{ lp_udev_hci_name_rules_file }}"
59     template:
60     src: hci-name.rules.j2
61     dest: "{{ lp_udev_rules_dir }}/{{ lp_udev_hci_name_rules_file }}"
62     owner: root
63     group: root
64     mode: "0644"
65     backup: "{{ lp_backup_conf }}"
66     loop: "{{ lp_udev_hci_name_rules }}"
67     notify: reload udev
68     tags: lp_udev_hciname
69
70 # hci run
71 - name: "udev: Configure {{ lp_udev_rules_dir }}/"
72     {{ lp_udev_hci_run_rules_file }}"
73     template:
74     src: hci-run.rules.j2
75     dest: "{{ lp_udev_rules_dir }}/{{ lp_udev_hci_run_rules_file }}"
76     owner: root
77     group: root
78     mode: "0644"
79     backup: "{{ lp_backup_conf }}"
80     loop: "{{ lp_udev_hci_run_rules }}"
81     notify: reload udev
82     tags: lp_udev_hcirun
83
84 # Service
85 - name: "udev: Start and enable udev"
86     service:
87     name: "{{ lp_udev_service }}"
88     state: started
89     enabled: true
90     when: lp_udev_enable|bool
91     tags: lp_udev_service

```

(continues on next page)

(continued from previous page)

```

92
93 - name: "udev: Stop and disable udev"
94   service:
95     name: "{{ lp_udev_service }}"
96     state: stopped
97     enabled: false
98   when: not lp_udev_enable|bool
99   tags: lp_udev_service
100
101 # EOF
102 ...

```

3.1.49 ufw.yml

Synopsis: Configure ufw.

Description of the task.

[ufw.yml]

```

1 ---
2 # linux-postinstall ufw
3
4 # Notes
5 #
6 # * Aliases of parameters in ufw module not implemented in task
7 # "Configure ufw".
8 # * It's not necessary to reload ufs after configuration has
9 # changed. Module ufw automatically updates the rules.
10 # * Best practice: First time 'lp_ufw_reset: true'; configure and enable
11 # ufs (configuration item {state: 'enabled'} reloads firewall and
12 # enables firewall on boot); 'lp_ufw_enable: true' start and enable ufw
13 # service.
14 # * Configuration on the fly: configure and enable ufs.
15 # * The last configuration item should be {state: 'enabled'}.
16 # * See: man ufw.
17
18 - name: "ufw: Debug"
19   vars:
20     msg: |
21       lp_ufw_enable [{{ lp_ufw_enable }}]
22       lp_ufw_reset [{{ lp_ufw_reset }}]
23       lp_ufw_reload [{{ lp_ufw_reload }}]
24       lp_ufw_packages
25       {{ lp_ufw_packages|to_nice_yaml }}
26       lp_ufw_conf
27       {{ lp_ufw_conf|to_yaml }}
28   debug:
29     msg: "{{ msg.split('\n')[:-1] }}"
30   when: lp_ufw_debug|bool
31   tags: lp_ufw_debug
32
33 - name: "ufw: Install packages"
34   include_tasks: fn/install-package.yml
35   loop: "{{ lp_ufw_packages }}"
36   tags: lp_ufw_packages

```

(continues on next page)

(continued from previous page)

```

37
38 - name: "ufw: Disable and reset firewall to installation defaults"
39   ufw:
40     state: reset
41   when: lp_ufw_reset|bool
42   tags: lp_ufw_reset
43
44 - name: "ufw: Reload firewall"
45   ufw:
46     state: reloaded
47   when: lp_ufw_reload|bool
48   tags: lp_ufw_reload
49
50 - name: "ufw: Configure ufw"
51   ufw:
52     comment: "{{ item.comment|default(omit) }}"
53     default: "{{ item.default|default(omit) }}"
54     delete: "{{ item.delete|default(omit) }}"
55     direction: "{{ item.direction|default(omit) }}"
56     from_ip: "{{ item.from_ip|default(omit) }}"
57     from_port: "{{ item.from_port|default(omit) }}"
58     insert: "{{ item.insert|default(omit) }}"
59     insert_relative_to: "{{ item.insert_relative_to|default(omit) }}"
60     interface: "{{ item.interface|default(omit) }}"
61     log: "{{ item.log|default(omit) }}"
62     logging: "{{ item.logging|default(omit) }}"
63     name: "{{ item.name|default(omit) }}"
64     proto: "{{ item.proto|default(omit) }}"
65     route: "{{ item.route|default(omit) }}"
66     rule: "{{ item.rule|default(omit) }}"
67     state: "{{ item.state|default(omit) }}"
68     to_ip: "{{ item.to_ip|default(omit) }}"
69     to_port: "{{ item.to_port|default(omit) }}"
70   loop: "{{ lp_ufw_conf }}"
71   tags: lp_ufw_conf
72
73 - name: "ufw: Start and enable ufw"
74   service:
75     name: "{{ lp_ufw_service }}"
76     state: started
77     enabled: true
78   register: result
79   when: lp_ufw_enable|bool
80   tags: lp_ufw_service
81
82 - name: "ufw: Debug enabled service"
83   debug:
84     var: result
85   when:
86     - lp_ufw_enable|bool
87     - lp_ufw_debug|bool
88   tags: lp_ufw_service
89
90 - name: "ufw: Stop and disable ufw"
91   service:
92     name: "{{ lp_ufw_service }}"
93     state: stopped

```

(continues on next page)

(continued from previous page)

```

94     enabled: false
95     register: result
96     when: not lp_ufw_enable|bool
97     tags: lp_ufw_service
98
99 - name: "ufw: Debug disabled service"
100   debug:
101     var: result
102     when:
103       - not lp_ufw_enable|bool
104       - lp_ufw_debug|bool
105     tags: lp_ufw_service
106
107 # EOF
108 ...

```

3.1.50 users.yml

Synopsis: Configure users.

Description of the task.

[users.yml]

```

1  ---
2  # linux-postinstall users
3
4  - name: "users: Debug"
5    vars:
6      msg: |
7        lp_users
8        {{ lp_users|default(['UNDEFINED'])|to_nice_yaml }}
9        lp_users_groups
10       {{ lp_users_groups|default(['UNDEFINED'])|to_nice_yaml }}
11    debug:
12      msg: "{{ msg.split('\n')[:-1] }}"
13    when: lp_users_debug|bool
14    tags: lp_users_debug
15
16 - name: "users: Manage user accounts"
17   user:
18     name: "{{ item.name }}"
19     authorization: "{{ item.authorization|default(omit) }}"
20     comment: "{{ item.comment|default(omit) }}"
21     create_home: "{{ item.create_home|default(omit) }}"
22     expires: "{{ item.expires|default(omit) }}"
23     force: "{{ item.force|default(omit) }}"
24     generate_ssh_key: "{{ item.generate_ssh_key|default(omit) }}"
25     group: "{{ item.group|default(omit) }}"
26     hidden: "{{ item.hidden|default(omit) }}"
27     home: "{{ item.home|default(omit) }}"
28     local: "{{ item.local|default(omit) }}"
29     login_class: "{{ item.login_class|default(omit) }}"
30     move_home: "{{ item.move_home|default(omit) }}"
31     non_unique: "{{ item.non_unique|default(omit) }}"
32     password: "{{ item.password|default(omit) }}"

```

(continues on next page)

(continued from previous page)

```

33 password_lock: "{{ item.password_lock|default(omit) }}"
34 profile: "{{ item.profile|default(omit) }}"
35 remove: "{{ item.remove|default(omit) }}"
36 role: "{{ item.role|default(omit) }}"
37 seuser: "{{ item.seuser|default(omit) }}"
38 shell: "{{ item.shell|default(omit) }}"
39 skeleton: "{{ item.skeleton|default(omit) }}"
40 ssh_key_bits: "{{ item.ssh_key_bits|default(omit) }}"
41 ssh_key_comment: "{{ item.ssh_key_comment|default(omit) }}"
42 ssh_key_file: "{{ item.ssh_key_file|default(omit) }}"
43 ssh_key_passphrase: "{{ item.ssh_key_passphrase|default(omit) }}"
44 ssh_key_type: "{{ item.ssh_key_type|default(omit) }}"
45 state: "{{ item.state|default(omit) }}"
46 system: "{{ item.system|default(omit) }}"
47 uid: "{{ item.uid|default(omit) }}"
48 update_password: "{{ item.update_password|default(omit) }}"
49 loop: "{{ lp_users|default([]) }}"
50 loop_control:
51   label: "{{ item.name }}"
52 tags: lp_users_accounts
53
54 - name: "users: Add users to additional groups"
55   user:
56     name: "{{ item.name }}"
57     groups: "{{ item.groups }}"
58     append: "{{ item.append|default(true) }}"
59     loop: "{{ lp_users_groups|default([]) }}"
60     tags: lp_users_groups
61
62 # EOF
63 ...

```

3.1.51 vars.yml

Synopsis: Configure vars.

Description of the task.

[vars.yml]

```

1 ---
2 # linux-postinstall vars
3
4 - name: "vars: Include firstfound default vars"
5   when: lp_vars_distro == "firstfound"
6   include_tasks: vars-firstfound.yml
7
8 - name: "vars: Include incremental default vars"
9   when: lp_vars_distro == "incremental"
10  include_tasks: vars-firstfound.yml
11
12 - name: "vars: Include firstfound default vars for various flavors"
13   when: lp_flavors_enable|bool
14   include_tasks: sub/vars-flavors.yml
15
16 # TODO "vars: Include incremental default vars for various flavors"

```

(continues on next page)

(continued from previous page)

```
17 # EOF
18 ...
19
```

3.1.52 vars-firstfound.yml

Synopsis: Configure OS specific default vars. Method first_found.

Description of the task.

[vars-firstfound.yml]

```
1 ---
2 # linux-postinstall vars-firstfound
3
4 - name: "vars-firstfound: Include default vars for
5     [{{ ansible_distribution_release }},
6     [{{ ansible_distribution }},
7     [{{ ansible_os_family }}}]"
8   include_vars: "{{ lookup('first_found', params) }}"
9   register: result
10  vars:
11    params:
12      files:
13        - "{{ ansible_distribution }}-{{ ansible_distribution_release }}.yaml"
14        - "{{ ansible_distribution }}.yaml"
15        - "{{ ansible_os_family }}.yaml"
16        - default.yaml
17        - defaults.yaml
18      paths:
19        - "{{ role_path }}/vars/defaults"
20  # [TODO]
21  # skip: "{{ lp_vars_distro_firstfound_skip|bool }}"
22  # skip doesn't work with first_found lookup #43833
23  # https://github.com/ansible/ansible/issues/43833
24  # workaround: Create empty defaults.yaml
25
26 - name: "vars-firstfound: Debug include default vars from"
27   debug:
28     var: result.ansible_included_var_files
29   when: lp_debug|bool
30
31 - name: "vars-firstfound: Include custom vars for
32     [{{ ansible_distribution_release }},
33     [{{ ansible_distribution }},
34     [{{ ansible_os_family }}}]"
35   register: result
36   include_vars: "{{ lookup('first_found', params) }}"
37   vars:
38     params:
39       files:
40        - "{{ ansible_distribution }}-{{ ansible_distribution_release }}.yaml"
41        - "{{ ansible_distribution }}.yaml"
42        - "{{ ansible_os_family }}.yaml"
43        - default.yaml
44        - defaults.yaml
```

(continues on next page)

(continued from previous page)

```

45     paths:
46     - "{{ role_path }}/vars"
47 # [TODO]
48 #     skip: "{{ lp_vars_distro_firstfound_skip|bool }}"
49 # skip doesn't work with first_found lookup #43833
50 # https://github.com/ansible/ansible/issues/43833
51 # workaround: Create empty defaults.yml
52
53 - name: "vars-firstfound: Debug include custom vars from"
54   debug:
55     var: result.ansible_included_var_files
56   when: lp_debug|bool
57
58 # EOF
59 ...

```

3.1.53 vars-incremental.yml

Synopsis: Configure OS specific default vars. Method incremental.

Description of the task.

[vars-incremental.yml]

```

1  ---
2  # linux-postinstall vars-incremental
3
4  - name: "vars-incemental: Include default vars for
5      [{{ ansible_os_family }},
6      {{ ansible_distribution }},
7      {{ ansible_distribution_release }}}]"
8    include_vars: "{{ item }}"
9    register: result
10   loop:
11     - "{{ my_path }}/defaults.yml"
12     - "{{ my_path }}/default.yml"
13     - "{{ my_path }}/{{ ansible_os_family }}.yml"
14     - "{{ my_path }}/{{ ansible_distribution }}.yml"
15     - "{{ my_path }}/{{ ansible_distribution }}-{{ ansible_distribution_release }}.yml"
16   ↪
17   when: item is exists
18   vars:
19     my_path: "{{ role_path }}/vars/defaults.incr"
20
21 - name: "vars-incemental: Debug include default vars"
22   debug:
23     var: result
24   when: lp_debug|bool
25
26 - name: "vars-incemental: Include custom vars for
27     [{{ ansible_os_family }},
28     {{ ansible_distribution }},
29     {{ ansible_distribution_release }}}]"
30   include_vars: "{{ item }}"
31   register: result
32   loop:

```

(continues on next page)

(continued from previous page)

```

32 - "{{ my_path }}/defaults.yml"
33 - "{{ my_path }}/default.yml"
34 - "{{ my_path }}/{{ ansible_os_family }}.yml"
35 - "{{ my_path }}/{{ ansible_distribution }}.yml"
36 - "{{ my_path }}/{{ ansible_distribution }}-{{ ansible_distribution_release }}.yml
↪ "
37 when: item is exists
38 vars:
39     my_path: "{{ role_path }}/vars"
40
41 - name: "vars-incemental: Debug include custom vars"
42   debug:
43     var: result
44   when: lp_debug|bool
45
46 # EOF
47 ...

```

3.1.54 virtualbox.yml

Synopsis: Configure virtualbox.

Description of the task.

[virtualbox.yml]

```

1 ---
2 # linux-postinstall virtualbox
3
4 - name: "virtualbox: Debug"
5   vars:
6     msg: |
7       ansible_lsb.description [{{ ansible_lsb.codename }}]
8       lp_virtualbox [{{ lp_virtualbox }}]
9       lp_virtualbox_ignore_errors [{{ lp_virtualbox_ignore_errors }}]
10      lp_virtualbox_keys [{{ lp_virtualbox_keys }}]
11      lp_virtualbox_repos [{{ lp_virtualbox_repos }}]
12      lp_virtualbox_install
13      {{ lp_virtualbox_install|to_nice_yaml }}
14      lp_virtualbox_services
15      {{ lp_virtualbox_services|to_nice_yaml }}
16   debug:
17     msg: "{{ msg.split('\n')[:-1] }}"
18   when: lp_virtualbox_debug|bool
19   tags: lp_virtualbox_debug
20
21 # TODO: assert lp_virtualbox_modules are loaded
22 #       when: lp_virtualbox|bool
23
24 - name: "virtualbox: Add signing key of VirtualBox"
25   apt_key:
26     url: "{{ item }}"
27     state: present
28   loop: "{{ lp_virtualbox_keys }}"
29   register: result
30   retries: "{{ lp_install_retries }}"

```

(continues on next page)

(continued from previous page)

```

31  until: result is succeeded
32  delay: "{{ lp_install_delay }}"
33  ignore_errors: "{{ lp_virtualbox_ignore_errors }}"
34  tags: lp_virtualbox_keys
35
36  - name: "virtualbox: Add repository of VirtualBox"
37    apt_repository:
38      repo: "{{ item }}"
39      state: present
40    loop: "{{ lp_virtualbox_repos }}"
41    ignore_errors: "{{ lp_virtualbox_ignore_errors }}"
42    tags: lp_virtualbox_repos
43
44  - name: "virtualbox: Install VirtualBox packages"
45    include_tasks: fn/install-package.yml
46    loop: "{{ lp_virtualbox_packages }}"
47    ignore_errors: "{{ lp_virtualbox_ignore_errors }}"
48    tags: lp_virtualbox_pkg
49
50  - name: "virtualbox: Enable and start services"
51    service:
52      name: "{{ item }}"
53      state: started
54      enabled: true
55    loop: "{{ lp_virtualbox_services }}"
56    when: lp_virtualbox_enable|bool
57    tags: lp_virtualbox_services
58
59  - name: "virtualbox: Disable and stop services"
60    service:
61      name: "{{ item }}"
62      state: stopped
63      enabled: false
64    loop: "{{ lp_virtualbox_services }}"
65    when: not lp_virtualbox_enable|bool
66    tags: lp_virtualbox_services
67
68  # EOF
69  ...

```

3.1.55 wpagui.yml

Synopsis: Configure wpagui.

Description of the task.

[wpagui.yml]

```

1  ---
2  # linux-postinstall wpa_gui
3  # Install wpa_gui and disable NetworkManager
4
5  - name: "wpagui: Debug"
6    vars:
7      msg: |
8        lp_wpagui_packages

```

(continues on next page)

(continued from previous page)

```

9      {{ lp_wpagui_packages|to_nice_yaml }}
10      lp_wpagui_systemd
11      {{ lp_wpagui_systemd|to_nice_yaml }}
12      lp_wpagui_service
13      {{ lp_wpagui_service|to_nice_yaml }}
14      lp_wpagui_service_mask
15      {{ lp_wpagui_service_mask|to_nice_yaml }}
16  debug:
17      msg: "{{ msg.split('\n')[:-1] }}"
18  when: lp_wpagui_debug|bool
19  tags: lp_wpagui_debug
20
21  - name: "wpagui: Install packages"
22    include_tasks: fn/install-package.yml
23    loop: "{{ lp_wpagui_packages }}"
24    tags: lp_wpagui_packages
25
26  - name: "wpagui: Disable NM /etc/init/network-manager.override"
27    template:
28      src: network-manager-override.j2
29      dest: /etc/init/network-manager.override
30      owner: root
31      group: root
32      mode: "0644"
33    tags: lp_wpagui_disableNM
34
35  - name: "wpagui: Configure managed=false
36    in /etc/NetworkManager/NetworkManager.conf"
37    lineinfile:
38      dest: /etc/NetworkManager/NetworkManager.conf
39      regexp: '^\\s*managed\\s*=\\s*(.*)$'
40      line: "managed=false"
41    tags: lp_wpagui_disableNM
42
43  # NetworkManager.service will be stopped and disabled in the next task
44  # - name: "wpagui: Stop and disable NM"
45  #   systemd:
46  #     name: "{{ item }}"
47  #     state: stopped
48  #     enabled: false
49  #   loop: "{{ lp_wpagui_systemd }}"
50  #   tags: lp_wpagui_disableNM
51
52  - name: "wpagui: Stop and disable all NM services"
53    service:
54      name: "{{ item }}"
55      state: stopped
56      enabled: false
57    loop: "{{ lp_wpagui_service }}"
58    tags: lp_wpagui_disableNM
59
60  - name: "wpagui: Mask NM services"
61    command: "systemctl mask {{ item }}"
62    args:
63      warn: false
64    loop: "{{ lp_wpagui_service_mask }}"
65    changed_when: false

```

(continues on next page)

(continued from previous page)

```

66 tags: lp_wpagui_mask_NM
67 #
68 # False Positives: Skipping Rules
69 # https://github.com/ansible/ansible-lint#false-positives-skipping-rules
70 # noqa 303 does not work
71 #
72 # ansible-lint "systemctl used in place of systemd module" No
73 # systemctl / systemd module. #48848
74 # https://github.com/ansible/ansible/issues/48848
75
76 # EOF
77 ...

```

3.1.56 wpasupplicant.yml

Synopsis: Configure wpasupplicant.

Description of the task.

[wpasupplicant.yml]

```

1 ---
2 # linux-postinstall wpasupplicant
3
4 - name: "wpasupplicant: Debug"
5   vars:
6     msg: |
7       lp_package_state [{{ lp_package_state }}]
8       lp_wpasupplicant_packages
9       {{ lp_wpasupplicant_packages|to_nice_yaml }}
10      lp_wpasupplicant_conf_only [{{ lp_wpasupplicant_conf_only }}]
11      lp_wpasupplicant_conf_dir [{{ lp_wpasupplicant_conf_dir }}]
12      lp_wpasupplicant_conf_file [{{ lp_wpasupplicant_conf_file }}]
13      lp_wpasupplicant_conf_owner [{{ lp_wpasupplicant_conf_owner }}]
14      lp_wpasupplicant_conf_group [{{ lp_wpasupplicant_conf_group }}]
15      lp_wpasupplicant_conf_mode [{{ lp_wpasupplicant_conf_mode }}]
16      lp_wpasupplicant_conf_ctrl_interface
17      [{{ lp_wpasupplicant_conf_ctrl_interface }}]
18      lp_wpasupplicant_conf_global
19      {{ lp_wpasupplicant_conf_global|to_yaml }}
20      lp_wpa_action_script [{{ lp_wpa_action_script }}]
21      lp_wpa_action_script_dir [{{ lp_wpa_action_script_dir }}]
22      lp_wpa_action_script_file [{{ lp_wpa_action_script_file }}]
23      lp_wpa_action_script_owner [{{ lp_wpa_action_script_owner }}]
24      lp_wpa_action_script_group [{{ lp_wpa_action_script_group }}]
25      lp_wpa_action_script_mode [{{ lp_wpa_action_script_mode }}]
26      lp_wpa_action_script_dhclient [{{ lp_wpa_action_script_dhclient }}]
27      lp_wpa_action_script_pidfile [{{ lp_wpa_action_script_pidfile }}]
28      lp_wpa_action_script_options_connect [{{ lp_wpa_action_script_options_connect }}]
29      lp_wpa_action_script_options_disconnect [{{ lp_wpa_action_script_options_
30      ↪disconnect }}]
31      lp_wpa_action_script_logfile [{{ lp_wpa_action_script_logfile }}]
32
33      {% if lp_wpasupplicant_debug_classified|bool %}
34      lp_wpasupplicant_conf

```

(continues on next page)

(continued from previous page)

```

34     {{ lp_wpasupplicant_conf|to_yaml }}
35     {% endif %}
36 debug:
37     msg: "{{ msg.split('\n')[:-1] }}"
38 when: lp_wpasupplicant_debug|bool
39 tags: lp_wpasupplicant_debug
40
41 - name: "wpasupplicant: Install packages"
42   include_tasks: fn/install-package.yml
43   loop: "{{ lp_wpasupplicant_packages }}"
44   tags: lp_wpasupplicant_packages
45
46 - name: "wpasupplicant: Configure {{ lp_wpasupplicant_conf_dir }}/{{
47     lp_wpasupplicant_conf_file }}.DEV"
48   template:
49     src: wpa_supplicant.conf.j2
50     dest: "{{ lp_wpasupplicant_conf_dir }}/{{
51         lp_wpasupplicant_conf_file }}.{{
52         item.dev }}"
53     owner: "{{ lp_wpasupplicant_conf_owner }}"
54     group: "{{ lp_wpasupplicant_conf_group }}"
55     mode: "{{ lp_wpasupplicant_conf_mode }}"
56     backup: "{{ lp_backup_conf }}"
57   register: lp_wpasupplicant_conf_changes
58   notify: reconfigure wpa_supplicant
59   loop: "{{ lp_wpasupplicant_conf }}"
60   loop_control:
61     label: "{{ item.dev }}"
62   no_log: "{{ not lp_wpasupplicant_debug_classified }}"
63   tags: lp_wpasupplicant_conf
64
65 - name: "wpasupplicant: Debug: registered lp_wpasupplicant_conf_changes"
66   debug:
67     var: lp_wpasupplicant_conf_changes
68   when: lp_wpasupplicant_debug_classified|bool
69
70 # - name: "wpasupplicant: Debug: wpa_cli reconfigure commands"
71 #   debug:
72 #     msg: >
73 #       'sh -c "[ -S {{ lp_wpasupplicant_conf_ctrl_interface }}/\
74 #           {{ item.item.dev }} ] &&
75 #       wpa_cli -p {{ lp_wpasupplicant_conf_ctrl_interface }} \
76 #           -i {{ item.item.dev }} reconfigure"'
77 #   loop: "{{ lp_wpasupplicant_conf_changes.results }}"
78 #   when:
79 #     - item.changed
80 #     - lp_wpasupplicant_debug
81
82 - name: "wpasupplicant: Create dir {{ lp_wpa_action_script_dir }}"
83   file:
84     state: directory
85     path: "{{ lp_wpa_action_script_dir }}"
86     owner: "{{ lp_wpa_action_script_owner }}"
87     group: "{{ lp_wpa_action_script_group }}"
88   when: lp_wpa_action_script|bool
89   tags: lp_wpa_action_script_dir
90

```

(continues on next page)

(continued from previous page)

```

91 - name: "wpasupplicant: Create script {{ lp_wpa_action_script_file }}"
92   template:
93     src: wpa_action.sh.j2
94     dest: "{{ lp_wpa_action_script_file }}"
95     owner: "{{ lp_wpa_action_script_owner }}"
96     group: "{{ lp_wpa_action_script_group }}"
97     mode: "{{ lp_wpa_action_script_mode }}"
98     backup: "{{ lp_backup_conf }}"
99   when: lp_wpa_action_script|bool
100   tags: lp_wpa_action_script_file
101
102 # EOF
103 ...

```

3.1.57 xen.yml

Synopsis: Configure xen.

Description of the task.

[xen.yml]

```

1 ---
2 # linux-postinstall xen
3
4 - name: "xen: Debug"
5   vars:
6     msg: |
7       lp_xen_packages
8       {{ lp_xen_packages|to_nice_yaml }}
9       lp_xen_dom0_mem
10      {{ lp_xen_dom0_mem|to_nice_yaml }}
11      lp_xen_default_grub_conf
12      {{ lp_xen_default_grub_conf|to_nice_yaml }}
13      lp_xen_global
14      {{ lp_xen_global|to_nice_yaml }}
15   debug:
16     msg: "{{ msg.split('\n')[:-1] }}"
17   when: lp_xen_debug|bool
18   tags: lp_xen_debug
19
20 - name: "xen: Install packages"
21   include_tasks: fn/install-package.yml
22   loop: "{{ lp_xen_packages }}"
23   tags: lp_xen_packages
24
25 - name: "xen: Configure /etc/default/grub"
26   lineinfile:
27     dest: /etc/default/grub
28     regexp: '^\\s*{{ item.key }}\\s*=\\s*(.*)$'
29     line: "{{ item.key }}={{ item.value }}"
30     backup: "{{ lp_backup_conf }}"
31     loop: "{{ lp_xen_default_grub_conf }}"
32     notify: update grub
33     tags: lp_xen_default_grub
34

```

(continues on next page)

(continued from previous page)

```

35 - name: "xen: Configure /etc/xen/xl.conf"
36   lineinfile:
37     dest: /etc/xen/xl.conf
38     regexp: '^\\s*{{ item.var }}\\s*=\\s*(.*)$'
39     line: "{{ item.var }}={{ item.value }}"
40     create: true
41     backup: "{{ lp_backup_conf }}"
42     loop: "{{ lp_xen_global }}"
43     tags: lp_xen_global
44
45 # EOF
46 ...

```

3.1.58 xorg.yml

Synopsis: Configure xorg.

Description of the task.

[xorg.yml]

```

1 ---
2 # linux-postinstall xen
3
4 - name: "xorg: Debug"
5   vars:
6     msg: |
7       lp_xorg_conf
8       {{ lp_xorg_conf|to_nice_yaml }}
9   debug:
10     msg: "{{ msg.split('\\n')[:-1] }}"
11   when: lp_xorg_debug|bool
12   tags: lp_xorg_debug
13
14 - name: "xorg: Configure {{ lp_xorg_conf_dir }}"
15   template:
16     src: xorg.conf.j2
17     dest: "{{ lp_xorg_conf_dir }}/{{ item.file }}"
18     backup: "{{ lp_backup_conf }}"
19     loop: "{{ lp_xorg_conf }}"
20     tags: lp_xorg_conf
21
22 # EOF
23 ...

```

3.1.59 zeitgeist.yml

Synopsis: Configure zeitgeist.

Description of the task.

[zeitgeist.yml]

```

1  ---
2  # linux-postinstall zeitgeist
3
4  # One-way atm
5  - name: Remove zeitgeist
6    apt:
7      state: absent
8      name: zeitgeist
9      purge: true
10   when:
11     - not lp_zeitgeist|bool
12     - ansible_os_family == "Debian"
13
14  - name: Remove zeitgeist-*
15    apt:
16      state: absent
17      name: zeitgeist-*
18      purge: true
19   when:
20     - not lp_zeitgeist|bool
21     - ansible_os_family == "Debian"
22
23  # - name: Disable zeitgeist
24  #   service:
25  #     name: zeitgeist
26  #     state: stopped
27  #     enabled: no
28  #   when: not lp_zeitgeist
29  # "Could not find the requested service zeitgeist"
30
31  # for i in zeitgeist-fts zeitgeist; do
32  #   systemctl --user disable $i;
33  #   systemctl --user stop $i;
34  #   systemctl --user mask $i;
35  # done
36
37  # EOF
38  ...

```

3.1.60 zfs.yml

Synopsis: Configure zfs.

Description of the task.

[zfs.yml]

```

1  ---
2  # linux-postinstall zfs
3
4  - name: "zfs: Debug"
5    vars:
6      msg: |
7        lp_zfs_install [{{ lp_zfs_install }}]
8        lp_zfs_packages
9        {{ lp_zfs_packages|to_nice_yaml }}

```

(continues on next page)

(continued from previous page)

```

10     lp_zfs_services
11     {{ lp_zfs_services|default([])|to_yaml }}
12     lp_zfs_manage
13     {{ lp_zfs_manage|to_yaml }}
14     lp_zfs_mountpoints
15     {{ lp_zfs_mountpoints|to_yaml }}
16 debug:
17     msg: "{{ msg.split('\n')[:-1] }}"
18 when: lp_zfs_debug|bool
19 tags: lp_zfs_debug
20
21 - name: "zfs: Install packages"
22   include_tasks: fn/install-package.yml
23   loop: "{{ lp_zfs_packages }}"
24   when: lp_zfs_install|bool
25   tags: lp_zfs_packages
26
27 - name: "zfs: Manage zfs services"
28   systemd:
29     name: "{{ item.name }}"
30     enabled: "{{ item.enabled|default(true) }}"
31     state: "{{ item.state|default('started') }}"
32   loop: "{{ lp_zfs_services|default([]) }}"
33   tags: lp_zfs_services
34
35 - name: "zfs: Manage zfs"
36   zfs:
37     name: "{{ item.name }}"
38     state: "{{ item.state }}"
39     origin: "{{ item.origin|default(omit) }}"
40     extra_zfs_properties: "{{ item.extra_zfs_properties|default(omit) }}"
41   loop: "{{ lp_zfs_manage }}"
42   tags: lp_zfs_manage
43
44 - name: "zfs: Set mode and ownership of zfs mountpoints"
45   file:
46     state: directory
47     path: "{{ item.mountpoint }}"
48     owner: "{{ item.owner|default(omit) }}"
49     group: "{{ item.group|default(omit) }}"
50     mode: "{{ item.mode|default(omit) }}"
51   loop: "{{ lp_zfs_mountpoints }}"
52   tags: lp_zfs_mountpoints
53
54 # EOF
55 ...

```

3.1.61 sub/vars-flavors.yml

Synopsis: Configure flavor specific variables.

Description of the task.

[sub/vars-flavors.yml]


```

1  ---
2  # linux-postinstall vars: vars-flavors
3
4  # Create dir lp_flavors_dir. Loop lp_flavors and get stat of
5  # release_file(s). If release_file exists include tasks specific to this
6  # flavor.
7
8  - name: 'sub: vars-flavors: Debug'
9    vars:
10     msg: |
11       lp_flavors_dir [{{ lp_flavors_dir }}]
12       lp_flavors_dir_owner [{{ lp_flavors_dir_owner }}]
13       lp_flavors_dir_group [{{ lp_flavors_dir_group }}]
14       lp_flavors_dir_mode [{{ lp_flavors_dir_mode }}]
15       lp_flavors
16       {{ lp_flavors|to_nice_yaml }}
17     debug:
18       msg: "{{ msg.split('\n')[:-1] }}"
19     when: lp_debug|bool
20
21 - name: 'sub: vars-flavors: Create {{ lp_flavors_dir }}'
22   file:
23     state: directory
24     path: '{{ lp_flavors_dir }}'
25     owner: '{{ lp_flavors_dir_owner }}'
26     group: '{{ lp_flavors_dir_group }}'
27     mode: '{{ lp_flavors_dir_mode }}'
28   delegate_to: localhost
29   run_once: true
30
31 - name: 'sub: vars-flavors: Detect flavor'
32   stat:
33     path: '{{ item.value.release_file }}'
34     loop: '{{ lp_flavors|dict2items }}'
35     register: result
36
37 - name: 'sub: vars-flavors: Debug result'
38   when: lp_debug|bool
39   debug:
40     msg: "{{ result.results|json_query('[?stat.exists].item') }}"
41
42 - name: 'sub: vars-flavors: Include tasks for flavor'
43   include_tasks: "{{ 'sub/vars-flavors-' ~ outer_item.key ~ '.yaml' }}"
44   loop: "{{ result.results|json_query('[?stat.exists].item') }}"
45   loop_control:
46     loop_var: outer_item
47
48 # EOF
49 ...

```

3.1.62 sub/vars-flavors-common.yml

Synopsis: Configure common flavor specific variables.

Description of the task.

[sub/vars-flavors-common.yml]

```

1  ---
2  # linux-postinstall vars: vars-flavors-common
3
4  # Fetch my_release_file from the remote host and store the file in
5  # lp_flavors_dir. Read release_attr from the fetched file and include
6  # vars that correspond the flavor, release and HW.
7
8  # my_release_file
9  - name: 'sub: vars-flavors-common: Set my_release_file'
10   set_fact:
11     my_release_file: '{{ outer_item.value.release_file }}'
12  - name: 'sub: vars-flavors-common: Debug my_release_file'
13   when: lp_debug|bool
14   debug:
15     var: my_release_file
16
17  # my_flavor
18  - name: 'sub: vars-flavors-common: Set my_flavor'
19   set_fact:
20     my_flavor: '{{ outer_item.key }}'
21  - name: 'sub: vars-flavors-common: Debug my_flavor'
22   when: lp_debug|bool
23   debug:
24     var: my_flavor
25
26  # my_release_file_fetch
27  - name: 'sub: vars-flavors-common: Set my_release_file_fetch'
28   set_fact:
29     my_release_file_fetch: "{{ lp_flavors_dir ~ '/' ~
30                               inventory_hostname ~ '-' ~
31                               my_flavor }}"
32  - name: 'sub: vars-flavors-common: Fetch {{ my_release_file }} to
33    {{ my_release_file_fetch }}'
34   fetch:
35     flat: true
36     src: '{{ my_release_file }}'
37     dest: '{{ my_release_file_fetch }}'
38
39  # my_release_keys
40  - name: 'sub: vars-flavors-common: Clear my_release_keys'
41   set_fact:
42     my_release_keys: []
43  - name: 'sub: vars-flavors-common: Set my_release_keys'
44   set_fact:
45     my_release_keys: "{{ my_release_keys|
46                           default([]) + [item.split('=')[0]|trim] }}"
47   loop: "{{ lookup('file', my_release_file_fetch).splitlines() }}"
48   when: item is match('^(\\s*[a-zA-Z0-9_]+\\s*)=(.*)$')
49  - name: 'sub: vars-flavors-common: Debug my_release_keys'
50   when: lp_debug|bool
51   debug:
52     var: my_release_keys
53
54  # my_release_dict
55  - name: 'sub: vars-flavors-common: Clear my_release_dict'
56   set_fact:
57     my_release_dict: {}

```

(continues on next page)

(continued from previous page)

```

58 - name: 'sub: vars-flavors-common: Set my_release_dict attributes'
59   set_fact:
60     my_release_dict: "{{ my_release_dict|
61                          combine({item: lookup('ini', item ~
62                                                ' type=properties file=' ~
63                                                my_release_file_fetch)}) }}"
64   loop: '{{ my_release_keys }}'
65 - name: 'sub: vars-flavors-common: Debug my_release_dict'
66   when: lp_debug|bool
67   debug:
68     var: my_release_dict
69
70 # my_release
71 - name: 'sub: vars-flavors-common: Add flavor to my_release'
72   set_fact:
73     my_release: '{{ my_release|
74                    default({})|
75                    combine({my_flavor: my_release_dict}) }}'
76 - name: 'sub: vars-flavors-common: Debug my_release'
77   when: lp_debug|bool
78   debug:
79     var: my_release
80
81 # my_labels
82 - name: 'sub: vars-flavors-common: Set my_labels'
83   set_fact:
84     my_labels: "{{ lp_flavors[my_flavor].file_labels|
85                   map('extract', my_release[my_flavor])|
86                   list }}"
87 - name: 'sub: vars-flavors-common: Debug my_labels'
88   when: lp_debug|bool
89   debug:
90     var: my_labels
91
92 # Include default vars for flavor
93 - name: 'sub: vars-flavors-common: Include default vars for flavor
94     [{{ my_labels.1 }},
95     {{ my_labels.0 }},
96     {{ my_flavor }}]'
97   include_vars: "{{ lookup('first_found', params) }}"
98   vars:
99     params:
100       files:
101         - '{{ my_flavor }}_{{ my_labels.0 }}_{{ my_labels.1 }}.yaml'
102         - '{{ my_flavor }}_{{ my_labels.0 }}.yaml'
103         - '{{ my_flavor }}.yaml'
104         - default.yaml
105         - defaults.yaml
106       paths:
107         - '{{ role_path }}/vars/flavors'
108 # [TODO]
109 #   skip: "{{ lp_vars.distro_firstfound_skip|bool }}"
110 # skip doesn't work with first_found lookup #43833
111 # https://github.com/ansible/ansible/issues/43833
112 # workaround: Create empty defaults.yaml
113
114 # EOF
    
```

(continues on next page)

(continued from previous page)

115

...

COPYRIGHT

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

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

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

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

LEGAL NOTICE

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

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

Symbols

`__main__`
 module, 12

M

module
 `__main__`, 12
 search path, 12
 sys, 12

P

password
 passwordstore, 12, 16
path
 module search, 12

S

search
 path, module, 12
sys
 module, 12