DebOps Contrib Documentation

Release master

DebOps Contrib Collective

Sep 03, 2017

Contents

Documentation of DebOps Contrib playbooks and roles. DebOps Contrib components are no official part of DebOps, but might be migrated to it in the future.

You can find all DebOps Contrib components under: https://github.com/debops-contrib/ Note that not every DebOps Contrib component has a corresponding documentation appearing here yet.

CHAPTER 1

DebOps Contrib

Additional Ansible roles of the DebOps project

Welcome. This organization is meant to hold the Ansible roles and playbooks that are not part of the official DebOps project, but might be integrated with it in the future.

If you have any roles that you would like to add here, you can either create an issue in this (`debops-contrib`) repository or contact the team at `#debops` IRC channel on FreeNode, or through the mailing list.

You should be able to import your role to Ansible Galaxy under the debops-contrib organization. The role would then be called `debops-contrib.$your_role` . Remember to rename your role appropriately and generate a new README indicating its correct name.

Example: debops-contrib.checkmk_server

DebOps Contrib Playbooks

As for the official DebOps project, DebOps Contrib also has a repository holding playbooks for the roles. The repository is called debops-contrib-playbooks.

CHAPTER 2

Adding roles to the documentation

If you maintain a DebOps Contrib role, you are encouraged to add your role to this end user documentation.

For this, check how DebOps does documentation by looking at debops/docs and for examples you can checkout an up-to-date role from the DebOps Status page. The links defined in DebOps docs global.rst can also be used for DebOps Contribs roles. The file gets injected into the docs build the same way as for DebOps itself. If you have additional links which fit into the file, you can add them to debops/docs and your changes will also be available here. Refer to Use global link definitions for details.

The `README.md` file which is used for GitHub and Ansible Galaxy can be generated using ansigenome and templates currently available here: https://github.com/ypid/ypid-ansible-common/tree/master/template_READMEs

For bonus points, setup Travis CI tests for your role and import it on Ansible Galaxy.

Feel free to add your role to: https://github.com/debops-contrib/docs You can add a role by running **`./add_new_role checkmk_server`** (replace `checkmk_server` with the name of the role). When you are member of the DebOps Contrib organization you should have write permissions to the repository and can merge your own pull request after the test for the PR passed.

If you push new commits to your role, this documentation should pick them up within two hours without further intervention.

CHAPTER 3

Ansible roles provided in DebOps Contrib

Ansible role: debops-contrib.apparmor

Introduction

AppArmor is able to restrict what programs can do and access based on policies for those programs.

See AppArmor in the Debian Wiki.

By default (e.g. no auditd installed) log messages from AppArmor are logged via syslog to the kernel facility which usually ends up under `/var/log/kern.log`.

Installation

This role requires at least Ansible `v2.1.3`. To install it, run:

```
ansible-galaxy install debops-contrib.apparmor
```

Getting started

Example inventory

To install and configure AppArmor, add the hosts to the `debops_service_apparmor` Ansible inventory host group:

```
[debops_service_apparmor]
hostname
```

Example playbook

Here's an example playbook that can be used to install and configure AppArmor:

```
---

- name: Install and configure AppArmor
  hosts: [ 'debops_service_apparmor' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops.grub
      grub_dependent_kernel_options: '{{ apparmor__kernel_options }}'
      when: (not (apparmor__manage_grub|d() | bool))
      tags: [ 'role::grub' ]

    - role: debops-contrib.apparmor
      tags: [ 'role::apparmor' ]
```

The playbooks is shipped with this role under `docs/playbooks/apparmor.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::apparmor` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

debops-contrib.apparmor default variables

Sections

- *Packages and installation*
- *AppArmor profiles*

Packages and installation

`apparmor__base_packages`

List of base packages to install.

```
apparmor__base_packages:
  - 'apparmor'
  - 'apparmor-utils'
  - 'apparmor-profiles'
  - '{{ []
        if (ansible_distribution == "Ubuntu" and not (ansible_distribution_
↪version|version_compare("15.10", ">=")))
        else [ "apparmor-profiles-extra" ] }}'
```

apparmor__packages

List of additional packages to install with AppArmor.

Example:

```
apparmor__packages:
  - 'apparmor-notify'
```

```
apparmor__packages: []
```

apparmor__enabled

Should AppArmor be enabled?

```
apparmor__enabled: True
```

apparmor__kernel_options

Default kernel options needed to enable AppArmor. You probably don't need to change this.

```
apparmor__kernel_options:
  - 'apparmor=1'
  - 'security=apparmor'
```

apparmor__manage_grub

How should role write the required kernel options into the Grub configuration. The default is delegate this to the debops.grub role. If set to False, this role will do that internally without debops.grub. Note that this role is not as flexible in configuring Grub as debops.grub is.

```
apparmor__manage_grub: False
```

apparmor__additional_kernel_parameters

Legacy: Only considered when *apparmor__manage_grub == True*.

```
apparmor__additional_kernel_parameters: ''
```

apparmor__mail_to

List of recipients to which a mail will be send in case a reboot is required.

```
apparmor__mail_to: [ 'root@{{ ansible_domain }}' ]
```

apparmor__mail_subject

Subject of the Email to be send in case a reboot is required to boot into a updated kernel version.

```
apparmor__mail_subject: 'Reboot required by AppArmor on {{ ansible_fqdn }}'
```

apparmor__mail_body

Body of the Email to be send in case a reboot is required to boot into a updated kernel version.

```
apparmor__mail_body: |
  Ansible has enabled AppArmor thought the boot loader configuration for the
  Linux kernel parameters on host {{ ansible_fqdn }}.
  You should check the status of the host and reboot it when convenient.
```

AppArmor profiles

apparmor__enforce_all_profiles

Put all profiles into enforcement mode. Use this only if you know what you are doing.

```
apparmor__enforce_all_profiles: False
```

apparmor__global_profile_status

Global configuration of the status of individual profiles. More specific matches overwrite more generic matches (example host overrules global).

Choices are:

enforce Result in enforcement of the policy defined in the profile as well as logging policy violation attempts.

complain This will not enforce the policy. Instead, it will log policy violations.

disable In this mode, policy violations are neither prevented nor logged.

Example:

```
apparmor__global_profile_status:
  'usr.sbin.nmbd': 'complain'
```

```
apparmor__global_profile_status: {}
```

apparmor__host_group_profile_status

Host group configuration of the status of individual profiles.

```
apparmor__host_group_profile_status: {}
```

apparmor__host_profile_status

Host configuration of the status of individual profiles.

```
apparmor__host_profile_status: {}
```

apparmor__local_config_global

Global additions or overrides of system profiles. Those changes will be configured in `/etc/apparmor.d/local/`. Check `/etc/apparmor.d/local/README` for details. All three dictionaries are merged into one profile configuration.

comment String, optional, default "Uncommented rule group". Comment for the given rules.

rules List of strings, required. AppArmor rules. Note that the rules are not comma terminated, this is done by the role template.

by_role Strings, optional, default "". Name of a role which manages the rules. Useful for using this role as role dependency.

delete Boolean, optional, default `False` . Delete the given rule(s).

Example:

```
apparmor__local_config_global:

  'usr.sbin.dnsmasq':
    - comment: 'Allow dnsmasq to read upstream DNS servers'
      rules:
        - '/etc/resolvconf/upstream.conf r'
        - '/etc/hosts.dnsmasq r'
      by_role: 'debops.dnsmasq'
    - comment: 'Allow dnsmasq to read /usr/share/dnsmasq-base/trust-anchors.conf'
      rules:
        - '/usr/share/dnsmasq-base/* r'
      by_role: 'debops.dnsmasq'

  'usr.bin.pidgin':
    - comment: 'Allow local Pidgin plugins'
      rules:
        - '@{HOME}/.purple/plugins/** rm'
```

```
apparmor__local_config_global: {}
```

apparmor__local_group_config

Host group additions or overrides of system profiles.

```
apparmor__local_group_config: {}
```

apparmor__local_host_config

Host additions or overrides of system profiles.

```
apparmor__local_host_config: {}
```

apparmor__local_dependent_config

System profiles managed by other roles using this role as dependency.

```
apparmor__local_dependent_config: {}
```

apparmor__global_tunables

Allows you to define or append variables which will be included by most profiles via the tunable concept of AppArmor. See also: https://wiki.ubuntu.com/DebuggingApparmor#Adjusting_Tunables

Examples:

```
apparmor__global_tunables: |
  @{HOMEDIRS}+=/exports/home/
```

```
apparmor__global_tunables: ''
```

apparmor__group_tunables

Host group definitions or additions to variables.

```
apparmor__group_tunables: ''
```

apparmor__host_tunables

Host definitions or additions to variables.

```
apparmor__host_tunables: ''
```

apparmor__tunables_dependent

Variable definitions managed by roles using this role as dependency.

```
apparmor__tunables_dependent: ''
```

Copyright

```
debops-contrib.apparmor - Install and configure AppArmor

Copyright (C) 2015-2017 Robin Schneider <ypid@riseup.net>
Copyright (C) 2015-2017 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.apparmor

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.apparmor v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]
- Added *apparmor__local_dependent_config* and *apparmor__tunables_dependent* to use this role as dependency for other roles.
- Added `delete` and `by_role` options to *apparmor__local_config_global*. [ypid]

Changed

- Renamed `apparmor_enable` to *`apparmor__enabled`*. [ypid]
- Changed namespace from `apparmor_` to `apparmor__`. `apparmor_[^_]` variables are hereby deprecated and you might need to update your inventory. This oneliner might come in handy to do this.

```
git ls-files -z | find -type f -print0 | xargs --null sed --in-place --regexp-
↪extended 's/(apparmor)_([^_])/\1__\2/g'
```

  [ypid]
- Use Ansible local fact `ansible_cmdline` to detect if kernel has been started with AppArmor enabled. [ypid]

Fixed

- Fix support for Ubuntu Trusty. [ypid]

Ansible role: debops-contrib.bitcoind

Introduction

The `debops-contrib.bitcoind` role allows you to manage and configure bitcoind. bitcoind can be used to run a full node in the distributed cryptocurrency network Bitcoin. A full node is a program that fully validates transactions and blocks and therefore enforces all of the rules of Bitcoin.

Refer to Running A Full Node for details.

Installation

This role requires at least Ansible `v2.1.5`. To install it, run:

```
ansible-galaxy install debops-contrib.bitcoind
```

Getting started

Example inventory

To manage bitcoind on a given host or set of hosts, they need to be added to the `[debops_service_bitcoind]` Ansible group in the inventory:

```
[debops_service_bitcoind]
hostname
```

Example playbook

If you are using this role without DebOps, here's an example Ansible playbook that uses the `debops-contrib.bitcoind` role:

```
---

- name: Setup and manage bitcoind
  hosts: [ 'debops_service_bitcoind' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops.etc_services
      tags: [ 'role::etc_services' ]
      etc_services__dependent_list:
        - '{{ bitcoind__etc_services__dependent_list }}'

    - role: debops.ferm
      tags: [ 'role::ferm' ]
      ferm__dependent_rules:
        - '{{ bitcoind__ferm__dependent_rules }}'

    - role: debops-contrib.bitcoind
      tags: [ 'role::bitcoind' ]
```

The playbook is shipped with this role under `./docs/playbooks/bitcoind.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::bitcoind` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::bitcoind:pkgs` Tasks related to system package management like installing or removing packages.

debops-contrib.bitcoind default variables

Sections

Packages and installation

bitcoind__base_packages

List of base packages to install.

```
bitcoind__base_packages:
  - 'bitcoind'
```

bitcoind__deploy_state

What is the desired state which this role should achieve? Possible options:

present Default. Ensure that bitcoind is installed and configured as requested.

absent Ensure that bitcoind is uninstalled and it's configuration is removed. This mode is not fully tested and might not remove all "traces".

```
bitcoind__deploy_state: 'present'
```

APT repository configuration

bitcoind__upstream_repository

APT URLs of the upstream Bitcoin repositories based on the OS distribution.

```
bitcoind__upstream_repository:
  Ubuntu: 'ppa:bitcoin/bitcoin'
```

bitcoind__upstream_key_fingerprint

The OpenPGP key fingerprint for the key by which the upstream APT repository is signed.

```
bitcoind__upstream_key_fingerprint: 'C70E F1F0 305A 1ADB 9986 DBD8 D46F 4542 8842 CE5E
↪'
```

Network configuration

bitcoind__allow

Allow access to bitcoind from specified IP addresses or CIDR networks. If not specified, allows access from all networks.

```
bitcoind__allow: []
```

bitcoind__group_allow

Allow access to bitcoind from specified IP addresses or CIDR networks. If not specified, allows access from all networks.

```
bitcoind__group_allow: []
```

bitcoind__host_allow

Allow access to bitcoind from specified IP addresses or CIDR networks. If not specified, allows access from all networks.

```
bitcoind__host_allow: []
```

bitcoind__interfaces

List of network interfaces from which to allow access to bitcoind. If not specified, allows access from all interfaces.

```
bitcoind__interfaces: []
```

bitcoind__port

Bitcoin P2P TCP port.

```
bitcoind__port: '{{ 8333 if (not bitcoind__testnet|bool) else 18333 }}'
```

bitcoind__rpc_port

Bitcoin JSON-RPC TCP port.

```
bitcoind__rpc_port: '{{ 8332 if (not bitcoind__testnet|bool) else 18332 }}'
```

bitcoind__max_connections

Maximum number of inbound+outbound connections. `{{ omit }}` will not configure this option explicitly which will cause bitcoind to fallback to it's compiled in default. Refer to *bitcoind(1)* for details.

```
bitcoind__max_connections: '{{ omit }}'
```

bitcoind__listen_onion

Automatically create Tor hidden service.

```
bitcoind__listen_onion: True
```

bitcoind__tor_control

Tor control port to use if onion listening enabled.

```
bitcoind__tor_control: '[::1]:9051'
```

bitcoind__tor_password

Tor control port password. Default is an empty password.

```
bitcoind__tor_password: ''
```

File and directory paths

bitcoind__home_path

The bitcoind system account home directory.

```
bitcoind__home_path: '{{ (ansible_local.root.home
                          if (ansible_local|d() and ansible_local.root|d() and
                              ansible_local.root.home|d())
                          else "/var/local") + "/" + bitcoind__user }}'
```

bitcoind__data_directory

The bitcoind data directory.

```
bitcoind__data_directory: '{{ bitcoind__home_path + "/.bitcoin" }}'
```

bitcoind__pid_file_path

The bitcoind PID file path.

```
bitcoind__pid_file_path: '{{ bitcoind__data_directory + "/bitcoind.pid" }}'
```

bitcoind__config_dir_path

The bitcoind config directory path. Not using `/etc/bitcoin` because Bitcoin tools do not expect this.

```
bitcoind__config_dir_path: '{{ bitcoind__data_directory }}'
```

bitcoind__config_file_path

The bitcoind config file path.

```
bitcoind__config_file_path: '{{ bitcoind__config_dir_path + "/bitcoin.conf" }}'
```

System user and group

bitcoind__user

System UNIX account under which bitcoind is run.

```
bitcoind__user: 'bitcoind'
```

bitcoind__group

System UNIX group used by bitcoind.

```
bitcoind__group: 'bitcoind'
```

bitcoind__gecos

Contents of the GECOS field set for the bitcoind account.

```
bitcoind__gecos: 'Bitcoin distributed currency'
```

bitcoind__shell

The default shell set on the bitcoind account.

```
bitcoind__shell: '/usr/sbin/nologin'
```

bitcoind configuration

bitcoind__testnet

Run on the test network instead of the real Bitcoin network.

```
bitcoind__testnet: False
```

bitcoind__txindex

Maintain a full transaction index, used by the `getrawtransaction` RPC call.

```
bitcoind__txindex: False
```

bitcoind__max_mem_pool_limit

The float variable used to limit the maximum RAM available for the transaction memory pool, by default ~50 % of system memory.

```
bitcoind__max_mem_pool_limit: 0.5
```

bitcoind__max_mem_pool

Keep the transaction memory pool below the given number of megabytes.

```
bitcoind__max_mem_pool: '{{ (ansible_memtotal_mb * bitcoind__max_mem_pool_limit) |␣
↪round | int }}'
```

bitcoind__max_upload_target

Tries to keep outbound traffic under the given target (in MiB per 24h), `0` means no limit.

```
bitcoind__max_upload_target: 0
```

bitcoind__print_to_console

Send trace/debug info to console instead of debug.log file. Can be set to `True` so that logging can be handled by systemd.

```
bitcoind__print_to_console: False
```

bitcoind__disable_wallet

Do not load the wallet and disable wallet RPC calls.

```
bitcoind__disable_wallet: True
```

bitcoind__custom_options

Custom options to append to the `bitcoin.conf` file.

```
bitcoind__custom_options: ''
```

Configuration for other Ansible roles

bitcoind__etc_services__dependent_list

Configuration for the debops.etc_services role which registers port numbers for bitcoind.

```
bitcoind__etc_services__dependent_list:

  - name: 'bitcoin'
    port: '{{ bitcoind__port }}'
    comment: 'bitcoin P2P'
    state: '{{ "present" if (bitcoind__deploy_state != "purged") else "absent" }}'

  - name: 'bitcoin-rpc'
    port: '{{ bitcoind__rpc_port }}'
    comment: 'bitcoin JSON-RPC'
    state: '{{ "present" if (bitcoind__deploy_state != "purged") else "absent" }}'
```

bitcoind__ferm__dependent_rules

Configuration for debops.ferm firewall. It should be added when debops.ferm role is used to configure bitcoind firewall rules.

```
bitcoind__ferm__dependent_rules:

  - type: 'accept'
    dport: [ 'bitcoin' ]
    saddr: '{{ bitcoind__allow + bitcoind__group_allow + bitcoind__host_allow }}'
    accept_any: True
    interface: '{{ bitcoind__interfaces }}'
    weight: '40'
    by_role: 'debops-contrib.bitcoind'
    name: 'bitcoin_p2p'
    rule_state: '{{ "present" if (bitcoind__deploy_state != "purged") else "absent" }}
↪'
```

Copyright

```
debops-contrib.bitcoind - Setup and manage bitcoind

Copyright (C) 2017 Robin Schneider <ypid@riseup.net>
Copyright (C) 2017 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.bitcoind

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.bitcoind v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]

Ansible role: debops-contrib.btrfs

Introduction

The `debops-contrib.btrfs` Ansible role allows you manage your Btrfs. Currently the role supports management of Btrfs subvolumes. More can be implemented as needed.

Installation

This role requires at least Ansible `v2.1.3`. To install it, run:

```
ansible-galaxy install debops-contrib.btrfs
```

Getting started

Example inventory

To manage Btrfs on host given in `debops_service_btrfs` Ansible inventory group:

```
[debops_service_btrfs]
hostname
```

Example playbook

Here's an example playbook that can be used to manage Btrfs:

```
---

- name: Manage Btrfs
  hosts: [ 'debops_service_btrfs' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops-contrib.btrfs
      tags: [ 'role::btrfs' ]
```

This playbooks is shipped with this role under `docs/playbooks/btrfs.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::btrfs` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::btrfs:pkts` Tasks related to the installation of required packages.

`role::btrfs:subvolumes` Tasks related to managing Btrfs subvolumes.

debops-contrib.btrfs default variables

Sections

Required packages

`btrfs__base_packages`

List of base packages to install.

```
btrfs__base_packages:
  - 'btrfs-tools'
```

Subvolumes

btrfs__subvolumes

Dict of BTRFS subvolumes.

```
btrfs__subvolumes: {}
```

btrfs__subvolumes_host_group

See *btrfs__subvolumes*.

```
btrfs__subvolumes_host_group: {}
```

btrfs__subvolumes_host

See *btrfs__subvolumes*.

```
btrfs__subvolumes_host: {}
```

Copyright

```
debops-contrib.btrfs - Manage Btrfs

Copyright (C) 2016 Robin Schneider <ypid@riseup.net>
Copyright (C) 2016 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.btrfs

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.btrfs v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]

Ansible role: debops-contrib.checkmk_agent

Introduction

This Ansible role allows you to install and manage the Check_MK agent. It is the client component of the Nagios-based Check_MK monitoring suite.

Installation

This role requires at least Ansible `v2.1.5`. To install it, run:

```
ansible-galaxy install debops-contrib.checkmk_agent
```

Getting started

Example inventory

You can install Check_MK agent on a host by adding it to the `[debops_services_checkmk_agent` host group in your Ansible inventory:

```
[debops_services_checkmk_agent]
hostname
```

Example playbook

Here's an example playbook that uses the `debops-contrib.checkmk_agent` role:

```
---

- name: Manage Check_MK agent
  hosts:  [ 'debops_service_checkmk_agent' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops-contrib.checkmk_agent/env
      tags: [ 'role::checkmk_agent', 'role::checkmk_agent:env', 'role::mariadb' ]

    - role: debops.apt_preferences
      tags: [ 'role::apt_preferences' ]
      apt_preferences__dependent_list:
        - '{{ checkmk_agent__apt_preferences__dependent_list }}'

    - role: debops.etc_services
      tags: [ 'role::etc_services' ]
      etc_services__dependent_list:
        - '{{ checkmk_agent__etc_services__dependent_list }}'
      when: ('xinetd' in checkmk_agent__type)
```

```
  - role: debops.ferm
    tags: [ 'role::ferm' ]
    ferm__dependent_rules:
      - '{{ checkmk_agent__ferm__dependent_rules }}'

  - role: debops.tcpwrappers
    tags: [ 'role::tcpwrappers' ]
    tcpwrappers__dependent_allow:
      - '{{ checkmk_agent__tcpwrappers__dependent_allow }}'

  - role: debops.authorized_keys
    tags: [ 'role::authorized_keys' ]
    authorized_keys__dependent_list:
      - '{{ checkmk_agent__authorized_keys__dependent_list }}'

  - role: debops.mariadb
    tags: [ 'role::mariadb' ]
    mariadb__dependent_users:
      - '{{ checkmk_agent__mariadb__dependent_users }}'
    when: ("mk_mysql" in checkmk_agent__combined_plugins)

  - role: debops-contrib.checkmk_agent
    tags: [ 'role::checkmk_agent' ]
```

The playbook is shipped with this role under `docs/playbooks/checkmk_agent.yml` from which you can symlink it to your playbook directory.

As you can see in this example playbook, the role makes use of a number of other roles to setup it's environment. Some of these dependency roles are only needed when services are detected. This is true for the debops.mariadb role which is used to manage a database user used to monitor the DBMS and databases by an automatically setup and configured agent plugin. To ensure that *`checkmk_agent__combined_plugins`* is valid in the context of other roles (in the same playbook) this variable is based on Ansible facts which are setup by the `debops-contrib.checkmk_agent/env` role prior to other dependency roles being called. For more details, refer to *`checkmk_agent__plugin_autodetect`*.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::checkmk_agent:env` Environment role tag, should be used in the playbook to execute a special environment role contained in the main role. The environment role prepares the environment for other dependency roles to work correctly.

`role::checkmk_agent` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::checkmk_agent:pkgs` Tasks related to system package management like installing or removing packages.

`role::checkmk_agent:plugins` Run tasks related to Check_MK agent plugin configuration.

`role::checkmk_agent:plugins:get` Run tasks related to Check_MK agent plugin retrieval.

debops-contrib.checkmk_agent default variables

Sections

Basic configuration options

checkmk_agent__base_packages

List of base packages to install.

```
checkmk_agent__base_packages:
  - 'check-mk-agent'
```

checkmk_agent__type

List of Check_MK agent query protocols. Valid options are **ssh** and **xinetd**.

```
checkmk_agent__type: [ 'ssh' ]
```

checkmk_agent__allow

List of IP addresses or network CIDR ranges allowed to connect to the Check_MK agent through the firewall. If list are empty, anyone can connect.

```
checkmk_agent__allow: []
```

checkmk_agent__deploy_state

What is the desired state which this role should achieve? Possible options:

present Default. Ensure that the Check_MK agent is installed and configured as requested.

absent Ensure that the Check_MK agent is uninstalled and it's configuration is removed.

```
checkmk_agent__deploy_state: 'present'
```

Monitoring site integration

checkmk_agent__server_inventory_group

Ansible inventory host group used to lookup the Check_MK server.

```
checkmk_agent__server_inventory_group: 'debops_service_checkmk_server'
```

checkmk_agent__server

Ansible inventory name of Check_MK server. By default it will be autodetected via *checkmk_agent__server_inventory_group* host group configuration. If the Check_MK server is not managed by Ansible, set this to `False`.

```
checkmk_agent__server: '{{ groups[checkmk_agent__server_inventory_group][0]
                           if (checkmk_agent__server_inventory_group in groups) and
                              (groups[checkmk_agent__server_inventory_group] | length␣
↪> 0)
                           else "" }}'
```

checkmk_agent__site

Define Check_MK monitoring site name where the agent is registered. By default it will be autodetected from the local facts stored under the `checkmk_server` dictionary key. Fallback to site name `debops` if *checkmk_agent__server* is undefined or the Ansible local facts for *checkmk_agent__server* can't be found. If the Check_MK server is managed manually this variable must be defined accordingly in the Ansible inventory.

```
checkmk_agent__site: '{{ hostvars[checkmk_agent__server].ansible_local.checkmk_server.
↪keys()[0]|d("debops")
                         if (checkmk_agent__server|d() and
                             (checkmk_agent__server in hostvars) and
                             ("ansible_local" in hostvars[checkmk_agent__server]) and
                             ("checkmk_server" in hostvars[checkmk_agent__server].
↪ansible_local))
                         else "debops" }}'
```

checkmk_agent__autojoin

Automatically add agent host to the Check_MK monitoring site. If the Check_MK server is not managed by Ansible and you want automated agent registration to work, manually define at least *checkmk_agent__autojoin_url*, *checkmk_agent__autojoin_secret* and *checkmk_agent__user_key*.

```
checkmk_agent__autojoin: '{{ True if checkmk_agent__autojoin_url else False }}'
```

checkmk_agent__autojoin_url

Check_MK server WebAPI URL for agent registration. By default it will be autodetected from the local facts stored under the `checkmk_server` dictionary key. If the Check_MK server is managed manually this variable must be defined accordingly in the Ansible inventory.

```
checkmk_agent__autojoin_url: '{{ hostvars[checkmk_agent__server].ansible_local.
↪checkmk_server[checkmk_agent__site].webapi_url|d("")
                               if (checkmk_agent__server|d() and
                                   (checkmk_agent__server in hostvars) and
                                   ("ansible_local" in hostvars[checkmk_agent__
↪server]) and
                                   ("checkmk_server" in hostvars[checkmk_agent__
↪server].ansible_local) and
                                   (checkmk_agent__site in hostvars[checkmk_agent__
↪server].ansible_local.checkmk_server))
                               else "" }}'
```

checkmk_agent__autojoin_user

Account for agent registration via Check_MK WebAPI.

```
checkmk_agent__autojoin_user: 'ansible'
```

checkmk_agent__autojoin_secret

Authentication secret for WebAPI registration. If the Check_MK server is managed manually the password path must be adjusted accordingly in the Ansible inventory.

```
checkmk_agent__autojoin_secret: '{{ lookup("password", secret + "/credentials/" +␣
↪hostvars[checkmk_agent__server].ansible_fqdn + "/checkmk_server/" + checkmk_agent__
↪site + "/" + checkmk_agent__autojoin_user + "/secret")|d("")
                                  if checkmk_agent__server|d() and checkmk_agent__
↪site|d() else "" }}'
```

checkmk_agent__fqdn

FQDN of the agent host used for registration.

```
checkmk_agent__fqdn: '{{ ansible_local.core.fqdn
                         if (ansible_local|d() and ansible_local.core|d() and
                             ansible_local.core.fqdn|d())
                         else ansible_fqdn }}'
```

checkmk_agent__host_attributes

Check_MK attributes and WATO tags used for managing the host. For more details check *checkmk_agent__host_attributes*.

```
checkmk_agent__host_attributes:
  tag_agent: '{{ "cmk-agent-ssh" if "ssh" in checkmk_agent__type|d(["ssh"]) else "cmk-
↪agent" }}'
```

checkmk_agent__discovery_mode

Service discovery mode. Possible values are `new` (only find new services), `remove` (remove exceeding services), `fixall` (remove exceeding and add new services), `refresh` (clean all autochecks and discover from scratch) and `False` (don't run service discovery).

```
checkmk_agent__discovery_mode: 'new'
```

Agent xinetd options

checkmk_agent__exec

Check_MK agent executable path. If you query the agent from multiple servers, you may want to set this to `/usr/bin/check_mk_caching_agent` .

```
checkmk_agent__exec: '/usr/bin/check_mk_agent'
```

checkmk_agent__port

Listen port for Check_MK agent.

```
checkmk_agent__port: '6556'
```

Agent SSH user options

checkmk_agent__ssh_user

SSH user to query Check_MK agent.

```
checkmk_agent__ssh_user: 'checkmk'
```

checkmk_agent__ssh_group

Primary group of SSH user querying Check_MK agent.

```
checkmk_agent__ssh_group: 'checkmk'
```

checkmk_agent__ssh_allow_group

Group membership required to access the system by SSH. If the `AllowGroups sshd_config` option is not managed by debops.sshd this variable might need to be defined accordingly in the Ansible inventory.

```
checkmk_agent__ssh_allow_group: '{{ "sshusers"
                                    if ("sshd" in ansible_local) and
                                       ("allow_groups" in ansible_local.sshd) and
                                       ("sshusers" in ansible_local.sshd.allow_groups)
                                    else "" }}'
```

checkmk_agent__user_home

Home directory of the SSH user querying the Check_MK agent.

```
checkmk_agent__user_home: '/var/lib/check_mk_agent'
```

checkmk_agent__user_key

Public key for user authentication when accessing the agent via SSH. By default it will be autodetected from the local facts stored under the `checkmk_server` dictionary key. If the Check_MK server is managed manually this variable must be defined accordingly in the Ansible inventory.

```
checkmk_agent__user_key: '{{ hostvars[checkmk_agent__server].ansible_local.checkmk_
↪server[checkmk_agent__site].ssh_public_key|d("")
                            if (checkmk_agent__server|d() and
                                (checkmk_agent__server in hostvars) and
                                ("ansible_local" in hostvars[checkmk_agent__server])␣
↪and
                                ("checkmk_server" in hostvars[checkmk_agent__server].
↪ansible_local) and
                                (checkmk_agent__site in hostvars[checkmk_agent__
↪server].ansible_local.checkmk_server))
                            else "" }}'
```

Agent plugins

checkmk_agent__plugins

List of upstream Check_MK agent plugins to always enable.

```
checkmk_agent__plugins: []
```

checkmk_agent__group_plugins

"Host Group" list of upstream Check_MK agent plugins to always enable.

```
checkmk_agent__group_plugins: []
```

checkmk_agent__host_plugins

"Host" list of upstream Check_MK agent plugins to always enable.

```
checkmk_agent__host_plugins: []
```

checkmk_agent__plugin_autodetect

Try to install Check_MK agent plugins for hardware and applications auto detected via Ansible facts.

```
checkmk_agent__plugin_autodetect: True
```

checkmk_agent__autodetected_plugins

Autodetected list of upstream Check_MK agent plugins to enable.

```
checkmk_agent__autodetected_plugins:
  - '{{ ["smart"] if (ansible_virtualization_role in ["host"]) else [] }}'
```

checkmk_agent__facts_plugin_map

Ansible local facts to Check_MK plugin mapping. If the Ansible local fact is present and optional conditions defined in the `templates/etc/ansible/facts.d/checkmk_agent.fact.j2` file are met, the Check_MK plugin will be enabled.

```
checkmk_agent__facts_plugin_map:
  mariadb: 'mk_mysql'
  mysql: 'mk_mysql'
  nginx: 'nginx_status'
  apache: 'apache_status'
```

checkmk_agent__combined_plugins

Combined list of all plugins which are going to be installed. Specified as Ansible local fact so that this variable is also valid in `when` conditions evaluated in the context of other roles called from the same playbook as this role.

```
checkmk_agent__combined_plugins: '{{ ansible_local.checkmk_agent.plugins
                                     if (ansible_local|d() and ansible_local.checkmk_
↪agent|d() and
                                         ansible_local.checkmk_agent.plugins|d())
                                     else [] }}'
```

checkmk_agent__plugin_path

Destination path to install the Check_MK agent plugins.

```
checkmk_agent__plugin_path: '/usr/lib/check_mk_agent/plugins'
```

MySQL/MariaDB monitoring plugins options

checkmk_agent__plugin_mysql

Determines how to configure the `mk_mysql` monitoring plugin. If this is set to `automatic` a database user which has read access to the database server will be created. Set to `manual` to configure it manually. See https://mathias-kettner.de/checkmk_mysql.html

```
checkmk_agent__plugin_mysql: 'automatic'
```

checkmk_agent__plugin_mysql_user

Database user account name to use for monitoring.

```
checkmk_agent__plugin_mysql_user: 'monitor'
```

checkmk_agent__plugin_mysql_password

Database user password to use for monitoring.

```
checkmk_agent__plugin_mysql_password: '{{
  lookup("password", secret + "/mariadb/" + (ansible_local.mariadb.delegate_to
  if (ansible_local.mariadb|d() and ansible_local.mariadb.delegate_to|d()) else "") +
  "/credentials/" + checkmk_agent__plugin_mysql_user + "/password length=48") }}'
```

checkmk_agent__plugin_mysql_priv

Privileges of the database user used for monitoring.

```
checkmk_agent__plugin_mysql_priv: '*.*:SELECT,SHOW DATABASES'
```

nginx monitoring plugins options

checkmk_agent__plugin_nginx_servers

This option allows you to configure the servers which the nginx_status plugin should monitoring. This might be required when the auto detection of the plugin fails for example because the default server does not allow `/nginx_status` . This can happen because the plugin tires to connect with the IP address set as Host. This is currently set manually to `localhost` as workaround. See https://github.com/debops-contrib/ansible-checkmk_agent/pull/3

Examples:

```
checkmk_agent__plugin_nginx_servers:
  - proto: 'http'
    ipaddress: 'some-appliance.corp.com'
    port: 80
  - proto: 'http'
    ipaddress: '[::1]'
    port: 80

# Or:
checkmk_agent__plugin_nginx_servers: 'automatic'
```

```
checkmk_agent__plugin_nginx_servers:
  - proto: 'http'
    ipaddress: 'localhost'
    port: 80
```

Agent plugins source options

checkmk_agent__git_dest_host

The host to which the Check_MK agent source directory should be cloned. Can be set to `localhost` so that the repo is only cloned one time and not once for each host.

```
checkmk_agent__git_dest_host: '{{ inventory_hostname }}'
```

checkmk_agent__git_repo

Check_MK agent source repository.

```
checkmk_agent__git_repo: 'https://git.mathias-kettner.de/check_mk.git'
```

checkmk_agent__git_dest

Check_MK agent source directory on the host.

```
checkmk_agent__git_dest: '{{ "/usr/local/src/check-mk/" + checkmk_agent__git_repo.
↪split("://")[1] }}'
```

checkmk_agent__git_version_map

Map from Check_MK release to git commit hash. This is done because Check_MK does not cryptographically signed their work and this role wants to comply with the DebOps Software Source Policy.

```
checkmk_agent__git_version_map:
  'v1.2.6p12': 'cf2aaf2f7d60ca0445a239915bfc41aa6f3ee739'
  'v1.2.6p20': '988e5d4e8fbcf9ac73365ffcfb2d12080c4ee052'
  'v1.2.8p16': 'e5e216abca9a946a29eab94334be30cc146e7fec'
```

checkmk_agent__git_version_unsigned_fallback

Defines the behavior when a requested version is not specified in *checkmk_agent__git_version_map*. When this is set to `True` and no mapping for the used release is found, the role will fallback to using the unsigned git tag directly!

```
checkmk_agent__git_version_unsigned_fallback: False
```

checkmk_agent__git_version

Check_MK agent git branch to deploy. Set `auto` to set version to dpkg package version.

```
checkmk_agent__git_version: 'auto'
```

Configuration for other Ansible roles

checkmk_agent__apt_preferences__dependent_list

Configuration for the debops.apt_preferences role.

```
checkmk_agent__apt_preferences__dependent_list:

  - package: 'check-mk-agent'
    backports: [ 'jessie' ]
    reason: 'Package not available in stable Debian Jessie'
    by_role: 'debops-contrib.checkmk_agent'
    state: '{{ "present"
               if (checkmk_agent__deploy_state in ["present"])
               else "absent" }}'
```

checkmk_agent__etc_services__dependent_list

Configuration for the debops.etc_services role which registers port numbers for Check_MK agent.

```
checkmk_agent__etc_services__dependent_list:

  - name: 'check-mk-agent'
    port: '{{ checkmk_agent__port }}'
    comment: 'Check_MK agent (via xinetd)'
    state: '{{ "present"
               if (("xinetd" in checkmk_agent__type) and
                   (checkmk_agent__deploy_state in ["present"]))
               else "absent" }}'
```

checkmk_agent__ferm__dependent_rules

Configuration for the debops.ferm role.

```
checkmk_agent__ferm__dependent_rules:

  - type: 'accept'
    dport: [ 'check-mk-agent' ]
    saddr: '{{ checkmk_agent__allow }}'
    accept_any: True
    weight: '20'
    by_role: 'debops-contrib.checkmk_agent'
    rule_state: '{{ "present"
                    if (("xinetd" in checkmk_agent__type) and
                        (checkmk_agent__deploy_state in ["present"]))
                    else "absent" }}'
```

checkmk_agent__tcpwrappers__dependent_allow

Configuration for the debops.tcpwrappers Ansible role.

```
checkmk_agent__tcpwrappers__dependent_allow:

  - daemon: 'inetd'
    comment: 'Ensure legacy tcpwrappers ACL is absent'
    by_role: 'debops-contrib.checkmk_agent'
    state: 'absent'

  - daemon:
      - 'check_mk_agent'
      - 'check_mk_caching_agent'
      ## Not required:
      # - 'inetd'
      # - 'xinetd'
    client: '{{ checkmk_agent__allow }}'
    accept_any: False
    weight: '50'
    comment: 'Allow remote connections to the Check_MK agent'
    by_role: 'debops-contrib.checkmk_agent'
    state: '{{ "present"
               if (("xinetd" in checkmk_agent__type) and
                   (checkmk_agent__deploy_state in ["present"]))
               else "absent" }}'
```

checkmk_agent__authorized_keys__dependent_list

Authorized key configuration for the debops.authorized_keys role.

```
checkmk_agent__authorized_keys__dependent_list:
  - name: '{{ checkmk_agent__ssh_user }}'
    group: '{{ checkmk_agent__ssh_group }}'
    sshkeys:
      - '{{ checkmk_agent__user_key }}'
    options: '{{ authorized_keys__options_map.strict }}'
    key_options: 'command="{{ "/usr/bin/sudo " if not checkmk_agent__ssh_user == "root
↪" else "" }}{{ checkmk_agent__exec }}"'
    state: '{{ "present"
               if (("ssh" in checkmk_agent__type) and
                   (checkmk_agent__deploy_state in ["present"]))
               else "absent" }}'
```

checkmk_agent__mariadb__dependent_users

Database user definition for the debops.mariadb role.

```
checkmk_agent__mariadb__dependent_users:

  - user: '{{ checkmk_agent__plugin_mysql_user }}'
    password: '{{ checkmk_agent__plugin_mysql_password }}'
    priv: '{{ checkmk_agent__plugin_mysql_priv }}'
    priv_default: False
    priv_aux: False
    append_privs: False
    owner: 'root'
    # group: '{{ checkmk_agent__ssh_group }}'
    creds_path: '/etc/check_mk/mysql.cfg'
    state: '{{ "present" if (checkmk_agent__deploy_state in ["present"]) else "absent
↪" }}'
```

Default variable details

Some of the `debops-contrib.checkmk_agent` default variables have more extensive configuration values than simple strings or lists, here you can find documentation and examples for them.

- *checkmk_agent__host_attributes*

checkmk_agent__host_attributes

This is a configuration dictionary defining the host attributes which are associated with this host in the Check_MK server Web interface (aka WATO). The following configuration keys are supported:

alias Optional. A comment or description of this host.

contactgroups Optional. Only members of the contact groups listed here have WATO permission to this host. The value for this configuration key is another dictionary where the following configuration keys must be defined:

groups Required. List of contact groups defined in WATO.

use_for_services Optional. With this option contact groups that are added to hosts are always being added to services, as well. This only makes a difference if you have assigned other contact groups to

services via rules in *Host & Service Parameters*. Allowed values are `True` and `False`. Defaults to `False`.

ipaddress Optional. In case the name of the host is not resolvable via `/etc/hosts` or DNS by your monitoring server, you can specify an explicit IP address or a resolvable DNS name of the host here.

parents Optional. List of host names which act as parents. Parents are used to configure the reachability of hosts by the monitoring server. A host is considered to be unreachable if all of its parents are unreachable or down.

site Optional. Name of the monitoring site that should monitor this host.

tag_<wato_tag> Optional. Any tag defined in the WATO Web interface or when using the server role via `checkmk_server__multisite_cfg_wato_host_tags` can be assigned here. Example: To set the WATO tag `criticality` to `test` this would be defined as `tag_criticality: test`.

Ansible facts

The role exposes part of it's state by means of Ansible local facts for other roles and playbooks to use. The interface is considered public and changes to it happen in compliance with Semantic Versioning of the role and will be mentioned in the *Changelog*. Here you can find documentation and examples for them.

Specification

ansible_local.checkmk_agent.plugins List of all Check_MK Agent plugin names which are enabled (and configured if necessary) by the role. Refer to *checkmk_agent__plugins* and related variables for details.

Availability: Always, after `debops-contrib.checkmk_agent/env` has been run.

Example

```
{
    "plugins": ["nginx_status"]
}
```

Copyright

```
debops-contrib.checkmk_agent - Setup Check_MK monitoring agent

Copyright (C) 2015-2017 Reto Gantenbein <reto.gantenbein@linuxmonk.ch>
Copyright (C) 2015-2017 Robin Schneider <ypid@riseup.net>
Copyright (C) 2016-2017 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.checkmk_agent

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ganto.

debops-contrib.checkmk_agent master - unreleased

Added

- New inventory variable *checkmk_agent__server_inventory_group* which can be used to define custom Ansible host group name for Check_MK server lookup. [ganto]
- Support *checkmk_agent__deploy_state*. [ypid]
- Automatically enable the `smart` Check_MK agent plugin on physical hosts to query Self-Monitoring, Analysis and Reporting data from disks. [ypid]
- Add *Ansible facts* documentation. [ypid]
- Add *checkmk_agent__git_dest_host* which can be used to clone the Check_MK only once to the Ansible controller. [ypid]

Changed

- Raise HTTP timeout for discovery and activation WebAPI calls to 120s to avoid timeout issues on large hosts with many service checks. [ganto]
- If possible run WebAPI invocation for automated agent registration and host attribute updates on the Check_MK server to avoid possible firewall issues. [ganto]
- Rename `checkmk_agent__hostname` to *checkmk_agent__fqdn*. You might need to update your inventory. [ypid]
- Rename `checkmk_agent__group_plugin_map` to *checkmk_agent__facts_plugin_map*. You might need to update your inventory. [ypid]
- Increase Ansible min version to `2.1.5`. Everything below is deprecated anyway and has vulnerabilities so you don't want to use that anymore. [ypid]

Removed

- Remove the `debops_checkmk_agent` Ansible inventory group. Make sure your hosts are in `debops_service_checkmk_agent`. [ypid]

Fixed

- Correctly use Ansible *changed* and *skipped* task filters. [ganto]
- Let xinetd bind on `AF_INET6` to ensure IPv6 reachability of the agent. [ypid]
- Fix TCP Wrappers support for xinetd. [ypid]
- Ensure the `/etc/check_mk` directory is present before running dependency roles. Fixes MariaDB credentials configuration. [ypid]

Security

- Enforce known good git commit hashes. As upstream does not cryptographically sign their work, the known good hashes have to be pinned manually in *`checkmk_agent__git_version_map`* of the role. [ypid]

debops-contrib.checkmk_agent v0.1.1 - 2017-01-23

Added

- `role::checkmk_agent:plugins:get` Ansible tag for cloning/pulling related tasks. [ypid]

Changed

- Run the debops.ferm role also when **`xinetd`** is not listed in *`checkmk_agent__type`* to allow to migrate between different types. [ypid]

Fixed

- Fix **`xinetd`** support which is filtered by `tcpwrappers` and which is configured by debops.tcpwrappers to deny all connections by default (whitelisting). [ypid]
- Fix lookup of non-default monitoring site specified as Ansible local fact by the debops-contrib.checkmk_server role. [ganto]

Security

- Change git clone URL used to install additional plugins from `http://` to https://git.mathias-kettner.de/check_mk.git to mitigate potential MITM attacks against the unauthenticated `http://` connection. That, together with using the latest git master branch by default could result in malicious code being executed on systems where the agent is installed. **`git pull`** will use the new URL from now on. Note that "GnuTLS recv error[s]" have been observed which might have to be fixed elsewhere. "GnuTLS recv error (-9): A TLS packet with unexpected length was received" [ypid]

debops-contrib.checkmk_agent v0.1.0 - 2016-11-07

Added

- Initial release [ganto]

Ansible role: debops-contrib.checkmk_server

Introduction

debops-contrib.checkmk_server is an Ansible role which installs and manages Check_MK, a Nagios-based system monitoring solution. Check_MK supports different monitoring backends such as Nagios or Icinga (v1.x) and features a powerful configuration language for creating check inventories.

Installation

This role requires at least Ansible `v2.1.5`. To install it, run:

```
ansible-galaxy install debops-contrib.checkmk_server
```

Getting started

By default Check_MK server is installed from the *check-mk-raw* Debian package as provided by Mathias Kettner upstream. It includes the **omd** tool which is used for managing the monitoring sites. The role will create a default site called 'debops'. After the setup it can be reached by accessing `https://<fqdn>/check_mk/debops`.

Example inventory

You can install Check_MK server on a host by adding it to the `[debops_service_checkmk_server]` host group in your Ansible inventory:

```
[debops_service_checkmk_server]
hostname
```

Example playbook

Here's an example playbook that uses the `debops-contrib.checkmk_server` role to install Check_MK server:

```
---

- name: Manage Check_MK server
  hosts: [ 'debops_service_checkmk_server' ]
  become: True

  roles:

    - role: debops.etc_services
      tags: [ 'role::etc_services' ]
      etc_services__dependent_list:
```

```
        - '{{ checkmk_server__etc_services__dependent_list }}'
      when: checkmk_server__multisite_livestatus|d()

    - role: debops.ferm
      tags: [ 'role::ferm' ]
      ferm__dependent_rules:
        - '{{ checkmk_server__ferm_dependent_rules }}'

    - role: debops-contrib.checkmk_server
      tags: [ 'role::checkmk_server' ]
```

The inclusion of the debops.ferm is optional. This playbooks is shipped with this role under `docs/playbooks/checkmk_server.yml` from which you can symlink it to your playbook directory.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::checkmk_server` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::checkmk_server:rules` Execute tasks which are generating the monitoring rules definitions.

`role::checkmk_server:multisite` Execute tasks which configure the Check_MK multisite Web interface.

`role::checkmk_server:mkp` Execute tasks to install Check_MK packages.

debops-contrib.checkmk_server default variables

Sections

General Configuration

`checkmk_server__version`

Check_MK software version.

```
checkmk_server__version: '1.2.8p25'
```

`checkmk_server__version_label`

Check_MK version label used with the **omd** tool.

```
checkmk_server__version_label: '{{ checkmk_server__version }}.cre'
```

checkmk_server__site_update

Update Check_MK site if current version is lower than *checkmk_server__version*

```
checkmk_server__site_update: False
```

checkmk_server__patches

Custom patches to apply after installing Check_MK package

```
checkmk_server__patches:
  - patch: 'check-mk-raw-1.2.8-set-https-proxy-header.patch'
    file: '/omd/versions/{{ checkmk_server__version_label }}/skel/etc/apache/apache-
↪own.conf'
  - patch: 'check-mk-raw-1.2.8p4-read-X-Forwarded-Port-header.patch'
    file: '/omd/versions/{{ checkmk_server__version_label }}/skel/etc/apache/conf.d/
↪omd.conf'
```

checkmk_server__ferm_dependent_rules

Firewall configuration using the debops.ferm Ansible role.

```
checkmk_server__ferm_dependent_rules: '{{
    checkmk_server__ferm_web_rules +
    (checkmk_server__ferm_livestatus_rules if checkmk_server__multisite_livestatus␣
↪else [])
  }}'
```

checkmk_server__ferm_web_rules

Firewall configuration for WATO Web access.

```
checkmk_server__ferm_web_rules:
  - type: 'accept'
    dport: '{{ [ "http", "https" ] if checkmk_server__pki else [ "http" ] }}'
    saddr: '{{ checkmk_server__web_allow }}'
    accept_any: True
    weight: '40'
    role: 'checkmk_server'
```

checkmk_server__ferm_livestatus_rules

Firewall configuration for Multisite Livestatus access.

```
checkmk_server__ferm_livestatus_rules:
  - type: 'accept'
    dport: [ '{{ checkmk_server__livestatus_port|string }}' ]
    saddr: '{{ checkmk_server__livestatus_allow }}'
    accept_any: True
    weight: '40'
    role: 'checkmk_server'
```

checkmk_server__web_allow

List of IP addresses or network CIDR ranges allowed to connect to the Check_MK Web interface. If list is empty, anyone can connect.

```
checkmk_server__web_allow: []
```

checkmk_server__livestatus_allow

List of IP addresses or network CIDR ranges allowed to connect to the Check_MK Livestatus TCP socket. If list is empty, anyone can connect.

```
checkmk_server__livestatus_allow: []
```

checkmk_server__etc_services__dependent_list

Add entry for Livestatus to `/etc/services` using the `debops.etc_services` role.

```
checkmk_server__etc_services__dependent_list:
  - name: 'check-mk-livestatus'
    port: '{{ checkmk_server__livestatus_port }}'
    comment: 'Check_MK server Livestatus'
```

checkmk_server__livestatus_port

TCP port for Multisite Livestatus socket.

```
checkmk_server__livestatus_port: 6557
```

checkmk_server__software_inventory

Enable collection of installed software. Requires the `mk_inventory` plugin to be installed on the Check_MK agents.

```
checkmk_server__software_inventory: True
```

APT packages

checkmk_server__raw_package

Check_MK RAW package download URL. Alternatively this can also be a local deb file or a package name in an already available apt repository.

```
checkmk_server__raw_package: 'https://mathias-kettner.de/support/{{ checkmk_server__
↪version }}/check-mk-raw-{{ checkmk_server__version }}_0.{{ ansible_distribution_
↪release }}_amd64.deb'
```

checkmk_server__prerequisite_packages

List of prerequisite packages which must be available before installing the Check_MK RAW package

```
checkmk_server__prerequisite_packages: [ 'apache2', 'python-passlib' ]
```

Check_MK Site Configuration

checkmk_server__site

Check_MK site name. Set to `False` to disable site configuration.

```
checkmk_server__site: 'debops'
```

checkmk_server__hostname

Set Check_MK server DNS hostname (e.g. for agent download, API calls, ...). FIXME: Rename to checkmk_server__fqdn.

```
checkmk_server__hostname: '{{ ansible_local.core.fqdn
                              if (ansible_local|d() and ansible_local.core|d() and
                                  ansible_local.core.fqdn|d())
                              else ansible_fqdn }}'
```

checkmk_server__site_url

Check_MK server site URL.

```
checkmk_server__site_url: '{{ ("https://" if checkmk_server__pki else "http://") +
                               checkmk_server__hostname + "/" +
                               checkmk_server__site
                              if checkmk_server__site|d() else "" }}'
```

checkmk_server__webapi_url

WebAPI URL of monitoring site.

```
checkmk_server__webapi_url: '{{ checkmk_server__site_url + "/check_mk/webapi.py"
                                if checkmk_server__site|d() else "" }}'
```

checkmk_server__omd_config

Check_MK site configuration set via **omd config**. Changing these values will shutdown Check_MK during reconfiguration. Check *checkmk_server__omd_config* for more details.

```
checkmk_server__omd_config: '{{
    checkmk_server__omd_config_email +
    checkmk_server__omd_config_core +
    (checkmk_server__omd_config_livestatus if checkmk_server__multisite_
↪livestatus|d() else [])
  }}'
```

checkmk_server__omd_config_email

Administrator email address set via OMD.

```
checkmk_server__omd_config_email:
  - var: 'ADMIN_MAIL'
    value: 'hostmaster@{{ ansible_domain if ansible_domain else ansible_hostname }}'
```

checkmk_server__omd_config_core

Monitoring core set via OMD. Possible values: `icinga` or `nagios` .

```
checkmk_server__omd_config_core:
  - var: 'CORE'
    value: 'icinga'
```

checkmk_server__omd_config_livestatus

Livestatus service configuration via OMD.

```
checkmk_server__omd_config_livestatus:
  - var: 'LIVESTATUS_TCP'
    value: 'on'
  - var: 'LIVESTATUS_TCP_PORT'
    value: '{{ checkmk_server__livestatus_port }}'
```

checkmk_server__sshkeys

Indicate if a SSH keypair should be provided to allow agent connections via SSH. For more information check *checkmk_server__sshkeys*.

```
checkmk_server__sshkeys:
  generate_keypair: True
```

checkmk_server__ssh_user

User account which is used to query Check_MK agent via SSH.

```
checkmk_server__ssh_user: 'checkmk'
```

checkmk_server__ssh_command

Command which is executed when querying the Check_MK agent via SSH. Set this to **/usr/bin/check_mk_caching_agent** when agents are queried by multiple servers.

```
checkmk_server__ssh_command: '{{ "/usr/bin/sudo " if (checkmk_server__ssh_user !=
↪"root") else "" }}/usr/bin/check_mk_agent'
```

checkmk_server__ssh_arguments

SSH arguments used when querying the Check_MK agent. For possible options check **man 5 ssh_config**.

```
checkmk_server__ssh_arguments: '-o BatchMode=yes -o StrictHostKeyChecking=no -o␣
↪ConnectTimeout=10s'
```

Multisite Web Configuration

checkmk_server__multisite_slave

Indicate if this site is a distributed monitoring slave which receives the Check_MK configuration from another Check_MK server instance.

```
checkmk_server__multisite_slave: False
```

checkmk_server__multisite_livestatus

Enable multisite Livestatus service. This is required for distributed monitoring of this site.

```
checkmk_server__multisite_livestatus: '{{ True if checkmk_server__multisite_slave|d()␣
↪else False }}'
```

checkmk_server__multisite_config_path

Configuration path for Check_MK multisite configurations. Relative to the site's chroot directory.

```
checkmk_server__multisite_config_path: 'etc/check_mk/multisite.d'
```

checkmk_server__multisite_config_map

List of dictionaries which will generate the Check_MK multisite configuration in *checkmk_server__multisite_config_path*.

```
checkmk_server__multisite_config_map: '{{ checkmk_server__multisite_cfg_wato_host_
↪tags +
                                        checkmk_server__multisite_cfg_wato_aux_tags␣
↪+
                                        checkmk_server__multisite_cfg_roles }}'
```

checkmk_server__multisite_cfg_wato_host_tags

Multisite `wato_host_tags` variable definition.

```
checkmk_server__multisite_cfg_wato_host_tags:
  - name: 'wato_host_tags'
    value: '{{ checkmk_server__multisite_default_wato_host_tags }}'
```

checkmk_server__multisite_default_wato_host_tags

Default upstream host tag configuration with additional `cmk-agent-ssh` tag to indicate SSH-based Check_MK agents.

```
checkmk_server__multisite_default_wato_host_tags:
  - agent:
      'Agent type':
        - 'cmk-agent-ssh':
            'Check_MK Agent (ssh)': []
        - 'cmk-agent':
            'Check_MK Agent (xinetd)': ['tcp']
        - 'snmp-only':
            'SNMP (Networking device, Appliance)': ['snmp']
        - 'snmp-v1':
            'Legacy SNMP device (using V1)': ['snmp']
        - 'snmp-tcp':
            'Dual: Check_MK Agent + SNMP': ['snmp', 'tcp']
        - 'ping':
            'No Agent': []
  - criticality:
      'Criticality':
        - 'prod':
            'Productive system': []
        - 'critical':
            'Business critical': []
        - 'test':
            'Test system': []
        - 'offline':
            'Do not monitor this host': []
  - networking:
      'Networking Segment':
        - 'lan':
            'Local network (low latency)': []
        - 'wan':
            'WAN (high latency)': []
        - dmz:
            'DMZ (low latency, secure access)': []
```

checkmk_server__multisite_cfg_wato_aux_tags

Multisite `wato_aux_tags` variable definition.

```
checkmk_server__multisite_cfg_wato_aux_tags:
  - name: 'wato_aux_tags'
```

```
    value: '{{ checkmk_server__multisite_default_wato_aux_tags }}'
```

checkmk_server__multisite_default_wato_aux_tags

Default upstream auxiliary tags configuration.

```
checkmk_server__multisite_default_wato_aux_tags:
  - snmp: 'monitor via SNMP'
  - tcp: 'monitor via Check_MK Agent'
```

checkmk_server__multisite_cfg_roles

Multisite user `roles` configuration.

```
checkmk_server__multisite_cfg_roles:
  - name: 'roles'
    value: '{{ checkmk_server__multisite_default_roles |
               combine(checkmk_server__multisite_debops_roles, recursive=True) |
               combine(checkmk_server__multisite_custom_roles, recursive=True) }}'
```

checkmk_server__multisite_default_roles

Default upstream Multisite user role definitions.

```
checkmk_server__multisite_default_roles:
  admin:
    alias: 'Administrator'
    builtin: True
    permissions: {}
  guest:
    alias: 'Guest User'
    builtin: True
    permissions: {}
  user:
    alias: 'Normal monitoring user'
    builtin: True
    permissions: {}
```

checkmk_server__multisite_debops_roles

Multisite user role definitions used by the Ansible role.

```
checkmk_server__multisite_debops_roles:
  api:
    alias: 'Automation API'
    basedon: 'user'
    permissions:
      'general.see_all': True
      'wato.all_folders': True
      'wato.hosttags': True
      'wato.see_all_folders': True
      'wato.seeall': True
      'wato.use': True
```

checkmk_server__multisite_custom_roles

Custom multisite user role definitions.

```
checkmk_server__multisite_custom_roles: {}
```

checkmk_server__multisite_users

Locally defined multisite users to be configured. See *checkmk_server__multisite_users* for more information.

```
checkmk_server__multisite_users: '{{ checkmk_server__multisite_debops_users |
                                     combine(checkmk_server__multisite_custom_users,␣
↪recursive=True) }}'
```

checkmk_server__multisite_debops_users

Multisite user definitions used by the Ansible role.

```
checkmk_server__multisite_debops_users:
  ansible:
    alias: 'Automation User used by Ansible'
    automation_secret: '{{ lookup("password", secret + "/credentials/" + ansible_fqdn␣
↪+ "/checkmk_server/" + checkmk_server__site + "/ansible/secret") }}'
    roles: [ 'api' ]
  sitesync:
    alias: 'Synchronization User for Multisite'
    password: '{{ lookup("password", secret + "/credentials/" + ansible_fqdn + "/
↪checkmk_server/" + checkmk_server__site + "/sitesync/password") }}'
    roles: [ 'admin' ]
```

checkmk_server__multisite_custom_users

Custom multisite user definitions.

```
checkmk_server__multisite_custom_users: {}
```

checkmk_server__multisite_user_defaults

Default user properties for local users defined in *checkmk_server__multisite_users*

```
checkmk_server__multisite_user_defaults:
  force_authuser: False
  force_authuser_webservice: False
  locked: False
  roles: [ 'user' ]
  start_url: 'dashboard.py'
```

checkmk_server__multisite_user_connections

LDAP user synchronization connection settings. See *checkmk_server__multisite_user_connections* for more information.

```
checkmk_server__multisite_user_connections: []
```

checkmk_server__multisite_user_connection_defaults

Default properties for LDAP user connections defined in *checkmk_server__multisite_user_connections*

```
checkmk_server__multisite_user_connection_defaults:
  active_plugins: {}
  cache_livetime: 300
  comment: ''
  debug_log: False
  description: ''
  directory_type: 'openldap'
```

```
  disabled: False
  docu_url: ''
  group_dn: ''
  group_scope: 'sub'
  id: 'default'
  user_dn: ''
  user_id_umlauts: 'keep'
  user_scope: 'sub'
```

checkmk_server__distributed_sites

Distributed monitoring sites configuration. For more details see *checkmk_server__distributed_sites*

```
checkmk_server__distributed_sites: {}
```

checkmk_server__distributed_sites_defaults

Default sites properties for distributed monitoring.

```
checkmk_server__distributed_sites_defaults:
  username: 'sitesync'
  password: '{{ lookup("password", secret + "/credentials/" + ansible_fqdn + "/
↪checkmk_server/" + checkmk_server__site + "/sitesync/password") }}'
  disabled: False
  disable_wato: True
  insecure: False
  multisiteurl: ''
  persist: False
  replicate_ec: False
  replicate_mkps: True
  replication: ''
  status_host: None
  timeout: 10
  url_prefix: ''
  user_login: True
```

Monitoring Rules

checkmk_server__site_config_path

Configuration path for Check_MK main configurations. Relative to the site's chroot directory.

```
checkmk_server__site_config_path: 'etc/check_mk/conf.d'
```

checkmk_server__site_config_map

List of configuration dictionaries which will generate the Check_MK monitoring definitions.

```
checkmk_server__site_config_map: '{{ checkmk_server__site_cfg_contactgroups +
                                     checkmk_server__site_cfg_rules +
                                     checkmk_server__site_cfg_hostgroups +
                                     checkmk_server__site_cfg_servicegroups +
                                     checkmk_server__site_cfg_datasource_programs +
                                     checkmk_server__site_cfg_netif_description +
                                     checkmk_server__site_cfg_notification_defaults +
                                     checkmk_server__site_cfg_software_inventory }}'
```

checkmk_server__contact_defaults

Default contact properties. For a list of valid contact properties see `checkmk_server__contact_properties` defined in `vars/main.yml` . They are described under *checkmk_server__multisite_users* .

```
checkmk_server__contact_defaults:
  contactgroups: [ 'all' ]
  disable_notifications: False
  email: ''
  host_notification_options: 'durfs'
  notification_method: 'email'
  notification_period: '24X7'
  notifications_enabled: False
  pager: ''
  service_notification_options: 'wucrfs'
```

checkmk_server__site_cfg_contactgroups

Define default contact group for all contacts.

```
checkmk_server__site_cfg_contactgroups:
  - name: 'define_contactgroups'
    value:
      all: 'Everything'
```

checkmk_server__site_cfg_rules

Define Check_MK monitoring rules.

```
checkmk_server__site_cfg_rules: '{{ checkmk_server__site_upstream_rules }}'
```

checkmk_server__site_upstream_rules

Default upstream rule definitions.

```
checkmk_server__site_upstream_rules:
  - name: 'bulkwalk_hosts'
    tags: [ 'snmp', '!snmp-v1' ]
    description: 'Hosts with the tag "snmp-v1" must not use bulkwalk'
  - name: 'extra_service_conf'
    element: 'check_interval'
    value: 1440
    conditions: [ 'Check_MK HW/SW Inventory$' ]
    description: 'Restrict HW/SW-Inventory to once a day'
  - name: 'host_contactgroups'
    value: 'all'
    description: 'Put all hosts into the contact group "all"'
  - name: 'only_hosts'
    tags: [ '!offline' ]
    description: 'Do not monitor hosts with the tag "offline"'
  - name: 'ping_levels'
    value:
      loss: [ 80.0, 100.0 ]
      packets: 6
      timeout: 20
      rta: [ 1500.0, 3000.0 ]
    tags: [ 'wan' ]
    description: 'Allow longer round trip times when pinging WAN hosts'
```

checkmk_server__site_cfg_hostgroups

Define host groups.

```
checkmk_server__site_cfg_hostgroups:
  - name: 'define_hostgroups'
    value: {}
```

checkmk_server__site_cfg_servicegroups

Define service groups.

```
checkmk_server__site_cfg_servicegroups:
  - name: 'define_servicegroups'
    value: {}
```

checkmk_server__site_cfg_datasource_programs

Define additional `datasource_programs` for agent access via SSH.

```
checkmk_server__site_cfg_datasource_programs:
  - name: 'datasource_programs'
    value: 'ssh {{ checkmk_server__ssh_arguments }} -l {{ checkmk_server__ssh_user }}
↪<IP> {{ checkmk_server__ssh_command }}'
    tags: [ 'cmk-agent-ssh' ]
    description: 'Check_MK Agent via SSH'
```

checkmk_server__site_cfg_software_inventory

Check_MK rules for enabling software inventory check. This check can be enabled/disabled by setting *checkmk_server__software_inventory*.

```
checkmk_server__site_cfg_software_inventory:
  - name: 'inventory_check_interval'
    value: 1440
    rule_state: '{{ "present" if (checkmk_server__software_inventory|d() | bool)
                    else "absent" }}'
  - name: 'active_checks'
    element: 'cmk_inv'
    description: 'Enable collection of hardware/software information'
    rule_state: '{{ "present" if (checkmk_server__software_inventory|d() | bool)
                    else "absent" }}'
```

checkmk_server__site_cfg_notification_defaults

Set fallback email address for rule based notifications. Must be set including domain otherwise it won't be accepted by Check_MK.

```
checkmk_server__site_cfg_notification_defaults:
  - name: 'notification_fallback_email'
    filename: 'global.mk'
    template: 'key_value'
    value: '{{ ansible_local.core.admin_public_email[0]
               if (("core" in ansible_local) and
                   ("admin_public_email" in ansible_local.core))
               else "root@" + ansible_domain }}'
```

checkmk_server__site_cfg_netif_description

Set interface name instead of index for network interface check via `if_inventory_uses_description`.

```
checkmk_server__site_cfg_netif_description:
  - name: 'if_inventory_uses_description'
```

```
    filename: 'networking.mk'
    template: 'key_value'
    value: 'True'
    wato: False
```

checkmk_server__site_packages

Additional Check_MK packages (MKP) to be installed. See *checkmk_server__site_packages* for more information.

```
checkmk_server__site_packages: []
```

PKI Configuration

checkmk_server__pki

Enable or disable support for HTTPS in Check_MK server (using debops.pki).

```
checkmk_server__pki: '{{ (True
                          if (ansible_local|d() and ansible_local.pki|d() and
                              ansible_local.pki.enabled|d() | bool)
                          else False) | bool }}'
```

checkmk_server__pki_path

Base path for PKI directory.

```
checkmk_server__pki_path: '{{ ansible_local.pki.path
                              if (ansible_local|d() and ansible_local.pki|d() and
                                  ansible_local.pki.path|d())
                              else "/etc/pki/realms" }}'
```

checkmk_server__pki_realm

Default PKI realm used by Check_MK server.

```
checkmk_server__pki_realm: '{{ ansible_local.pki.realm
                               if (ansible_local|d() and ansible_local.pki|d() and
                                   ansible_local.pki.realm|d())
                               else "domain" }}'
```

checkmk_server__pki_ca

Root CA certificate, relative to *checkmk_server__pki_realm*.

```
checkmk_server__pki_ca: 'CA.crt'
```

checkmk_server__pki_crt

Host certificate, relative to *checkmk_server__pki_realm*.

```
checkmk_server__pki_crt: 'default.crt'
```

checkmk_server__pki_key

Host private key, relative to *checkmk_server__pki_realm*.

```
checkmk_server__pki_key: 'default.key'
```

checkmk_server__tls_options

Additional Apache mod_ssl options. Valid configuration keys: `SSLCipherSuite`, `SSLHonorCipherOrder`, `SSLProtocols`, `SSLStrictSNIVHostCheck`

```
checkmk_server__tls_options:
  SSLHonorCipherOrder: 'On'
  SSLCipherSuite: 'ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:
↪ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS'
```

Default variable details

Some of the debops-contrib.checkmk_server default variables have more extensive configuration than simple strings or lists, here you can find documentation and examples for them.

checkmk_server__omd_config

omd is a command line utility which is used to manage Check_MK monitoring sites. Some basic configuration options of the site will be set via this tool. These options are defined in *checkmk_server__omd_config* which is a list of YAML dictionaries, each with two key/value pairs defining the OMD property to be set. One key has to be `var` with the variable name to be set as value. The other key has to be `value` with the variable value to be set as value.

See *checkmk_server__omd_config_core* for an example.

checkmk_server__sshkeys

This configuration variable indicates if SSH keys should be configured for accessing the Check_MK agent. If set to a non-empty value a additional Check_MK host tag "Check_MK Agent via SSH" is configured and the SSH public key is set as Ansible fact, so that it can be used by the debops-contrib.checkmk_agent role to configure SSH-based agent access. The *checkmk_server__sshkeys* variable is a dictionary which support the following keys:

generate_keypair Generate a new SSH keypair for the Check_MK site user. Possible values: `True` or `False`

keysize Specify the number of bits used when generating a new keypair. Only valid when `generate_keypair` is set to `True`. Defaults to `4096`.

privatekey_file Pre-generated SSH private key file which should be configured for SSH-based Check_MK agent access.

publickey_file Pre-generated SSH public key file. Must be the public key of the private key set with `privatekey_file`.

checkmk_server__site_packages

Check_MK has a plugin system where site customizations such as additional checks can be installed. This is done via `.mkp` packages. For more information see the upstream documentation about Check_MK extension packages.

Packages which should be installed for the current Check_MK site are defined as a list of YAML dictionaries with the following configuration keys. One of `path` or `url` must be given:

`name` Name of the package, required.

`path` Optional. Local file system path of the `.mkp` package archive on the Ansible controller. Cannot be combined with the `url` parameter.

`url` Optional. Download URL of the `.mkp` package archive. Cannot be combined with the `path` parameter.

`checksum` Optional. Checksum of the download archive given in the `url` parameter. Cannot be combined with the `path` parameter. Refer to the Ansible get_url module for the accepted parameter format.

checkmk_server__multisite_users

Configuration dictionary to define local WATO users. When running Ansible they are merged into the `users.mk` user database of Check_MK. Users already defined in WATO or synchronized from an identity management system such as LDAP won't be overwritten.

The dictionary key has to be the user name to create or manage. The following properties can be set via Ansible inventory:

`alias` Full name, required.

`automation_secret` Optional. Automation secret for machine accounts. Set this instead of `item.password` if the account is used for authentication of WebAPI calls.

`contactgroups` Optional. List of contact groups the user is a member of. Defaults to `[]`.

`disable_notifications` Optional. Temporarily disable all notifications for this user. Defaults to `False`.

`email` Optional. Email address.

`force_authuser` Optional. Only show hosts and services the user is a contact for. Defaults to `False`.

`force_authuser_webservice` Optional. Export only hosts and services the user is a contact for. Defaults to `False`.

`host_notification_options` Optional. Host events which should be notified. String combined of the following letters: `d` : Host goes down `u` : Host get unreachable `r` : Host goes up again `f` : Start or end of flapping state `s` : Start or end of a scheduled downtime Defaults to `durfs`.

`locked` Optional. Disable login to this account. Defaults to `False`.

`notification_method` Optional. Event notification method. Defaults to `email` (currently only supported method).

`notification_period` Optional. Notification time period. Default to `24x7` (currently only supported period).

`notifications_enabled` Optional. Generally enable notifications for this user. Defaults to `False`.

`pager` Optional. Pager address.

`password` Optional. Set given password in Apache `htpasswd` file. Can be used for form-based authentication in WATO and HTTP basic authentication in Icinga, PNP4Nagios and NagVis.

`roles` Optional. List of permission roles defined in *`checkmk_server__multisite_cfg_roles`*. Defaults to `[ 'user' ]`.

service_notification_options Optional. Service events which should be notified. String combined of the following letters: `w` : Service goes into warning state `u` : Service goes into unknown state `c` : Service goes into critical state `r` : Service recovers to OK `f` : Start or end of flapping state `s` : Start or end of a scheduled downtime Defaults to `wucrfs` .

start_url Optional. Start URL to display in main frame. Defaults to `dashboard.py` .

Example

Create custom administrator account with random password:

```
checkmk_server__multisite_users:

  bob:
    alias: 'Bob Admin'
    password: '{{ lookup("password", secret + "/credentials/" + ansible_fqdn + "/
↪checkmk_server/" + checkmk_server__site + "/bob/password length=15") }}'
    roles: [ 'admin' ]
```

checkmk_server__multisite_user_connections

List of LDAP user synchronization connection definitions. Multiple connection definitions are allowed. Each connection can define the following properties via Ansible inventory:

binddn Distinguished name used for authenticating against the LDAP server, required.

bindpw Password used for authenticating against the LDAP server, required.

server LDAP server host name, required.

group_dn Base DN for LDAP group queries, required.

userdn Base DN for LDAP user queries, required.

active_plugins Optional. Configuration dictionary of attribute synchronization plugins. See *LDAP Attribute Synchronization Plugins* for more details.

cache_livetime Optional. Time in seconds how long to cache LDAP user information. Defaults to: `300` .

comment Optional. Comment about user connection definition.

connect_timeout Optional. Connect timeout.

debug_log Optional. Enable debug logging for LDAP user synchronization. Allowed values are `True` or `False` . Defaults to: `False`

description Optional. Short description of user connection definition being displayed in the connection list.

directory_type Optional. LDAP directory type used to set default user and group attributes. Allowed values are `openldap` , `389directoryserver` or `ad` . Defaults to: `openldap` .

disabled Optional. Do not enable user connection. Allowed values are `True` or `False` . Defaults to: `False`

docu_url Optional. Documentation URL.

failover_servers Optional. List of failover LDAP host names.

group_filter Optional. Group search filter (e.g. `(objectclass=groupOfNames)`). This will overwrite the default set by `item.directory_type` .

group_member Optional. Group member attribute name (e.g. `member`).

group_scope Optional. Group search scope. Allowed values are `sub` (search whole subtree below base DN), `base` (search only the entry at the base DN) or `one` (search all entries one level below the base DN). Defaults to: `sub` .

id Optional. Connection identifier. Defaults to `default` .

lower_user_ids Optional. Set lower case user IDs. Allowed values are `True` or `False` . Defaults to: `False`

no_persistent Optional. Don't use persistent LDAP connections. Allowed values are `True` or `False` . Defaults to: `False`

port Optional. TCP port. Defaults to: `389`

response_timeout Optional. Response timeout.

suffix Optional. LDAP connection suffix.

use_ssl Optional. Encrypt the network connection using SSL. Allowed values are `True` or `False` . Defaults to: `False`

user_filter Optional. User search filter (e.g. `(objectclass=account)`). This will overwrite the default set by `item.directory_type` .

user_filter_group Optional. Filter users by group.

user_id Optional. User ID attribute name (e.g. `uid`).

user_id_umlauts Optional. Translate Umlauts in user IDs (deprecated). Allowed values are `keep` or `replace` . Defaults to `keep` .

user_scope Optional. User search scope. Allowed values are `sub` (search whole subtree below base DN), `base` (search only the entry at the base DN) or `one` (search all entries one level below the base DN). Defaults to: `sub` .

LDAP Attribute Synchronization Plugins

The LDAP user synchronization connector supports various plugins for setting WATO user properties based on LDAP attributes and filters. Each plugin is a configuration dictionary with the plugin name as key.

alias Set user alias based on LDAP attribute.

attr Optional. LDAP attribute to sync. Defaults to `cn` .

auth_expire Checks wether or not the user auth must be invalidated.

attr Optional. LDAP attribute to be used as indicator. Defaults to `krbpasswordexpiration` .

disable_notifications Disable notifications based on LDAP attribute.

attr Optional. LDAP attribute to sync.

email Set email address based on LDAP attribute.

`attr` Optional. LDAP attribute to sync. Default to `mail` .

force_authuser Set visibility of host/services based on LDAP attribute.

attr Optional. LDAP attribute to sync.

force_authuser_webservice Set visibility of host/services for WebAPI access based on LDAP attribute.

attr Optional. LDAP attribute to sync.

groups_to_attributes Set custom user attributes based on the group memberships in LDAP.

nested Optional. Handle nested group memberships (Active Directory only at the moment)

other_connections Optional. List of alternative LDAP connection IDs to sync group membership.

groups_to_contactgroups Add the user to contactgroups based on the group memberships in LDAP.

nested Optional. Handle nested group memberships (Active Directory only at the moment)

other_connections Optional. List of alternative LDAP connection IDs to sync contactgroup membership.

groups_to_roles Set user roles based on distinguished names from LDAP. This is a configuration dictionary with the role name defined in *checkmk_server__multisite_cfg_roles* as key and a list of group references as value. Each group reference supports the following properties.

group_dn Group DN used for role assignment.

connection Optional. Alternative connection ID used for group query.

pager Set pager number based on LDAP attribute.

attr Optional. LDAP attribute to be used as indicator. Defaults to `mobile`.

start_url Set WATO start URL based on LDAP attribute.

attr Optional. LDAP attribute to sync. Defaults to `start_url`.

Example

Small example configuration for user authentication via LDAP showing the use of some LDAP plugins:

```
checkmk_server__multisite_user_connections:
  - server: 'localhost'
    binddn: 'cn=admin,dc=example,dc=com'
    bindpw: 'secret'
    group_dn: 'ou=groups,dc=example,dc=com'
    user_dn: 'ou=users,dc=example,dc=com'
    user_filter: '(objectclass=posixAccount)'
    active_plugins:
      alias:
        attr: 'gecos'
      groups_to_roles:
        admin:
          - group_dn: 'cn=wato-admin,ou=groups,dc=example,dc=com'
```

This will synchronize all users in from the DN `ou=users,dc=example,dc=com` to WATO, fills the user's alias property with the value from the `gecos` LDAP attribute and assign the admin role to the members of the 'wato-admin' group.

checkmk_server__distributed_sites

This setting will define Check_MK Multisite connections to other Check_MK monitoring sites. Each site entry is a nested YAML dictionary with the site name as top key. The following sub keys are supported as site properties.

alias An alias or description of the site, required.

disabled Optional. Temporarily disable this connection. Defaults to `False`.

disable_wato Optional. Disable configuration via WATO on this site. Defaults to `True`.

insecure Optional. Ignore SSL certificate errors. Defaults to `False`.

multisiteurl Optional. URL of the remote Check_MK site including `/check_mk/` . This will be used by the main site to fetch resources from this site.

password Optional. User password for user defined in `item.username` used for authentication on this site.

persist Optional. Use persistent connections to this site. Defaults to `False` .

replicate_ec Optional. Replicate Event Console configuration to this site. Defaults to `False` .

replicate_mkps Optional. Replicate extensions (MKPs and files in `~/local/`). Defaults to `True` .

replication Optional. WATO replication allows you to manage several monitoring sites with a logically centralized WATO. Slave sites receive their configuration from master sites. By default this value is unset which means that the there is no replication with this site. Set this to `slave` to enable configuration push to this site.

socket Optional. Livestatus connection socket. By default this value is unset which corresponds to the local site. In case this is a foreign site on localhost or a remote site, this value must be set to a TCP or UNIX socket such as `tcp:<hostname>:<port>` or `unix:<path>` . When connecting to remote site make sure that Livestatus over TCP is activated there.

status_host Optional. By specifying a status host for each non-local connection you prevent Multisite from running into timeouts when remote sites do not respond. The value must be specified as `[ '<site>','<hostname>' ]` . By default this value is unset. Check the upstream documentation for more information.

timeout Optional. Connect timeout in seconds before this site is considered to be unreachable. Defaults to `10` .

url_prefix Optional. The URL prefix will be prepended to links of addons like PNP4Nagios or the classical Icinga GUI when a link to such applications points to a host or service on that site.

username Optional. User name used to synchronize configuration data with this site in case `item.replication` is set to `slave` . Defaults to `sitesync` .

user_login Optional. Allow users to directly directly login into the Web GUI of this site. Defaults to `True` .

The default values for the distributed sites configuration are defined in *`checkmk_server__distributed_sites_defaults`* and can be overwritten via Ansible inventory.

A lot of parameter descriptions are copied from the upstream source code which is copyrighted by Mathias Kettner and released under the GPL-2.0.

Guides and examples

Alternative Package Source

Unfortunately Mathias Kettner, the upstream packager of Check_MK, doesn't provide a Apt repository for the `check-mk-raw` Debian package. By default the role will therefore download the package from the official download URL before installing it.

However, it is possible to define an alternative installation sources for the `check-mk-raw` package:

- In case the package is managed in a custom Apt repository the package name can be specified. E.g.:

```
checkmk_server__raw_package: 'check-mk-raw-{{ checkmk_server__version }}'
```

Important

The application version is always part of the package name. This will allow multiple versions to be installed at once.

- If no direct Internet connection and no local repository is available, for example in a simple Vagrant environment, a local file path can be defined. E.g.:

```
checkmk_server__raw_package: '/vagrant/check-mk-raw-{{ checkmk_server__version }}_
↪0.{{ ansible_distribution_release }}_amd64.deb'
```

Manually setup Monitoring Site

By default the role will setup a monitoring site named according to *checkmk_server__site*. Sometimes it might be desired to not let Ansible generate a site configuration by itself but use the **omd** tool manually instead. This can be achieved by simply setting:

```
checkmk_server__site: False
```

When not managing the site configuration through Ansible, the debops-contrib.checkmk_agent role won't be able to auto-detect the server properties. They need to be specified manually in the Ansible inventory. For more details check the agent role documentation.

Copyright

```
debops-contrib.checkmk_server - Manage Check_MK monitoring server

Copyright (C) 2016-2017 Reto Gantenbein <reto.gantenbein@linuxmonk.ch>
Copyright (C) 2016-2017 DebOps https://debops.org/

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see https://www.gnu.org/licenses/
```

Credits

- Reto Gantenbein <reto.gantenbein_at_linuxmonk.ch>
  - author of the debops-contrib.checkmk_server role

Changelog

debops-contrib.checkmk_server

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ganto.

debops-contrib.checkmk_server master - unreleased

Added

- Initial release [ganto]

Fixed

- Fix `checkmk_server__ssh_command` which would have been wrongly generated with `checkmk_server__ssh_user` set to `root`. [ypid]

Ansible role: debops-contrib.dropbear_initramfs

Introduction

The `debops-contrib.dropbear_initramfs` role allows you to setup SSH access to the initramfs prior to the root filesystem being mounted using Dropbear as SSH server.

This can be used to unlock a full disk encrypted host remotely via SSH.

Installation

This role requires at least Ansible `v2.1.4`. To install it, run:

```
ansible-galaxy install debops-contrib.dropbear_initramfs
```

Getting started

Which version to use

The current version of dropbear provided in Debian jessie is a bit old and does not provide SOTA cryptography. The role already supports the updated dropbear version from Debian stretch which is now available as dropbear-initramfs. The proper way to install it on Debian jessie is to use debops.reprepro.

It has also been tested to install the version from stretch on jessie. Note that this is discouraged by Debian and DebOps but you might decide to make an exception in this case when you know what you are doing.

If you do, all you have to do is to enable the stretch repositories and use APT pinning to ensure that no unwanted packages are pulled from stretch. And tell `debops-contrib.dropbear_initramfs` that you want the newer version. If you are using DebOps, you can set the following in your inventory:

```
## Load APT pinning presets.
apt_preferences__group_list:
  - '{{ apt_preferences__preset_list | list }}'

apt__group_sources:
  - comment: 'Enable Debian stretch repository'
    uri: '{{ ansible_local.apt.default_sources_map.Debian[0]
             if (ansible_local|d() and ansible_local.apt|d() and
                 ansible_local.apt.default_sources_map|d() and
                 ansible_local.apt.default_sources_map.Debian|d() and
                 ansible_local.apt.default_sources_map.Debian[0]|d())
             else "http://deb.debian.org/debian" }}'
    suites:
      - 'stretch'
    component:
      - 'main'

dropbear_initramfs__base_packages:
  - 'dropbear-initramfs'
```

Example inventory

To setup the dropbear ssh server in initramfs of a given host or a set of hosts, they need to be added to the `[debops_service_dropbear_initramfs]` Ansible group in the inventory:

```
[debops_service_dropbear_initramfs]
hostname
```

Example playbook

Here's an example playbook that uses the `debops-contrib.dropbear_initramfs` role:

```
---

- name: Setup the dropbear ssh server in initramfs
  hosts: [ 'debops_service_dropbear_initramfs' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:
```

```
- role: debops.apt_preferences
  tags: [ 'role::apt_preferences' ]
  apt_preferences__dependent_list:
    - '{{ dropbear_initramfs__apt_preferences__dependent_list }}'

- role: debops-contrib.dropbear_initramfs
  tags: [ 'role::dropbear_initramfs' ]
```

The playbook is shipped with this role under `./docs/playbooks/dropbear_initramfs.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::dropbear_initramfs` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::dropbear_initramfs:pkgs` Tasks related to system package management like installing or removing packages.

debops-contrib.dropbear_initramfs default variables

Sections

Packages and installation

`dropbear_initramfs__base_packages`

List of base packages to install.

Supported versions:

- `dropbear-initramfs`
- `dropbear`

```
dropbear_initramfs__base_packages:
  - '{{ "dropbear"
        if (
          (ansible_distribution == "Debian" and ansible_distribution_release in [
↪"jessie"]) or
          (ansible_distribution == "Ubuntu" and ansible_distribution_release in [
↪"precise", "trusty"])
        )
        else "dropbear-initramfs" }}'
```

dropbear_initramfs__packages

List of additional packages to install.

```
dropbear_initramfs__packages: []
```

dropbear_initramfs__deploy_state

What is the desired state which this role should achieve? Possible options:

present Default. Ensure that dropbear is configured in the initramfs to allow ssh connections.

absent Ensure that dropbear and related configuration maintained by this role are absent.

```
dropbear_initramfs__deploy_state: 'present'
```

Simple initramfs network

Refer to https://www.kernel.org/doc/Documentation/filesystems/nfs/nfsroot.txt for support configuration options.

Note that the `IP` kernel parameter as of Debian jessie only supports legacy IPv4. But don't worry, the role has you covered. Refer to *dropbear_initramfs__interfaces*.

dropbear_initramfs__network_autoconf

Method to use for autoconfiguration. Use `off` or `none` for manual network configuration (see below).

```
dropbear_initramfs__network_autoconf: 'dhcp'
```

dropbear_initramfs__network_device

Default network device.

```
dropbear_initramfs__network_device: 'eth0'
```

dropbear_initramfs__network_address

Manual network address to set.

```
dropbear_initramfs__network_address: '{{ ansible_default_ipv4.address }}'
```

dropbear_initramfs__network_netmask

Manual subnet mask to set.

```
dropbear_initramfs__network_netmask: '{{ ansible_default_ipv4.netmask }}'
```

dropbear_initramfs__network_gateway

Manual gateway to set.

```
dropbear_initramfs__network_gateway: '{{ ansible_default_ipv4.gateway }}'
```

dropbear_initramfs__network_manual

The `IP` kernel parameter used when *dropbear_initramfs__network_autoconf* is disabled.

The *ipwrap* filter causes IPv6 address to work on some platforms. Refer to: https://serverfault.com/questions/445296/is-there-a-linux-kernel-boot-parameter-to-configure-an-ipv6-address/701451#701451

```
dropbear_initramfs__network_manual: '{{
  (dropbear_initramfs__network_address|ipwrap) + "::" +
  (dropbear_initramfs__network_gateway|ipwrap) + ":" +
  dropbear_initramfs__network_netmask + "::" +
  dropbear_initramfs__network_device + ":none" }}'
```

dropbear_initramfs__network

The `IP` kernel parameter as it is configured by the role.

```
dropbear_initramfs__network: '{{ dropbear_initramfs__network_manual
                                 if (dropbear_initramfs__network_autoconf in ["off",
↪"none"])
                                 else dropbear_initramfs__network_autoconf }}'
```

Complex initramfs network

These variables are dictionaries with additional network configuration. See *dropbear_initramfs__interfaces* documentation for more details.

dropbear_initramfs__interfaces

Dictionary which holds the configuration of additional network configuration for all hosts in the Ansible inventory.

```
dropbear_initramfs__interfaces: {}
```

dropbear_initramfs__group_interfaces

Dictionary which holds the configuration of additional network configuration for hosts in a specific Ansible inventory group.

```
dropbear_initramfs__group_interfaces: {}
```

dropbear_initramfs__host_interfaces

Dictionary which holds the configuration of additional network configuration for specific hosts in the Ansible inventory.

```
dropbear_initramfs__host_interfaces: {}
```

dropbear_initramfs__combined_interfaces

Dictionary which combines all of the other network interface configuration variables and is used in the role tasks and templates to generate the configuration.

```
dropbear_initramfs__combined_interfaces: '{{ lookup("template", "lookup/dropbear_
↪initramfs__combined_interfaces.j2", convert_data=False) | from_yaml }}'
```

Initramfs generation

dropbear_initramfs__update_options

Additional options for the **update-initramfs** command. The default is to regenerate the initramfs for all installed kernel versions.

```
dropbear_initramfs__update_options: '-k all'
```

Authorized ssh keys

See *dropbear_initramfs__authorized_keys* for more details.

dropbear_initramfs__authorized_keys

List of authorized ssh keys configured on all hosts in the Ansible inventory.

```
dropbear_initramfs__authorized_keys: []
```

dropbear_initramfs__group_authorized_keys

List of authorized ssh keys configured on a group of hosts in the Ansible inventory.

```
dropbear_initramfs__group_authorized_keys: []
```

dropbear_initramfs__host_authorized_keys

List of authorized ssh keys configured on specific hosts in the Ansible inventory.

```
dropbear_initramfs__host_authorized_keys: []
```

dropbear_initramfs__combined_authorized_keys

Combines list of authorized ssh keys as used in the role tasks.

```
dropbear_initramfs__combined_authorized_keys: '{{ dropbear_initramfs__authorized_keys␣
↪+
                                                  dropbear_initramfs__group_
↪authorized_keys +
                                                  dropbear_initramfs__host_authorized_
↪keys }}'
```

dropbear_initramfs__authorized_keys_options

List of default SSH options added to all public keys. If it's set to `{{ omit }}`, no options will be added automatically. The list of options can be overridden by the `item.options` parameter. Refer to *dropbear(8)* for details.

```
dropbear_initramfs__authorized_keys_options: '{{ omit }}'
```

Configuration for other Ansible roles

dropbear_initramfs__apt_preferences__dependent_list

Configuration for the debops.apt_preferences role.

```
dropbear_initramfs__apt_preferences__dependent_list:

  - package: 'dropbear-initramfs'
    pin: 'release o=Debian,n=stretch'
    priority: 800
    reason: 'Stronger cryptography. dropbear 2014.65-1 only offers: hmac-sha1-96,hmac-
↪sha1,hmac-md5'
    by_role: 'debops-contrib.dropbear_initramfs'
    state: '{{ "present"
               if (("dropbear-initramfs" in dropbear_initramfs__base_packages) and
                   (ansible_distribution == "Debian" and ansible_distribution_release␣
↪in ["jessie"]))
               else "absent" }}'
```

Default variable details

Some of `debops-contrib.dropbear_initramfs` default variables have more extensive configuration than simple strings or lists, here you can find documentation and examples for them.

dropbear_initramfs__interfaces

The *dropbear_initramfs__interfaces* and similar dictionaries behave similar to the `ifupdown__*_interfaces` dictionaries of the debops.ifupdown role. Refer to the documentation of debops.ifupdown for details.

Compared to the debops.ifupdown, only a limited subset of parameters is currently supported:

type Optional. Anything other than `ether` will be ignored.

inet Optional. IPv4 configuration method used by a given interface. If you set this parameter to `False`, no IPv4 configuration will be applied. Currently only `static` (default) and `False` is supported.

inet6 Optional. IPv6 configuration method used by a given interface. If you set this parameter to `False`, no IPv6 configuration will be applied. Currently only `static` (default) and `False` is supported.

address or addresses Optional. A string or a list of IPv4 and/or IPv6 addresses to set on a given network interface, in the form of `ipaddress/prefix` or CIDR. Remember that you need to specify the host IP address and not the network; the `192.0.2.1/24` is the correct notation, and `192.0.2.0/24` is incorrect.

gateway or gateways Optional. Specify the IPv4 or IPv6 address of the network gateway to which outgoing packets will be directed. If it's a list of addresses, first valid address for a network type will be used as the gateway.

Examples

Configure `eth0` with a global IPv6 address.

```
dropbear_initramfs__interfaces:
  'eth0':
    inet: False
    inet6: 'static'
    addresses:
      - '2001:db8::23/64'
    gateways:
      - '2001:db8::'
```

dropbear_initramfs__authorized_keys

The *dropbear_initramfs__authorized_keys* and similar variables are used to define what SSH keys should be allowed for remote initramfs login. Each list item is a dictionary with the following supported options:

sshkeys Required. String containing either a SSH public key, or an URL to a resource which returns a file with SSH public keys (only one URL is allowed at the moment), or a list of SSH public keys.

options Optional. String or list of SSH options which should be set for each key specified on the `item.sshkeys` list. Refer to *dropbear(8)* for details.

If this parameter is not specified, SSH public keys will use options set in the *dropbear_initramfs__authorized_keys_options* variable. To override this variable for a particular entry, set the `item.options` parameter as empty string or list.

The specified SSH key options are applied to all keys specified in the `item.sshkeys` parameter in this specific entry. To use different key options for different SSH keys, specify them in separate entries on the list.

key_options Optional. Additional set of options to add to the SSH public keys. This can be used with `item.options` parameter to easily combine a list of options from another variable with a custom additional options.

exclusive Optional, boolean. If defined and `True` , the role will remove all other SSH public keys and set only the SSH public keys defined by `item.sshkeys` .

state Optional. If undefined or `present` , the SSH public keys specified in the `item.sshkeys` parameter will be added. If `absent` , the specified SSH public keys will be removed.

Examples

Set SSH keys from a file on the Ansible Controller as the only allowed keys for remote initramfs login:

```
dropbear_initramfs__authorized_keys:
  - sshkeys: '{{ lookup("file", "/path/to/admin23.pub") }}'
    exclusive: True
```

Ensure that given SSH public keys are allowed for remote initramfs login:

```
dropbear_initramfs__group_authorized_keys:
  - sshkeys: [ 'ssh-rsa AAAAB3NzaC1yc2EAAAA...', 'ssh-rsa AAAAB3NzaC1yc2EAAAA...' ]
```

Related projects

- FDEunlock – Check and unlock full disk encrypted systems via ssh
- Mandos – System for allowing servers with encrypted root file systems to reboot unattended and/or remotely
- Comparison of Mandos and FDEunlock

The following Ansible roles do the same thing as this role and have been deprecated by this role:

- systemli.rootcrypto
- martin-v.sshpreluks

Refer to *Migrating from other Ansible roles* for details.

Design goals

- Don't overwrite global configuration files like `/etc/initramfs-tools/initramfs.conf` and similar as this can lead to problems like newer package versions trying to upgrade the file. `/etc/initramfs-tools/conf.d` and other `*.d` directories are preferred and used for this.
- If additional kernel modules need to be loaded in the initramfs then this functionally should be added to the debops-contrib.kernel_module role. Note that all modules listed in `/etc/initramfs-tools/modules` are force loaded as can be read in `/usr/sbin/mkinitramfs`. An initramfs hook should be used instead of touching the `/etc/initramfs-tools/modules` file.

Copyright

Changelog

debops-contrib.dropbear_initramfs

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

Refer to the *Upgrade notes* when you intend to upgrade to a new release.

debops-contrib.dropbear_initramfs master - unreleased

debops-contrib.dropbear_initramfs v0.2.0 - 2017-03-31

Added

- Add IPv6 support. [ypid]

Changed

- Rename role to `debops-contrib.dropbear_initramfs`. [ypid]

- Changed license from AGPL-3.0 to GPL-3.0. [ypid]
- Major rewrite which breaks backwards compatibility and legacy support. Refer to *Upgrade from v0.1.0 to v0.2.0* for details. [ypid]
- Require Ansible version 2.1.4 or above. [ypid]

ypid.cryptsetup_remote_unlock v0.1.0 - 2016-07-18

Added

- Initial coding and design. Dates back to 2015-03-01. [ypid]

Upgrade notes

The upgrade notes only describe necessary changes that you might need to make to your setup in order to use a new role release. Refer to the *Changelog* for more details about what has changed.

Upgrade from v0.1.0 to v0.2.0

All inventory variables have been renamed so you might need to update your inventory. You will need to read the updated documentation and upgrade your inventory manually.

Migrating from other Ansible roles

This role tries to work for all common use cases and combine similar roles previously created by independent authors which basically do the same thing. Refer to Combine effords for details.

From martin-v.sshpreluks

All inventory variables have been renamed so you might need to update your inventory. You will need to read the role documentation and upgrade your inventory manually.

From systemli.rootcrypto

All inventory variables have been renamed so you might need to update your inventory. A subset of them can be automatically updated using this script:

```
#!/bin/bash
## Upgrade inventory variables for migration from systemli.rootcrypto to debops-
↪contrib.dropbear_initramfs.
## The script is idempotent.

git ls-files -z "$(git rev-parse --show-toplevel)" | xargs --null -I '{}' find '{}' -
↪type f -print0 \
 | xargs --null sed --in-place --regexp-extended '
     s/rootcrypto_network_device/dropbear_initramfs__network_device/g;
     s/rootcrypto_network_address/dropbear_initramfs__network_address/g;
     s/rootcrypto_network_netmask/dropbear_initramfs__network_netmask/g;
     s/rootcrypto_network_gateway/dropbear_initramfs__network_gateway/g;
   '
```

The script is bundled with this role under `./docs/scripts/migrate-from-systemli.rootcrypto-to-debops-contri` and can be invoked from there.

You will need to read the role documentation and upgrade your remaining inventory manually.

Ansible role: debops-contrib.etckeeper

Introduction

`debops-contrib.etckeeper` will install etckeeper which puts `/etc` under version control. To do this it hooks into the package management and from now on automatically commit changes to a local git repository under `/etc/.git`.

This makes it easy to see which changes are applied on a specific host and quickly revert them, if something breaks.

Installation

This role requires at least Ansible `v2.1.3`. To install it, run:

```
ansible-galaxy install debops-contrib.etckeeper
```

Getting started

Initial configuration

By default **`git`** is used as VCS. This can be changed by the inventory variables *`etckeeper__vcs`*.

Example inventory

```
## If you don't what to track hashed passwords.
etckeeper__gitignore_group:
  - 'shadow'
  - 'shadow-'
```

In Ansible's inventory.

Example playbook

Here's an example playbook that uses the `debops-contrib.etckeeper` role:

```
---

- name: Put /etc under version control using etckeeper
  hosts: [ 'debops_service_etckeeper' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops-contrib.etckeeper
      tags: [ 'role::etckeeper' ]
```

This playbooks is shipped with this role under `./docs/playbooks/etckeeper.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::etckeeper` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::etckeeper:vcs_config` Run tasks related to configuring VCS options.

debops-contrib.etckeeper default variables

Sections

Package management options

`etckeeper__highlevel_package_manager`

The high-level package manager that's being used. (**`apt`**, **`pacman-g2`**, **`yum`**, **`dnf`**, **`zypper`** etc). This will only be used when your distribution was not able to predefine this.

```
etckeeper__highlevel_package_manager: '{{ ansible_pkg_mgr }}'
```

`etckeeper__lowlevel_package_manager`

The low-level package manager that's being used. (**dpkg**, **rpm**, **pacman**, **pacman-g2**, etc) This will only be used when your distribution was not able to predefine this.

```
etckeeper__lowlevel_package_manager: |
  {{ etckeeper__highlevel_to_lowlevel_package_manager_mapping[etckeeper__highlevel_
↪package_manager]|d("") }}
```

Version control ignore list

etckeeper__ignore_role_list

Role defaults list of files and directories which should not be kept under version control.

```
etckeeper__ignore_role_list:
  ## There is no benefit in tracking Tor keys and it is a potential security␣
↪vulnerability.
  - 'tor/keys/'

  ## Same with SSH host keys.
  - 'ssh/ssh_host_*_key'

  - 'X11/xorg.conf.backup'

  ## Files are generated and managed by libvirt and it is believed that there
  ## is very little benefit in tracking these files.
  - 'apparmor.d/libvirt/*.files'

  - 'zfs/zpool.cache'
```

etckeeper__ignore_list

Global list of files and directories which should not be kept under version control.

```
etckeeper__ignore_list: []
```

etckeeper__ignore_host_group_list

Host group list of files and directories which should not be kept under version control.

```
etckeeper__ignore_host_group_list: []
```

etckeeper__ignore_host_list

Host list of files and directories which should not be kept under version control.

```
etckeeper__ignore_host_list: []
```

Version control options

etckeeper__vcs

Which VCS to use to version `/etc/` . Choices are:

- **git** (default)
- **hg**
- **bzr**

- **darcs**

Note that any other VCS than **git** has not really been tested. You might have to fix some bugs in this role when you want to use them.

```
etckeeper__vcs: 'git'
```

etckeeper__vcs_user

User for **etckeeper** to use in commits if no interactive user was detected. Defaults to an empty string which results in no changes regarding the user use by the VCS.

```
etckeeper__vcs_user: ''
```

etckeeper__vcs_email

Email address for **etckeeper** to use in commits if no interactive user was detected.

Example:

```
etckeeper__vcs_email: '{{ etckeeper__vcs_user + "@" + ansible_fqdn }}'
```

Defaults to an empty string which results in no changes regarding the email address use by the VCS.

```
etckeeper__vcs_email: ''
```

etckeeper__git_commit_options

Options passed to git commit when run by etckeeper.

```
etckeeper__git_commit_options: ''
```

etckeeper__hg_commit_options

Options passed to hg commit when run by etckeeper.

```
etckeeper__hg_commit_options: ''
```

etckeeper__bzr_commit_options

Options passed to bzr commit when run by etckeeper.

```
etckeeper__bzr_commit_options: ''
```

etckeeper__darcs_commit_options

Options passed to darcs record when run by etckeeper.

```
etckeeper__darcs_commit_options: '-a'
```

etckeeper__avoid_daily_autocommits

Uncomment to avoid **etckeeper** committing existing changes to `/etc` automatically once per day.

```
etckeeper__avoid_daily_autocommits: False
```

etckeeper__avoid_special_file_warning

Uncomment the following to avoid special file warning (the option is enabled automatically by cronjob regardless).

```
etckeeper__avoid_special_file_warning: False
```

etckeeper__avoid_commit_before_install

Uncomment the following to avoid special file warning (the option is enabled automatically by cronjob regardless).

```
etckeeper__avoid_commit_before_install: False
```

etckeeper__push_remote

To push each commit to a remote, put the name of the remote here. (eg, "origin" for git). Space-separated lists of multiple remotes also work (eg, "origin gitlab github" for git).

```
etckeeper__push_remote: ''
```

Copyright

```
debops-contrib.etckeeper - Put /etc under version control using etckeeper

Copyright (C) 2016 Robin Schneider <ypid@riseup.net>
Copyright (C) 2016 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.etckeeper

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.etckeeper v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]
- Added support to configure the user and email used by etckeeper. [joh6nn, ypid]

Changed

- Renamed `etckeeper_gitignore` to `etckeeper_ignore_list`, `etckeeper_gitignore_group` to `etckeeper_ignore_host_group_list`, `etckeeper_ignore_host_list` to `etckeeper_gitignore_host`. [ypid]
- Moved role default ignore list from `etckeeper_ignore_list` to it's own ignore list `etckeeper_ignore_role_list`. [ypid]
- Changed namespace from `etckeeper_` to `etckeeper__`. `etckeeper_[^_]` variables are hereby deprecated and you might need to update your inventory. This oneliner might come in handy to do this.

```
git ls-files | xargs sed --in-place --regexp-extended 's/etckeeper_([^_])/
↪etckeeper__\1/g'
```

  [ypid]

Ansible role: debops-contrib.firejail

Introduction

Firejail is a SUID program that reduces the risk of security breaches by restricting the running environment of untrusted applications using Linux namespaces and seccomp-bpf.

This Ansible role allows you to setup and configure Firejail.

Features

- Install Firejail from jessie-backports or other configured APT repositories. debops.apt can be used to enable Backports if needed.
- Sandbox programs system wide by placing a symlink to **`firejail`** into the `PATH` so that **`firejail`** can wrap program invocations and sandbox the invoked program using security profiles that Firejail ships or that the system administrator defines.

Installation

This role requires at least Ansible `v2.1.3`. To install it, run:

```
ansible-galaxy install debops-contrib.firejail
```

Note that this role uses features recently introduced in Jinja2, namely the *equalto* filter which was released with Jinja 2.8 and thus requires Jinja 2.8. If you use Debian Jessie, you can install it from Debian Jessie Backports.

Getting started

Example inventory

To setup and configure Firejail on a given host it should be included in the `debops_service_firejail` Ansible inventory group:

```
[debops_service_firejail]
hostname
```

Example playbook

Here's an example playbook that uses the `debops-contrib.firejail` role:

```
---

- name: Setup and configure Firejail
  hosts: [ 'debops_service_firejail' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops-contrib.firejail
      tags: [ 'role::firejail' ]
```

The playbooks is shipped with this role under `./docs/playbooks/firejail.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::firejail` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::firejail:pkgs` Tasks related to system package management like installing or removing packages.

`role::firejail:profile` Tasks related to Firejail security profile management like copying or removing profile files.

debops-contrib.firejail default variables

Sections

- *Program sandboxes*
- *Workaround for desktop files*

Packages and installation

firejail__base_packages

List of base packages to install.

```
firejail__base_packages:
  - 'firejail'
```

firejail__packages

List of optional global packages. This variable is intended to be used in Ansible's global inventory.

```
firejail__packages: []
```

firejail__group_packages

List of optional group packages. This variable is intended to be used in a host inventory group of Ansible (only one host group is supported).

```
firejail__group_packages: []
```

firejail__host_packages

List of optional host packages. This variable is intended to be used in the inventory of hosts.

```
firejail__host_packages: []
```

firejail__deploy_state

What is the desired state which this role should achieve? Possible options:

present Default. Ensure that Firejail is installed and configured as requested.

absent Ensure that Firejail is uninstalled and it's configuration is removed.

```
firejail__deploy_state: 'present'
```

System paths

firejail__config_path

Directory where the system wide Firejail configuration and profiles are stored.

```
firejail__config_path: '/etc/firejail'
```

firejail__program_file_path

File path of the **firejail** binary. When set to `auto` , the role tries to figure out the file path via the **which** command. Note that **which** is executed in the context of the root user who might have a different `PATH` variable then normal users.

To use **firejail** from another location, set:

```
firejail__program_file_path: '/usr/local/bin/firejail'
```

in your Ansible inventory.

```
firejail__program_file_path: 'auto'
```

firejail__system_local_bin_path

Directory in which to create the symlinks when enabling a profile system wide. This directory path must be included in the `PATH` variable before the directory which contains the real program so that the symlink pointing to **firejail** is used when users try to execute the program.

```
firejail__system_local_bin_path: '{{ ansible_local.root.bin
                                     if (ansible_local|d() and ansible_local.root|d()
↪and
                                         ansible_local.root.root|d())
                                     else "/usr/local/bin" }}'
```

Program sandboxes

Program sandboxes can be defined using dictionary variables on different inventory levels which are combined together.

For more details refer to *program_sandboxes* in the *Default variable details* section.

firejail__program_sandboxes

This variable is intended to be used in Ansible's global inventory.

```
firejail__program_sandboxes: {}
```

firejail__group_program_sandboxes

This variable is intended to be used in a host inventory group of Ansible (only one host group is supported).

```
firejail__group_program_sandboxes: {}
```

firejail__host_program_sandboxes

This variable is intended to be used in the inventory of hosts.

```
firejail__host_program_sandboxes: {}
```

firejail__role_program_sandboxes

Program sandbox definitions used/set internally by this role.

```
firejail__role_program_sandboxes:
  default:
    # The "default" profile is not intended to correspond to a program called
    # "default". Ensure that even if such a program exists, it will not be
    # sandboxed system wide without the role maintainers approving it first.
    system_wide_sandboxed: 'absent'
  ssh:
    # Might conflict with other programs using it. For example, Ansible and
    # BorgBackup did not work with this enabled system wide.
    system_wide_sandboxed: 'absent'
  tar:
```

```
    # Causes dpkg install tasks to fail.
    system_wide_sandboxed: 'absent'
  unrar:
    # Did not extract when run in sandbox.
    # `unp` seems unable to detect that `unrar` is installed when with the symlink.
    system_wide_sandboxed: 'absent'
  git:
    # Needed everywhere. Did not work well with zsh and does not work for root␣
↪because the profile uses --noroot.
    system_wide_sandboxed: 'absent'
```

firejail__combined_program_sandboxes

Combined dictionary of program sandboxes as it is used by the role. This defines the order in which dictionary keys might "mask" previous once.

```
firejail__combined_program_sandboxes: '{{
  firejail__role_program_sandboxes
  | combine(firejail__program_sandboxes)
  | combine(firejail__group_program_sandboxes)
  | combine(firejail__host_program_sandboxes) }}'
```

firejail__global_profiles_system_wide_sandboxed

Sandbox all programs for which Firejail ships profiles or which have otherwise been configured below *firejail__config_path* system wide using the method described in *item.system_wide_sandboxed*. This variable only applies when the program was not configured using *program_sandboxes*. For that case refer to *firejail__program_sandboxes_system_wide_sandboxed*.

```
firejail__global_profiles_system_wide_sandboxed: 'if_installed'
```

firejail__program_sandboxes_system_wide_sandboxed

Default value for *item.system_wide_sandboxed*.

```
firejail__program_sandboxes_system_wide_sandboxed: 'if_installed'
```

Workaround for desktop files

Some desktop files include a full path to the executable which would result in the program being executed without Firejail sandboxing it. For this, Firejail provides the **firecfg --fix** command which fixes those desktop files and saves them under `~/.local/share/applications/`.

This section provides variables which you should use to do this for the users.

Those variables have the same structure as the `users__accounts` onces from the debops.users role. This allows you to include all users you configured using the debops.users by putting this:

```
1 firejail__fix_for_users: '{{ users__accounts|d([]) }}'
2 firejail__group_fix_for_users: '{{ users__group_accounts|d([]) }}'
3 firejail__host_fix_for_users: '{{ users__host_accounts|d([]) }}'
```

into your global inventory.

firejail__fix_for_users

Global list of users for which the desktop files workaround should be applied. The list should contain a dictionary for each user with the username in the `name` key of the dictionary.

```
firejail__fix_for_users: []
```

firejail__group_fix_for_users

Host group list of users for which the desktop files workaround should be applied.

```
firejail__group_fix_for_users: []
```

firejail__host_fix_for_users

Host list of users for which the desktop files workaround should be applied.

```
firejail__host_fix_for_users: []
```

firejail__combined_fix_for_users

Combined list of users as it is used by the role.

```
firejail__combined_fix_for_users: '{{
  (firejail__fix_for_users       | list ) +
  (firejail__group_fix_for_users | list ) +
  (firejail__host_fix_for_users  | list ) }}'
```

firejail__ansible_log

Enable or disable Ansible task logging for *firejail__combined_fix_for_users* which might contains sensitive information. This variable should not be changed other than for debugging.

```
firejail__ansible_log: False
```

Default variable details

Some of debops-contrib.firejail default variables have more extensive configuration than simple strings or lists, here you can find documentation and examples for them.

- *program_sandboxes*

program_sandboxes

The *firejail__program_sandboxes* and similar dictionaries allow you to configure program sandboxes using Firejail profiles (*firejail-profile(5)*). The dictionary key is the program name, the value is a dictionary with the following supported keys:

system_wide_sandboxed Optional, string. Should the program be sandboxed with **firejail** for all users of the system by creating a symlink under /usr/local/bin/{{ item.key }} with the **firejail** program binary file path as target. The directory path where the symlink is being created/removed (/usr/local/bin/) can be changed via *firejail__system_local_bin_path* . This option relies on the feature of **firejail** to be called via a different file path which causes **firejail** to act as a wrapper around the real program.

These options are supported:

present The sandbox should be present system wide.

if_installed The sandbox should be present system wide but only if the program is installed (is found in `PATH`) on role run. This can be used to not make it look like the program is installed (by creating a symlink with the name in the `PATH`) and to avoid the case where a user tries to run the program and **firejail** complaining with "Error: cannot find the program in the path". If the program is not found, then the system wide sandbox will be made `absent` .

absent The sandbox should be absent system wide.

Defaults to *firejail__global_profiles_system_wide_sandboxed* . Refer to *firejail(1)* under "Desktop Integration" or Firejail 0.9.38 Release Announcement under "Symlink invocation".

profile Optional, dictionary. Use a provided profile by copying it from the Ansible controller into the *firejail__config_path* directory of the remote system using the Ansible copy module. `profile` is basically just passed to the module. Refer to it's documentation for details with the exception that the `state` parameter is handled properly. `state` defaults to `present` but can be set to `absent` which will cause the profile on the remote systems to become absent. Refer to *Examples for providing additional profiles* for how this can be used.

Examples for sandboxing additional programs

Sandbox the given programs on all hosts even if Firejail does not yet ship with a profile for them:

```
firejail__program_sandboxes:
  jq: {}
  my_cool_program:
    system_wide_sandboxed: 'present'
```

The symlink for **jq** will only be created if **jq** is installed. The symlink for **my_cool_program** will be created regardless whether it has been found in the `PATH` .

Example to exclude a program from being sandboxed

Depending on the value of *firejail__global_profiles_system_wide_sandboxed* , the role might sandbox all programs which are installed and for which security profiles are defined. Check out the following example in case you want to exclude programs from being sandboxed system wide:

```
firejail__program_sandboxes:
  less:
    # Less can't possibly have an issue with parsing untrusted input (TM).
    # I know what I am doing! Don't sandbox it!
    system_wide_sandboxed: 'absent'
```

Examples for providing additional profiles

Copy Firejail security profiles from the Ansible controller to all remote systems:

```
firejail__program_sandboxes:
  smplayer:
    profile:
      src: '/home/user/.config/firejail/smplayer.profile'

      ## `content` can be used alternatively to `src` to provide the profile inlined
      ## (supports Jinja templating as usual):
      # content: |
```

```
      #   # {{ ansible_managed }}
      #   # smplyer security profile.
      #   noblacklist ${HOME}/.config/smplayer
      #   # And so on.

      ## `state` can be used to make the profile absent:
      # state: 'absent'
```

This will create `/etc/firejail/smplayer.profile` on all remote systems.

Ansible integration and role design

Design goals

- **`firecfg`** is not being used to enabling/disabling *system wide sandboxes*. This is done by the role itself to have more control over the process.

  Note that running **`firecfg`** without arguments will have a similar affect than when using this role with *`firejail__global_profiles_system_wide_sandboxed`* set to if_installed but without all the other logic of this role. So **`firecfg`** might change settings done by the role. You can rerun the role to ensure that the state defined by Ansible is present on the system.

Alternative roles

As of 2016-10-31 ypid was aware of two alternative Ansible roles for Firejail:

- gbraad.firejail, targets Fedora, has a major security issue: Installation can be trivially MITMed leading to the system being comprised. Only deals with installing the Firejail suite itself.
- Firejail role by aaaaaaaaaaaaaaaaaaaa1, targets system which use APT. Only deals with building and installing Firejail itself.

None of the existing roles where found to be a suitable start for this role so it has been designed and written from scratch.

Copyright

```
debops-contrib.firejail - Setup and configure Firejail

Copyright (C) 2016-2017 Robin Schneider <ypid@riseup.net>
Copyright (C) 2016-2017 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.firejail

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.firejail v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]
- Support to fix desktop files using **`firecfg --fix`**. Please use this to make sure that programs are actually sandboxed! [ypid]

Changed

- Optimized performance by only checking if programs are installed when this actually matters (when *item.system_wide_sandboxed* is `if_installed`). [ypid]

Fixed

- The role did not handle *`firejail__global_profiles_system_wide_sandboxed`* set to `absent` correctly and instead (was handled as it was set to `present`). [ypid]
- Note in the documentation that this role requires Jinja 2.8 or later. [ypid]
- Don't sandbox `tar` , `unrar` and **`git`** by default. [ypid]

Ansible role: debops-contrib.foodsoft

Introduction

The `debops-contrib.foodsoft` role allows to setup your own Foodsoft instance. Foodsoft is a web-based software to manage a non-profit food coop (product catalog, ordering, accounting, job scheduling).

The role is based on the following documentation:

- Deployment (Debian)
- SETUP_DEVELOPMENT.md
- Dockerfile

Note that the author of this role currently does not have an production deployment using this role but he will try to maintain this role as good as possible. He just wrote this role just for fun to try out Foodsoft.

Installation

This role requires at least Ansible `v2.1.5`. To install it, run:

```
ansible-galaxy install debops-contrib.foodsoft
```

Getting started

Database support

It is recommended that you install a database server. You can install one on the same host as Foodsoft or choose a different host:

```
[debops_service_mariadb_server]
hostname
```

In case you chose a different host, you will need to specify which of your database servers the Foodsoft instance should use by specifying the database server host as *`foodsoft__database_server`*.

Webserver support

Currently, only Nginx is supported using debops.nginx.

You will need to install Nginx with Passenger support by setting:

```
nginx_flavor: 'passenger'
```

in your inventory.

Example inventory

To manage Foodsoft on a given host or set of hosts, they need to be added to the `[debops_service_foodsoft_nginx]` Ansible group in the inventory:

```
[debops_service_foodsoft_nginx]
hostname

[debops_service_mariadb_server]
hostname
```

Example playbook

Ansible playbook that uses the `debops-contrib.foodsoft` role together with debops.nginx:

```
---

- name: Setup and manage Foodsoft with Nginx as webserver
  hosts: [ 'debops_service_foodsoft_nginx' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops.apt_preferences
      tags: [ 'role::apt_preferences' ]
      apt_preferences__dependent_list:
        - '{{ ruby__apt_preferences__dependent_list }}'
        - '{{ nginx__apt_preferences__dependent_list }}'

    - role: debops.ferm
      tags: [ 'role::ferm' ]
      ferm__dependent_rules:
        - '{{ nginx__ferm__dependent_rules }}'

    - role: debops.mariadb
      tags: [ 'role::mariadb' ]
      mariadb__dependent_databases: '{{ foodsoft__mariadb__dependent_databases }}'
      mariadb__dependent_users: '{{ foodsoft__mariadb__dependent_users }}'
      when: (foodsoft__database == 'mariadb')

    - role: debops.ruby
      tags: [ 'role::ruby' ]

    - role: debops.nginx
      tags: [ 'role::nginx' ]
      nginx__dependent_servers:
        - '{{ foodsoft__nginx__dependent_servers }}'

    - role: debops-contrib.foodsoft
      tags: [ 'role::foodsoft' ]
```

The playbook is shipped with this role under `./docs/playbooks/` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::foodsoft` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

role::foodsoft:pkgs Tasks related to system package management like installing or removing packages.

role::foodsoft:config Tasks related to configuring Foodsoft.

debops-contrib.foodsoft default variables

Sections

System packages

foodsoft__base_packages

List of base packages required by Foodsoft.

```
foodsoft__base_packages:
  - '{{ ["ruby2.0", "ruby2.0-dev"] if (ansible_distribution == "Ubuntu" and ansible_
↪distribution_release in ["trusty"]) else [] }}'

  - 'libcurl3-dev'
  - 'libxml2-dev'
  - 'libxslt-dev'
  - 'libffi-dev'
  - 'libreadline-dev'

  ## charlock_holmes
  - 'g++'
  ## https://stackoverflow.com/questions/15553792/error-installing-charlock-holmes-
↪error-installing-gitlab/15556110#15556110
  - 'libicu-dev'

  ## RMagick
  - 'pkg-config'
  - 'libmagickwand-dev'
  - 'ruby-magic'
  - 'libmagic-dev'

  ## sqlite3
  - '{{ ["libsqlite3-dev"] if (foodsoft__database in ["sqlite"]) else [] }}'

  ## mysql2
```

```
  - '{{ ["libmysqlclient-dev", "libmariadbd-dev"] if (foodsoft__database in ["mariadb
↪"]) else [] }}'

  ## Install via gem
  # - 'ruby-charlock-holmes'
  # - 'ruby-rmagick'
```

foodsoft__deploy_state

What is the desired state which this role should achieve? Possible options:

present Default. Ensure that Foodsoft is installed and configured as requested.

absent Ensure that Foodsoft is uninstalled and it's configuration is removed.

purged Same as `absent` but additionally also ensures that the database and other persistent data is removed.

```
foodsoft__deploy_state: 'present'
```

FQDN and DNS addresses

foodsoft__fqdn

The Fully Qualified Domain Name of the Foodsoft instance. This address is used to configure the webserver frontend.

```
foodsoft__fqdn: 'foodsoft.{{ foodsoft__domain }}'
```

foodsoft__domain

Domain that will be configured for the Foodsoft instance.

```
foodsoft__domain: '{{ ansible_local.core.domain
                      if (ansible_local|d() and ansible_local.core|d() and
                          ansible_local.core.domain|d())
                      else (ansible_domain if ansible_domain else ansible_hostname) }}
↪'
```

Database configuration

foodsoft__database

Autodetected variable containing the database management system which should be used. The supported and tested option is `mariadb` .

Refer to *Getting started* for details.

```
foodsoft__database: '{{ ansible_local.foodsoft.database
                        if (ansible_local|d() and ansible_local.foodsoft|d() and
                            ansible_local.foodsoft.database|d())
                        else ("mariadb"
                              if (ansible_local|d() and ansible_local.mariadb is␣
↪defined)
                              else ("postgresql"
                                    if (ansible_local|d() and ansible_local.
↪postgresql is defined)
                                    else "no-database-detected")) }}'
```

foodsoft__database_server

FQDN of the database server. It will be configured by the debops.mariadb or debops.postgresql role.

```
foodsoft__database_server: '{{ ansible_local[foodsoft__database].server }}'
```

foodsoft__database_port

Port database is listening on.

```
foodsoft__database_port: '{{ ansible_local[foodsoft__database].port }}'
```

foodsoft__database_name

Name of the database to use for Foodsoft.

```
foodsoft__database_name: 'foodsoft'
```

foodsoft__database_user

Database user to use for Foodsoft.

```
foodsoft__database_user: 'foodsoft'
```

foodsoft__database_password_path

Path to database password file.

```
foodsoft__database_password_path: '{{ secret + "/" + foodsoft__database + "/"
                                    + ansible_local[foodsoft__database].delegate_to
                                    + (("/" + ansible_local[foodsoft__database].
↪port)
                                       if (foodsoft__database == "postgresql")
                                       else "")
                                    + "/credentials/" + foodsoft__database_user + "/
↪password" }}'
```

foodsoft__database_password

Database password for Foodsoft.

```
foodsoft__database_password: '{{ lookup("password", foodsoft__database_password_path␣
↪+ " length=48 chars=ascii_letters,digits,.:-_") }}'
```

foodsoft__database_name_map

Database name mapping from the names as used in DebOps to Ruby database adapter names.

```
foodsoft__database_name_map:
  'mariadb': 'mysql2'
  'sqlite': 'sqlite3'

  # Legacy:
  'mysql': 'mysql2'
```

foodsoft__database_config

Database configuration for Foodsoft. Written to `config/database.yml`.

```
foodsoft__database_config:
  production:
    adapter: '{{ foodsoft__database_name_map[foodsoft__database] }}'
    # socket: '/tmp/mysql.sock'
```

```
    host: '{{ foodsoft__database_server }}'
    reconnect: False
    pool: 5
    username: '{{ foodsoft__database_user }}'
    password: '{{ foodsoft__database_password }}'
    database: '{{ foodsoft__database_name }}'
    encoding: 'utf8'
```

Webserver configuration

foodsoft__webserver

Autodetected variable containing the webserver which should be used. Currently only Nginx is supported.

```
foodsoft__webserver: '{{ ansible_local.foodsoft.webserver
                         if (ansible_local|d() and ansible_local.foodsoft|d() and
                             ansible_local.foodsoft.webserver|d())
                         else ("nginx"
                               if (ansible_local|d() and ansible_local.nginx|d() and␣
↪ansible_local.nginx.enabled|d()|bool)
                                else ("apache"
                                      if (ansible_local|d() and ansible_local.
↪apache|d() and ansible_local.apache.enabled|d()|bool)
                                      else "no-webserver-detected")) }}'
```

foodsoft__webserver_user

Name of the webserver user account which will be granted read only access to the Foodsoft application directory.

```
foodsoft__webserver_user: '{{ ansible_local.nginx.user
                              if (ansible_local|d() and ansible_local.nginx|d() and
                                  ansible_local.nginx.user|d())
                               else "www-data" }}'
```

Directory paths

foodsoft__home_path

The Foodsoft system account home directory.

```
foodsoft__home_path: '{{ (ansible_local.nginx.www
                          if (ansible_local|d() and ansible_local.nginx|d()
                              and ansible_local.nginx.www|d())
                          else "/srv/www") + "/" + foodsoft__user }}'
```

foodsoft__www_path

Base web root directory for Foodsoft.

```
foodsoft__www_path: '{{ foodsoft__git_dest + "/public" }}'
```

System user and group

foodsoft__user

System UNIX account used by the Foodsoft.

```
foodsoft__user: 'foodsoft'
```

foodsoft__group

System UNIX group used by the Foodsoft.

```
foodsoft__group: 'foodsoft'
```

foodsoft__gecos

Contents of the GECOS field set for the Foodsoft account.

```
foodsoft__gecos: 'Foodsoft'
```

foodsoft__shell

The default shell set on the foodsoft account.

```
foodsoft__shell: '/usr/sbin/nologin'
```

Foodsoft sources and deployment

foodsoft__git_repo

The URI of the Foodsoft git source repository. There is also https://github.com/foodcoop-adam/foodsoft.git which you can choose alternatively.

```
foodsoft__git_repo: 'https://github.com/foodcoops/foodsoft.git'
```

foodsoft__git_version

The git branch or tag which will be installed. Defaults to the commit hash of latest release (4.5.1). This is done because Foodsoft development is not cryptographically signed and this role wants to comply with the DebOps Software Source Policy.

```
foodsoft__git_version: 'a7b6b0c803ca4a79ddab7cea92545b8cc188f952'
```

foodsoft__git_dest

Path where the Foodsoft sources will be checked out (installation path).

```
foodsoft__git_dest: '{{ foodsoft__home_path + "/foodcoops-foodsoft" }}'
```

foodsoft__git_update

Should new revisions be retrieved from the origin repository?

```
foodsoft__git_update: True
```

foodsoft__bundler_exclude_groups

Don't install the Gems in the listed groups.

```
foodsoft__bundler_exclude_groups:
  - 'test'

  ## Contains SQLite gem.
  - 'development'
```

Foodsoft configuration

foodsoft__name

Name of this Foodsoft instance.

```
foodsoft__name: 'Foodcoop'
```

foodsoft__contact

Foodcoop contact information (used for FAX messages).

```
foodsoft__contact:
  street: 'Grüne Straße 23'
  zip_code: '12323'
  city: 'Berlin'
  country: 'Deutschland'
  email: '{{ foodsoft__email_sender }}'
  phone: '030 323 232323'
```

foodsoft__default_scope

If *foodsoft__multi_coop_install* is true you have to use a coop name, which you you wanna be selected by default.

```
foodsoft__default_scope: 'f'
```

foodsoft__homepage

Homepage URL.

```
foodsoft__homepage: 'https://{{ foodsoft__fqdn }}/{{ foodsoft__default_scope }}'
```

foodsoft__page_footer

Page footer (html allowed). Default is a Foodsoft footer. Set to the word "blank" for no footer. If unchanged, the default footer of Foodsoft will be used.

```
foodsoft__page_footer: '<a href="{{ foodsoft__homepage }}/">{{ foodsoft__name }}</a>,␣
↪setup by <a href="https://debops.org/">DebOps</a>.'
```

foodsoft__email_sender

Email address to be used as sender.

```
foodsoft__email_sender: 'foodsoft@{{ foodsoft__domain }}'
```

foodsoft__error_recipients

Email address to be used as sender.

```
foodsoft__error_recipients:
  - 'admin@{{ foodsoft__domain }}'
```

foodsoft__multi_coop_install

If you wanna serve more than one Foodcoop with one installation. Don't forget to setup databases for each Foodcoop. See also MULTI_COOP_INSTALL.

```
foodsoft__multi_coop_install: False
```

foodsoft__upstream_config

Configuration as defined by upstream Foodcoop in `config/app_config.yml.SAMPLE`.

```
foodsoft__upstream_config: '{{ lookup("file", "vars/sample_app_config.yml")|from_yaml␣
↪}}'
```

foodsoft__role_config

This dict is managed by the role itself, controlled by other default variables.

```
foodsoft__role_config:

  multi_coop_install: '{{ foodsoft__multi_coop_install|bool }}'
  default_scope: '{{ foodsoft__default_scope }}'
  name: '{{ foodsoft__name }}'
  contact: '{{ foodsoft__contact }}'
  homepage: '{{ foodsoft__homepage }}'

  # Default timezone, e.g. UTC, Amsterdam, Berlin, etc.
  # FIXME: Foodsoft/Ruby seem to expect a different format than what debops.core␣
↪returns.
  # Potentially splitting at "/" and returning the second half of the string
  # would do the job but that would need testing.
  # Change manually if needed.
  # time_zone: '{{ ansible_local.timezone if (ansible_local|d() and ansible_local.
↪timezone|d()) else "Etc/UTC" }}'

  # Page footer (html allowed). Default is a Foodsoft footer. Set to `blank` for no␣
↪footer.
  page_footer: '{{ foodsoft__page_footer }}'

  email_sender: '{{ foodsoft__email_sender }}'

  # Config for the exception_notification plugin.
  notification:
    error_recipients: '{{ foodsoft__error_recipients }}'
    sender_address: '"Foodsoft Error" <{{ foodsoft__email_sender }}>'
    email_prefix: "[Foodsoft]"
```

foodsoft__config

This dict is intended to be used in Ansible's global inventory as needed.

```
foodsoft__config: {}
```

foodsoft__group_config

This dict is intended to be used in a host inventory group of Ansible (only one host group is supported) as needed.

```
foodsoft__group_config: {}
```

foodsoft__host_config

This dict is intended to be used in the inventory of hosts as needed.

```
foodsoft__host_config: {}
```

foodsoft__combined_config

The configuration written to `config/app_config.yml` .

```
foodsoft__combined_config: '{{ foodsoft__upstream_config.default
                               | combine(foodsoft__role_config)
                               | combine(foodsoft__config)
                               | combine(foodsoft__group_config)
                               | combine(foodsoft__host_config) }}'
```

Configuration for other Ansible roles

foodsoft__mariadb__dependent_databases

Configuration of the foodsoft database managed by the debops.mariadb role.

```
foodsoft__mariadb__dependent_databases:

  - database: '{{ foodsoft__database_name }}'
    state: '{{ "present" if (foodsoft__deploy_state != "purged") else "absent" }}'
```

foodsoft__mariadb__dependent_users

Configuration of the foodsoft database user managed by the debops.mariadb role.

```
foodsoft__mariadb__dependent_users:

  - database: '{{ foodsoft__database_name }}'
    state: '{{ "present" if (foodsoft__deploy_state == "present") else "absent" }}'
    user: '{{ foodsoft__database_user }}'
    password: '{{ foodsoft__database_password }}'
```

foodsoft__nginx__dependent_servers

Configuration of the foodsoft nginx server, used by the debops.nginx Ansible role.

```
foodsoft__nginx__dependent_servers:

  - name: '{{ foodsoft__fqdn }}'
    filename: 'debops.foodsoft'
    by_role: 'debops-contrib.foodsoft'
    enabled: True
    type: 'rails'
    root: '{{ foodsoft__www_path }}'

    # Foodsoft manages this by itself by default.
    # TOOD: Should probably be disabled in Foodsoft so that DebOps can manage it.
    hsts_enabled: False
    frame_options: False
    content_type_options: False
    xss_protection: '{{ omit }}'

    # Phusion Passenger options
    passenger_user: '{{ foodsoft__user }}'
    passenger_group: '{{ foodsoft__group }}'
```

Copyright

Changelog

debops-contrib.foodsoft

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.foodsoft master - unreleased

Changed

- Full role rewrite. All inventory variables have been renamed. When updating, handle it like a different role and refer to the documentation. [ypid]

ypid.foodsoft v0.1.0 - 2015-11-01

Added

- Initial coding and design. [ypid]

Ansible role: debops-contrib.fuse

Introduction

The `debops-contrib.fuse` role allows you to install and configure Filesystem in Userspace (FUSE).

Installation

This role requires at least Ansible `v2.1.3`. To install it, run:

```
ansible-galaxy install debops-contrib.fuse
```

Getting started

Example inventory

To manage and configure FUSE on a given host it should be included in the `debops_service_fuse` Ansible inventory group:

```
[debops_service_fuse]
hostname
```

Example playbook

Here's an example playbook that uses the `debops-contrib.fuse` role:

```
---

- name: Install and configure Filesystem in Userspace (FUSE)
  hosts: [ 'debops_service_fuse' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops-contrib.fuse
      tags: [ 'role::fuse' ]
```

This playbooks is shipped with this role under `./docs/playbooks/fuse.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

role::fuse Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

debops-contrib.fuse default variables

Sections

Required packages

fuse_base_packages

List of base packages to install.

```
fuse_base_packages: [ 'fuse' ]
```

Fuse options

fuse_mount_max

Set the maximum number of FUSE mounts allowed to non-root users. Set to `default` to use the number chosen by your distribution which is 1000 in Debian Jessie.

```
fuse_mount_max: 'default'
```

fuse_user_allow_other

Allow non-root users to specify the allow_other or allow_root mount options.

```
fuse_user_allow_other: False
```

Fuse hardening

fuse_restrict_access

Should access to fuse and `/dev/fuse` be restricted to root and the members of the fuse group? Debian used to have a group called fuse and users which should be allowed to use FUSE needed to be in that group. As of Debian Jessie, no group is being created by default and every user has access to fuse. https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=733312 This was done to make it work by default with other packages which are based on FUSE.

```
fuse_restrict_access: False
```

fuse_group

Name of the system group ob `/dev/fuse`. Only users who are members of the *`fuse_group`* and `root` are allowed to use FUSE when *`fuse_restrict_access`* is `True`.

```
fuse_group: 'fuse'
```

fuse_permissions

Unix permissions of `/dev/fuse`. It defaults to `0600` so that only the file owner (`root`) and users in the *fuse_group* have access to FUSE.

```
fuse_permissions: '0660'
```

fuse_users

Which users should be allowed to use FUSE? Only takes affect when *fuse_restrict_access* is `True`. This variable is intended to be used in Ansible's global inventory.

```
fuse_users: []
```

fuse_users_host_group

Which users should be allowed to use FUSE? Only takes affect when *fuse_restrict_access* is `True`. This variable is intended to be used in a host inventory group of Ansible (only one host group is supported).

```
fuse_users_host_group: []
```

fuse_users_host

Which users should be allowed to use FUSE? Only takes affect when *fuse_restrict_access* is `True`. This variable is intended to be used in the inventory of hosts.

```
fuse_users_host: []
```

Copyright

Changelog

debops-contrib.fuse

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.fuse v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]

Ansible role: debops-contrib.gdnsd

Introduction

gdnsd is a powerful Authoritative-only DNS server with some advanced features such as geographic (or other sorts of) balancing, redirection, wighting and service-state-conscious failover at the DNS layer.

The `debops-contrib.gdnsd` Ansible role installs and configures the name service and is able to generate zone files from name records defined in the Ansible inventory and properly increase the serial on zone updates.

Installation

This role requires at least Ansible `v1.9.0`. To install it run:

```
user@host:~$ ansible-galaxy install debops-contrib.gdnsd
```

Getting started

DNS zone configuration

By default the role will create a forward and reverse zone defining the host as primary name server and add a `A` and `PTR` record for the host to the zone files. This configuration can be overwritten by defining *`gdnsd__zones`*.

Example inventory

The `debops-contrib.gdnsd` role can be included in your Ansible setup by assigning the DNS host(s) to a custom inventory group such as `gdnsd_service`. E.g.:

```
[debops_gdnsd_service]
hostname
```

Example playbook

Here's a minimal example Ansible playbook that uses the `debops-contrib.gdnsd` role:

```
---

- name: Manage gdnsd authoritative DNS servers
  hosts: debops_gdnsd_service
  become: True

  roles:
    - role: debops-contrib.gdnsd
      tags: [ 'role::gdnsd' ]
```

Default variables

Sections

Basic options

gdnsd__packages

gdnsd packages to install.

```
gdnsd__packages: [ 'gdnsd' ]
```

gdsnd__listen

List of local IP addresses (and ports) where gdsnd should listen for DNS requests. If emtpy, it will listen on all interfaces on port 53.

```
gdnsd__listen: []
```

Zone configuration

gdnsd__zones

List of DNS zones. For a detailed explanation see *gdnsd__zones*.

```
gdnsd__zones: []
```

gdnsd__ttl

Default TTL for zone entries.

```
gdnsd__ttl: 86400
```

gdnsd__mailbox

Mailbox name of the person responsible for this zone.

```
gdnsd__mailbox: 'hostmaster'
```

gdnsd__refresh

Time interval before the zone should be refreshed.

```
gdnsd__refresh: '3h'
```

gdnsd__retry

Time interval that should elapse before a failed refresh should be retried.

```
gdnsd__retry: '1h'
```

gdnsd__expire

Specifies how long zone data is considered valid in case the zone cannot be refreshed from the primary name server.

```
gdnsd__expire: '1w'
```

gdnsd__negative_cache

Negative caching TTL.

```
gdnsd__negative_cache: '1h'
```

gdnsd__reverse_zones

Support reverse zones. Set this to `False` in case reverse name lookup should not be supported.

```
gdnsd__reverse_zones: True
```

gdnsd__default_reverse_zone:

Default reverse zone. This value is used in case *network* is not specified in the zone definition and *gdnsd__reverse_zones* is enabled.

```
gdnsd__default_reverse_zone: '{{ ansible_default_ipv4.address.split(".")[:3] |␣
↪reverse | join(".") }}.in-addr.arpa'
```

Default variable details

Some of `debops-contrib.gdnsd` default variables have more extensive configuration than simple strings or lists, here you can find documentation and examples for them.

- *gdnsd__zones*

gdnsd__zones

This list is used to configure the DNS zones which are served by **gdnsd**. Each list element corresponds to zone entry which is a YAML dictionary with the following parameters:

domain Domain name of the DNS zone. Required, if this is a forward zone.

reverse_zone Optional. Set this to `True` to mark zone entry as reverse zone. Defaults to `False`.

reverse_network Network definition for reverse zone. Required, if this is a reverse zone.

primary_nameserver Optional. DNS name of the primary name server for this zone which is added to the zone's SOA record. Must be a FQDN. Defaults to the host name where **gdnsd** is installed.

mailbox Optional. Mail address of the person in charge of the zone. Defaults to `hostmaster@{{ item.domain }}`.

refresh Optional. Time interval before the zone should be refreshed. Defaults to *gdnsd__refresh*.

retry Optional. Time interval that should elapse before a failed refresh should be retried. Defaults to *gdnsd__retry*.

expire Optional. Specifies how long zone data is considered valid in case the zone cannot be refreshed from the primary name server. Defaults to *gdnsd__expire*.

negative_cache Optional. Negative caching TTL. Defaults to *gdnsd__negative_cache*.

nameservers Optional. List of authoritative name servers for this zone. For each entry a NS record will be added to the zone file. By default only the host running **gdnsd** will be set as authoritative name server.

records Optional. List of DNS records for this zone. Each records is defined as a YAML dictionary with the following properties:

name Required (except for `item.type: MX`). Record name. Depending on the record type, this is e.g. a host name.

type Optional. Record type. Supported are `A`, `CNAME`, `MX`, `SRV` and `TXT`. Defaults to `A` record.

do_reverse Optional. If `item.type` is `A`, add a reverse zone entry for this record. Defaults to `True` if *gdnsd__reverse_zones* is `True`.

target Required. Resource data which is served when querying the record. Depending on the record type this is e.g. a host address.

ttl Optional. Individual record TTL.

preference Optional. Preference given to this record among others with the same data. Lower values are preferred. Only valid for `MX` and `SRV` record types. Defaults to `5`.

weight Optional. A server selection mechanism. The weight field specifies a relative weight for entries with the same preference. Larger weights should be given a proportionately higher probability of being selected. Only valid for `SRV` record type. Defaults to `0`.

port Required. The port on this target host of this service. Only valid for `SRV` record type.

Copyright

```
debops-contrib.gdnsd - Manage gdnsd, an authoritative-only DNS server

Copyright (C) 2016 Reto Gantenbein <reto.gantenbein@linuxmonk.ch>
Copyright (C) 2015-2016 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
```

```
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.gdnsd

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ganto.

debops-contrib.gdnsd master - unreleased

Added

- Initial release. [ganto]

Ansible role: debops-contrib.homeassistant

Introduction

The `debops-contrib.homeassistant` role allows you to setup and manage Home Assistant. Home Assistant is a home automation platform running on Python 3. It is able to track and control all devices at home and offer a platform for automating control.

Installation

This role requires at least Ansible `v2.2.2` . To install it, run:

```
ansible-galaxy install debops-contrib.homeassistant
```

Getting started

Example inventory

To setup and manage Home Assistant on a given host or set of hosts, they need to be added one of the `[debops_service_homeassistant.*]` Ansible groups in the inventory depending on your way of deployment:

```
[debops_service_homeassistant_nginx]
hostname
```

Example playbook

Ansible playbook that uses the `debops-contrib.homeassistant` role and does not setup a reverse proxy:

```
---

- name: Setup and manage Home Assistant
  hosts: [ 'debops_service_homeassistant' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops-contrib.homeassistant
      tags: [ 'role::homeassistant' ]
```

Ansible playbook that uses the `debops-contrib.homeassistant` role together with debops.nginx as reverse proxy:

```
---

- name: Setup and manage Home Assistant with Nginx as reverse proxy
  hosts: [ 'debops_service_homeassistant_nginx' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops.apt_preferences
      tags: [ 'role::apt_preferences' ]
      apt_preferences__dependent_list:
        - '{{ nginx__apt_preferences__dependent_list }}'

    - role: debops.ferm
      tags: [ 'role::ferm' ]
      ferm__dependent_rules:
        - '{{ nginx__ferm__dependent_rules }}'

    - role: debops.nginx
      tags: [ 'role::nginx' ]
      nginx__dependent_maps:
        - '{{ homeassistant__nginx__dependent_maps }}'
```

```
      nginx__dependent_upstreams:
        - '{{ homeassistant__nginx__dependent_upstreams }}'
      nginx__dependent_servers:
        - '{{ homeassistant__nginx__dependent_servers }}'

    - role: debops-contrib.homeassistant
      tags: [ 'role::homeassistant' ]
```

These playbooks are shipped with this role under `./docs/playbooks/` from which you can symlink them to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::homeassistant` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::homeassistant:pkgs` Tasks related to system package management like installing or removing packages.

debops-contrib.homeassistant default variables

Sections

Packages and installation

`homeassistant__base_packages`

List of base packages to install.

```
homeassistant__base_packages:
  - 'python3-dev'
  - 'python3-pip'

  - '{{ ("python3-virtualenv"
         if (ansible_distribution_release in [ "wheezy", "precise" ])
```

```
         else "virtualenv")
       if (homeassistant__virtualenv|bool and ansible_distribution_release not in [
↪"trusty"])
       else [] }}'
```

homeassistant__packages

List of additional packages to install.

```
homeassistant__packages: []
```

homeassistant__dependency_python_packages

List of Home Assistant core dependency packages. This refers to Debian system packages and not Python packages.

```
homeassistant__dependency_python_packages:
  - 'python3-requests'
  - 'python3-yaml'
  - 'python3-tz'
  - 'python3-jinja2'
  - 'python3-voluptuous'

  ## Not available in Debian jessie. Debian stretch ships with Python 3.5 which
  # should eliminate the need for it.
  # - 'python3-typing'

  - '{{ ["python3-aiohttp"]
       if (ansible_distribution_release not in ["trusty"])
       else [] }}'
  - '{{ ["python3-async-timeout"]
       if (ansible_distribution == "Debian" and ansible_distribution_major_
↪version|int >= 9)
       else [] }}'
  - 'python3-chardet'
```

homeassistant__optional_python_packages

List of optional packages. This refers to Debian system packages and not Python packages.

```
homeassistant__optional_python_packages:
  - '{{ ["python3-colorlog"]
       if (ansible_distribution_release not in ["trusty"])
       else [] }}'
  - 'libffi-dev'
  - 'libssl-dev'
  - 'python3-crypto'
  - 'python3-cryptography'
  - 'python3-pyparsing'
  - 'python3-appdirs'
```

homeassistant__combined_packages

List of all system packages which will be installed by the role.

```
homeassistant__combined_packages: '{{ (
      homeassistant__base_packages +
      homeassistant__packages +
      (homeassistant__dependency_python_packages if (not homeassistant__
↪virtualenv|bool) else []) +
```

```
      (homeassistant__optional_python_packages if (not homeassistant__
↪virtualenv|bool) else [])
    ) | unique | sort }}'
```

homeassistant__deploy_state

What is the desired state which this role should achieve? Possible options:

present Default. Ensure that Home Assistant is installed and configured as requested.

absent Ensure that Home Assistant is uninstalled. Not fully supported yet.

```
homeassistant__deploy_state: 'present'
```

FQDN and DNS addresses

homeassistant__fqdn

The Fully Qualified Domain Name of the Home Assistant instance. This address is used to configure the webserver frontend.

```
homeassistant__fqdn: 'ha.{{ homeassistant__domain }}'
```

homeassistant__domain

Domain that will be configured for the Home Assistant instance.

```
homeassistant__domain: '{{ ansible_local.core.domain
                           if (ansible_local|d() and ansible_local.core|d() and
                               ansible_local.core.domain|d())
                           else (ansible_domain if ansible_domain else ansible_
↪hostname) }}'
```

Reverse proxy configuration

homeassistant__verify_client_certificate

Should a valid client certificate be required to access Home Assistant?

```
homeassistant__verify_client_certificate: False
```

Directory paths

homeassistant__home_path

The Home Assistant system account home directory.

```
homeassistant__home_path: '{{ (ansible_local.root.home
                               if (ansible_local|d() and ansible_local.root|d() and
                                   ansible_local.root.home|d())
                               else "/var/local") + "/" + homeassistant__user }}'
```

homeassistant__virtualenv_path

Path to the virtualenv where Home Assistant will be installed.

```
homeassistant__virtualenv_path: '{{ homeassistant__home_path + "/prod_venv" }}'
```

System user and group

homeassistant__user

System UNIX account under which Home Assistant is run.

```
homeassistant__user: 'homeassistant'
```

homeassistant__group

System UNIX group used by Home Assistant.

```
homeassistant__group: 'homeassistant'
```

homeassistant__groups

List of additional system groups of the system UNIX account. The `dialout` group grants accesses to devices typically used for home automation which can be found under `/dev/ttyACM*` for example. If you don't use such devices, you can remove the group from the list.

```
homeassistant__groups: [ 'dialout' ]
```

homeassistant__gecos

Contents of the GECOS field set for the Home Assistant account.

```
homeassistant__gecos: 'Home Assistant'
```

homeassistant__shell

The default shell set on the Home Assistant account.

```
homeassistant__shell: '/usr/sbin/nologin'
```

Home Assistant sources and deployment

homeassistant__virtualenv

Should a Python virtualenv be created and used for Home Assistant deployment? Disabled by default so that the Python dependencies packaged by Debian can be used.

```
homeassistant__virtualenv: True
```

homeassistant__release_channel

Which release channel should be installed?

Choices:

- `release` : Latest release.
- `develop` : Latest development version.

```
homeassistant__release_channel: 'release'
```

homeassistant__git_repo

The URI of the Home Assistant git source repository.

```
homeassistant__git_repo: 'https://github.com/home-assistant/home-assistant.git'
```

homeassistant__git_version

The git branch or tag which will be installed. Refer to the releasing documentation for details.

```
homeassistant__git_version: '{{ "master" if (homeassistant__release_channel in [
↪"release"]) else "dev" }}'
```

homeassistant__git_dest

Path where the Home Assistant sources will be checked out (installation path).

```
homeassistant__git_dest: '{{ homeassistant__home_path + "/home-assistant" }}'
```

homeassistant__git_recursive

Should the git repository be cloned recursively?

```
homeassistant__git_recursive: True
```

homeassistant__git_update

Should new revisions be retrieved from the origin repository?

```
homeassistant__git_update: True
```

homeassistant__daemon_path

File path where the Home Assistant console script is located.

```
homeassistant__daemon_path: '{{ (homeassistant__home_path + "/prod_venv/bin/hass")
                                if (homeassistant__virtualenv|bool)
                                else (homeassistant__home_path + "/.local/bin/hass") }
↪}'
```

Configuration for other Ansible roles

homeassistant__nginx__dependent_maps

Configuration of nginx maps, managed by the debops.nginx Ansible role.

```
homeassistant__nginx__dependent_maps:

  - name: 'debops.homeassistant'
    map: '$http_upgrade $homeassistant_connection_upgrade'
    default: 'upgrade'
    mapping: ''""      close;'
```

homeassistant__nginx__dependent_upstreams

Configuration of the Home Assistant nginx upstream, used by the debops.nginx Ansible role.

```
homeassistant__nginx__dependent_upstreams:

  - name: 'homeassistant'
    type: 'default'
```

```
    state: '{{ "present" if (homeassistant__deploy_state == "present") else "absent" }
↪}'
    enabled: True
    server: 'localhost:8123'
```

homeassistant__nginx__dependent_servers

Configuration of the Home Assistant nginx server, used by the debops.nginx Ansible role.

```
homeassistant__nginx__dependent_servers:

  - name: '{{ homeassistant__fqdn }}'
    filename: 'debops.homeassistant'
    by_role: 'debops-contrib.homeassistant'
    state: '{{ "present" if (homeassistant__deploy_state == "present") else "absent" }
↪}'
    type: 'proxy'
    ssl_verify_client: '{{ homeassistant__verify_client_certificate|bool }}'
    options: 'proxy_buffering off;'
    proxy_pass: 'http://homeassistant'
    proxy_options: |
      proxy_redirect http:// https://;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection $homeassistant_connection_upgrade;
```

Copyright

```
debops-contrib.homeassistant - Setup and manage Home Assistant

Copyright (C) 2017 Robin Schneider <ypid@riseup.net>
Copyright (C) 2017 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.homeassistant

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.homeassistant v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]

Ansible role: debops-contrib.kernel_module

Introduction

The `debops-contrib.kernel_module` role allows you to manage Linux kernel modules.

Features

- Module blacklisting
- Module loading (with optional parameters)
- Optionally forces that a module is loaded with certain parameters by unloading it first.
- Either make changes permanent or only to the running system. Default is permanent.

Installation

This role requires at least Ansible `v2.1.3`. To install it, run:

```
ansible-galaxy install debops-contrib.kernel_module
```

Getting started

Example inventory

To manage Linux kernel modules on a given host it should be included in the `debops_service_kernel_module` Ansible inventory group:

```
[debops_service_kernel_module]
hostname
```

Example playbook

Here's an example playbook that uses the `debops-contrib.kernel_module` role:

```
---

- name: Manage Linux kernel modules
  hosts: [ 'debops_service_kernel_module' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops-contrib.kernel_module
      tags: [ 'role::kernel_module' ]
```

The playbooks is shipped with this role under `./docs/playbooks/kernel_module.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::kernel_module` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

debops-contrib.kernel_module default variables

Sections

- *Configuration presets*
- *Lists of module definitions*
- *Default options*
- *Configuration files used for configuring*

Configuration presets

`kernel_module__security_list`

A preset of kernel module configuration intended to increase the security of the system against known attacks.

```
kernel_module__security_list:

  ## Protecting against Firewire DMA:
  ## * https://security.stackexchange.com/a/49158/79474
  ## * https://github.com/lfit/itpol/blob/master/linux-workstation-security.md
↪#blacklisting-modules
  - name: 'firewire-sbp2'
    blacklist: True
  - name: 'firewire-ohci'
    blacklist: True
  - name: 'firewire-core'
    blacklist: True
  - name: 'thunderbolt'
    blacklist: True
```

kernel_module__common_list

A preset of kernel module configuration which is often chosen.

```
kernel_module__common_list:
  - name: 'pcspkr'
    blacklist: True
```

Lists of module definitions

kernel_module__list

"Global" kernel module configuration.

For more details refer to *kernel_module__list* in the *Default variable details* section.

```
kernel_module__list: |
  {{
    kernel_module__security_list|list +
    kernel_module__common_list|list
  }}
```

kernel_module__group_list

"Host group" kernel module configuration.

```
kernel_module__group_list: []
```

kernel_module__host_list

"Host" kernel module configuration.

```
kernel_module__host_list: []
```

kernel_module__combined_list

List of combined kernel module configuration as it is used by the role internally.

```
kernel_module__combined_list: '{{
  (kernel_module__list       | list) +
  (kernel_module__group_list | list) +
  (kernel_module__host_list  | list) }}'
```

Default options

kernel_module__params_force

If `True` , force that the module parameters are applied (via unloading and loading of the module).

```
kernel_module__params_force: False
```

Configuration files used for configuring

kernel_module__blacklist_file

File path where the role maintains blacklisted modules.

```
kernel_module__blacklist_file: '/etc/modprobe.d/blacklist-ansible-kernel_module-role.
↪conf'
```

kernel_module__options_file

File path where the role maintains module options.

```
kernel_module__options_file: '/etc/modprobe.d/options-ansible-kernel_module-role.conf'
```

kernel_module__load_file

File path where the role maintains modules to be loaded on system start.

```
kernel_module__load_file: '/etc/modules-load.d/ansible-kernel_module-role.conf'
```

Default variable details

Some of `debops-contrib.kernel_module` variables have more extensive configuration. Here you can find documentation and examples for them.

kernel_module__list

`kernel_module__list` and similar lists consist of dictionaries with the following supported keys:

`name` Required, string. Name of the kernel module.

`blacklist` If true, blacklist the module. Note that blacklist dominates the loading of modules. Defaults to `False` .

`state` Optional, string. If `present` load the module unless it is blacklisted. Use `absent` to unload the module. Defaults to `present` .

`persistent` Optional, boolean. If `True` , make changes permanent else the changes will not persist a reboot. Defaults to `True` .

`params` Optional, string or list of strings. Kernel module parameters. Example:

```
- name: 'aacraid'
  params: [ 'expose_physicals=1', 'cache=0' ]
```

`params_force` Optional, boolean. If `True` , force that the module parameters are applied (via unloading and loading of the module). Defaults to the value of *`kernel_module__params_force`* which defaults to `False` .

Examples

```
kernel_module__list:

    ## Ensure that ``nf_conntrack_snmp`` is loaded and automatically during each boot.
  - name: 'nf_conntrack_snmp'

    ## Ensure that ``pcspkr`` is blacklisted.
  - name: 'pcspkr'
    blacklist: yes

    ## Ensure that ``aacraid`` is loaded with the kernel module parameter
    ## ``expose_physicals=1``.
  - name: 'aacraid'
    params: 'expose_physicals=1'
    params_force: True
```

Copyright

```
debops-contrib.kernel_module - Manage Linux kernel modules

Copyright (C) 2015-2017 Robin Schneider <ypid@riseup.net>
Copyright (C) 2016-2017 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.kernel_module

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.kernel_module v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]

Changed

- Migrate role to DebOps Contrib as `debops-contrib.kernel_module`. You might need to update your inventory. This oneliner might come in handy to do this.

```
git ls-files -z | xargs --null -I '{}' find '{}' -type f -print0 | xargs --null␣
↪sed --in-place --regexp-extended 's/ypid_service_kernel_modules/debops_service_
↪kernel_module/g;'
```

  [ypid]

- Changed namespace from `kernel_module_` to `kernel_module__`. `kernel_module_[^_]` variables are hereby deprecated and you might need to update your inventory. This oneliner might come in handy to do this.

```
git ls-files -z | xargs --null -I '{}' find '{}' -type f -print0 | xargs --null␣
↪sed --in-place --regexp-extended 's/\<(kernel_module)_([^_])/\1__\2/g;'
```

  [ypid]

- Blacklist `firewire-core` and `thunderbolt` by default using the *`kernel_module__security_list`* variable. [ypid]
- Blacklist also `firewire-ohci` to allow successful unloading of `firewire-core` if the modules are already loaded. [ypid]

Ansible role: debops-contrib.neurodebian

Introduction

The `debops-contrib.neurodebian` role allows you to configure the NeuroDebian repository and install packages from it.

NeuroDebian provides a large collection of popular neuroscience research software for the Debian operating system as well as Ubuntu and other derivatives.

Installation

This role requires at least Ansible `v2.1.5`. To install it, run:

```
ansible-galaxy install debops-contrib.neurodebian
```

Getting started

- *Example inventory*
- *Example playbook*
- *Ansible tags*

Example inventory

To install packages from NeuroDebian on a given host or set of hosts, they need to be added to the `[debops_service_neurodebian]` Ansible group in the inventory:

```
[debops_service_neurodebian]
hostname
```

Example playbook

If you are using this role without DebOps, here's an example Ansible playbook that uses the `debops-contrib.neurodebian` role:

```
---

- name: Install packages from the NeuroDebian repository
  hosts: [ 'debops_service_neurodebian' ]
  become: True

  environment: '{{ inventory__environment | d({})
                 | combine(inventory__group_environment | d({}))
                 | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops.apt_preferences
      tags: [ 'role::apt_preferences' ]
      apt_preferences__dependent_list:
        - '{{ neurodebian__apt_preferences__dependent_list }}'

    - role: debops-contrib.neurodebian
      tags: [ 'role::neurodebian' ]
```

The playbook is shipped with this role under `./docs/playbooks/neurodebian.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::neurodebian` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::neurodebian:pkgs` Tasks related to system package management like installing or removing packages.

debops-contrib.neurodebian default variables

Sections

NeuroDebian packages and installation

neurodebian__packages

List of global packages for NeuroDebian. This variable is intended to be used in Ansible's global inventory.

```
neurodebian__packages: []
```

neurodebian__group_packages

List of group packages for NeuroDebian. This variable is intended to be used in a host inventory group of Ansible (only one host group is supported).

```
neurodebian__group_packages: []
```

neurodebian__host_packages

List of host packages for NeuroDebian. This variable is intended to be used in the inventory of hosts.

```
neurodebian__host_packages: []
```

neurodebian__dependent_packages

List of APT packages to install for other Ansible roles, for usage as a dependent role.

```
neurodebian__dependent_packages: []
```

neurodebian__deploy_state

What is the desired state which this role should achieve? Possible options:

present Default. Ensure that repositories and packages provided by NeuroDebian are installed and configured as requested.

absent Ensure that repositories and packages provided by NeuroDebian are absent.

```
neurodebian__deploy_state: 'present'
```

APT repository configuration

Refer to neuro.debian.net for details.

neurodebian__apt_components

The NeuroDebian repository component/flavor to enable. Supported choices: `main`, `contrib`, `non-free`.

```
neurodebian__apt_components:
  - 'main'
```

neurodebian__apt_source_types

List of source types to configure for the NeuroDebian repository. Supported choices: `deb` , `deb-src` .

```
neurodebian__apt_source_types: [ 'deb' ]
```

neurodebian__apt_mirror_uri

The NeuroDebian APT repository mirror URI to use. Refer to neuro.debian.net for the full list.

```
neurodebian__apt_mirror_uri: 'http://neurodebian.g-node.org'
```

neurodebian__apt_key_fingerprint

The OpenPGP key fingerprint for the key by which the NeuroDebian APT repository is signed.

```
neurodebian__apt_key_fingerprint: 'DD95CC430502E37EF840ACEEA5D32F012649A5A9'
```

APT pinning configuration

neurodebian__apt_preferences__dependent_list

APT pinning for packages from the NeuroDebian repository. By default (without this pinning), both the official Debian repositories and NeuroDebian have the same preference which would lead to APT picking the package with the highest version regardless from which repository it comes from. As NeuroDebian provides many additional packages along with more recent versions of packages already available in official Debian releases, APT pinning is used to ensure that package versions available in official Debian releases are preferred even if NeuroDebian provides newer versions. The job of setting up the APT pinning is offloaded to the debops.apt_preferences role which is instructed using this variable.

```
neurodebian__apt_preferences__dependent_list:

  - package: '*'
    reason: |-
      Pin NeuroDebian with priority 80 which is lower then the official Debian␣
↪backports (100).
      This also works when `apt_preferences__preset_list` is set which increases
      Debian backports to 400 and decreases Debian testing to 50.
    by_role: 'debops.neurodebian'
    pin: 'release o=NeuroDebian'
    priority: '80'
    state: '{{ "present" if (neurodebian__deploy_state == "present") else "absent" }}'
```

Copyright

```
debops-contrib.neurodebian - Install packages from the NeuroDebian repository

Copyright (C) 2017 Robin Schneider <ypid@riseup.net>
Copyright (C) 2017 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
```

```
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.neurodebian

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.neurodebian master - unreleased

debops-contrib.neurodebian v0.1.0 - 2017-05-02

Added

- Initial coding and design. [ypid]

Ansible role: debops-contrib.roundcube

Introduction

This Ansible role allows you to install and manage Roundcube, a IMAP Web client written in PHP.

Installation

This role requires at least Ansible `v2.3.0`. To install it, run:

```
ansible-galaxy install debops-contrib.roundcube
```

Getting started

Default setup

If you don't specify any configuration values, the role will setup a Nginx HTTP server running a default installation of the latest Roundcube stable release which is then accessible via `https://roundcube.<your-domain>`. SQLite is used as database backend for storing the user settings.

Example inventory

Roundcube can be installed on a given host by adding it to the `[debops_service_roundcube]` Ansible inventory group:

```
[debops_service_roundcube]
hostname
```

Example playbook

The following playbook can be used with DebOps. If you are using these role without DebOps you might need to adapt them to make them work in your setup.

```
---

- name: Manage Roundcube Web mail
  hosts: [ 'debops_service_roundcube' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops.php/env
      tags: [ 'role::php', 'role::php:env' ]

    - role: debops.apt_preferences
      tags: [ 'role::apt_preferences', 'role::nginx', 'role::php' ]
      apt_preferences__dependent_list:
        - '{{ nginx__apt_preferences__dependent_list }}'
        - '{{ php__apt_preferences__dependent_list }}'

    - role: debops.logrotate
      tags: [ 'role::logrotate' ]
      logrotate__dependent_config:
        - '{{ php__logrotate__dependent_config }}'

    - role: debops.ferm
      tags: [ 'role::ferm', 'role::nginx' ]
      ferm__dependent_rules:
        - '{{ nginx__ferm__dependent_rules }}'

    - role: debops.php
      tags: [ 'role::php' ]
      php__dependent_packages:
        - '{{ roundcube__php__dependent_packages }}'
      php__dependent_pools:
```

```
      - '{{ roundcube__php__dependent_pools }}'

  - role: debops.nginx
    tags: [ 'role::nginx' ]
    nginx__dependent_servers:
      - '{{ roundcube__nginx__dependent_servers }}'
    nginx__dependent_upstreams:
      - '{{ roundcube__nginx__dependent_upstreams }}'

  - role: debops.mariadb
    tags: [ 'role::mariadb' ]
    mariadb__dependent_users:
      - database: '{{ roundcube__database_map[roundcube__database].dbname }}'
        user: '{{ roundcube__database_map[roundcube__database].dbuser }}'
        password: '{{ roundcube__database_map[roundcube__database].dbpass }}'
        owner: '{{ roundcube__user }}'
        group: '{{ roundcube__group }}'
        home: '{{ roundcube__home }}'
        system: True
        priv_aux: False
    mariadb__server: '{{ roundcube__database_map[roundcube__database].dbhost }}'
    when: roundcube__database_map[roundcube__database].dbtype == 'mysql'

  - role: debops-contrib.roundcube
    tags: [ 'role::roundcube' ]
```

This playbook is also shipped with the role under `docs/playbooks/`.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::roundcube` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::roundcube:pkg` Run tasks related to system package installation.

`role::roundcube:deployment` Run tasks related to the application deployment and update.

`role::roundcube:config` Run tasks related to the Roundcube application configuration.

`role::roundcube:database` Run tasks related to setup or update the database user and schema.

debops-contrib.roundcube default variables

Sections

Packages and installation

roundcube__required_php_packages

List of PHP packages required by Roundcube. Refer to the official Roundcube documentation for details.

```
roundcube__required_php_packages:
  - 'iconv'
  - 'openssl'
  - 'session'
  - 'sockets'
  - 'xml'
  # Included in the xml package
  #- 'dom'
  - 'mbstring'
  - 'json'
```

roundcube__optional_php_packages

List of recommended/optional PHP packages for Roundcube. Refer to the official Roundcube documentation for details.

```
roundcube__optional_php_packages:
  - 'fileinfo'
  - 'pspell'
  - 'zip'
```

roundcube__custom_php_packages

List of user defined PHP packages for Roundcube.

```
roundcube__custom_php_packages: []
```

roundcube__base_php_packages

List of base PHP packages required by Roundcube.

```
roundcube__base_php_packages:
  - '{{ roundcube__required_php_packages }}'
  - '{{ roundcube__apt_php_packages }}'
  - '{{ roundcube__optional_php_packages }}'
  - '{{ [ "mysql" ] if (roundcube__database_map[roundcube__database].dbtype == "mysql
↪") else [] }}'
  - '{{ [ "sqlite3" ] if (roundcube__database_map[roundcube__database].dbtype ==
↪"sqlite") else [] }}'
```

roundcube__apt_php_packages

PHP packages which are installed via APT repository if they are available in a sufficiently new version in the current distribution. The required minimal versions are taken from the file:*composer.json.dist* of the Roundcube 1.3.0 release. If you install an older version of Roundcube you may want to adjust this list.

```
roundcube__apt_php_packages: '{{ [ "mail-mime", "net-smtp", "pear" ]
                                 if ansible_distribution_release in [ "stretch",
↪"buster", "sid", "xenial", "yakkety", "zesty", "artful" ]
                                 else [] }}'
```

roundcube__packages

List of additional APT packages (e.g. language dictionaries) that should be installed with Roundcube.

```
roundcube__packages: []
```

roundcube__base_packages

APT packages required for the Roundcube installation.

```
roundcube__base_packages: [ 'curl', 'file', 'unzip' ]
```

roundcube__composer_packages

APT packages required to install PHP composer.

```
roundcube__composer_packages: [ 'composer ']
```

roundcube__composer_phar

If this is set to `True` the `composer.phar` script will be downloaded from the *roundcube__composer_phar_url* and used to install the missing PHP packages. If this is set to `False` the system-wide **composer** is used. WARNING: Setting this variable to `True` has some security implications as the download is not cryptographically verified. This is only meant to be a work-around for old distribution releases not supporting the downstream packaged **composer**.

```
roundcube__composer_phar: '{{ True
                              if ansible_distribution_release in [ "jessie", "trusty"␣
↪]
                              else False }}'
```

roundcube__composer_phar_url

URL to the file:*composer.phar* script which will be used to install PHP packages not available in the APT repository on distribution releases which don't package PHP composer. If this is set to `False` , **composer** will be installed via APT package manager.

```
roundcube__composer_phar_url: 'https://getcomposer.org/composer.phar'
```

Roundcube user account

roundcube__user

Roundcube system user account.

```
roundcube__user: 'roundcube'
```

roundcube__group

Roundcube system user group.

```
roundcube__group: 'roundcube'
```

roundcube__home

Path to the home directory of the Roundcube system account.

```
roundcube__home: '{{ (ansible_local.root.home
                      if (ansible_local|d() and ansible_local.root|d() and
                          ansible_local.root.home|d())
                      else "/var/local") + "/" + roundcube__user }}'
```

roundcube__comment

The GECOS string set for the Roundcube account.

```
roundcube__comment: 'Roundcube Webmail'
```

roundcube__shell

The default shell of the Roundcube account.

```
roundcube__shell: '/usr/sbin/nologin'
```

Roundcube source and deployment

roundcube__git_repo

Roundcube source repository.

```
roundcube__git_repo: 'https://github.com/roundcube/roundcubemail.git'
```

roundcube__git_dest

Roundcube source directory on the host.

```
roundcube__git_dest: '{{ roundcube__src + "/" + roundcube__git_repo.split("://")[1] }}
↪'
```

roundcube__git_version

Roundcube release tag to deploy.

```
roundcube__git_version: '1.3.0'
```

roundcube__git_checkout

Default path where Roundcube source files will be deployed.

```
roundcube__git_checkout: '{{ roundcube__www + "/sites/" +
                             (roundcube__domain if roundcube__domain is string else␣
↪roundcube__domain[0]) +
                             "/public" }}'
```

roundcube__src

Base path for git bare repository with Roundcube source.

```
roundcube__src: '{{ (ansible_local.root.src
                     if (ansible_local|d() and ansible_local.root|d() and
                         ansible_local.root.src|d())
                     else "/usr/local/src") + "/" + roundcube__user }}'
```

roundcube__www

Base web root directory for Roundcube website.

```
roundcube__www: '{{ ansible_local.nginx.www
                    if (ansible_local|d() and ansible_local.nginx|d())
                    else "/srv/www" }}'
```

roundcube__webserver_user

Roundcube webserver user (needs read-only access to the website code).

```
roundcube__webserver_user: '{{ ansible_local.nginx.user
                               if (ansible_local|d() and
                                   ansible_local.nginx|d() and
                                   ansible_local.nginx.user|d())
                               else "www-data" }}'
```

Database configuration

roundcube__database

Database definition to use from the *roundcube__database_map*.

```
roundcube__database: 'sqlite-default'
```

roundcube__database_user

Database user account to use for Roundcube.

```
roundcube__database_user: 'roundcube'
```

roundcube__database_password_path

Path to the database password file.

```
roundcube__database_password_path: '{{ secret + "/credentials/" + ansible_fqdn
                                       + "/roundcube/" + roundcube__database
                                       + "/" + roundcube__database_user + "/password"␣
↪}}'
```

roundcube__database_password

Database password for the account given in *roundcube__database_user*.

```
roundcube__database_password: '{{ lookup("password", roundcube__database_password_
↪path + " length=30") }}'
```

roundcube__database_name

Name of the database to use for Roundcube.

```
roundcube__database_name: 'roundcubemail'
```

roundcube__database_map

Database connection definitions. Selet the database connection to use in *roundcube__database*.

```
roundcube__database_map:

  sqlite-default:
    dbtype: 'sqlite'
    dbname: 'db/roundcube.db'

  mysql-default:
    dbtype: 'mysql'
    dbname: '{{ roundcube__database_name }}'
    dbuser: '{{ roundcube__database_user }}'
    dbpass: '{{ roundcube__database_password }}'
    dbhost: 'localhost'
    dbtableprefix: ''
```

roundcube__database_schema

Initial Roundcube database schema loaded by Ansible.

```
roundcube__database_schema: '{{ roundcube__git_checkout + "/SQL/mysql.initial.sql"
    if (roundcube__database_map[roundcube__database].dbtype == "mysql") else ""}}'
```

Roundcube application options

roundcube__domain

String or List of domains which will be used to access the roundcube instance.

```
roundcube__domain: [ 'roundcube.{{ ansible_domain }}' ]
```

roundcube__default_host

Mail host chosen to perform the log-in.

```
roundcube__default_host: 'localhost'
```

roundcube__smtp_server

SMTP server host (for sending mails).

```
roundcube__smtp_server: ''
```

roundcube__smtp_port

SMTP port.

```
roundcube__smtp_port: '25'
```

roundcube__smtp_user

SMTP username (if required) if you use %u as the username Roundcube will use the current username for login.

```
roundcube__smtp_user: ''
```

roundcube__smtp_pass

SMTP password (if required) if you use %p as the password Roundcube will use the current user's password for login.

```
roundcube__smtp_pass: ''
```

roundcube__des_key

Encryption key for the users imap password which is stored in the session record (and the client cookie if remember password is enabled).

```
roundcube__des_key: '{{ lookup("password", secret + "/credentials/" + ansible_fqdn +
↪"/roundcube/des_key chars=hexdigits length=24") }}'
```

roundcube__local_config_map

Custom configuration values for the Roundcube config.inc.php.

```
roundcube__local_config_map: {}
```

roundcube__group_local_config_map

Custom configuration values which can be defined on a group level and eventually are merged with *roundcube__local_config_map*.

```
roundcube__group_local_config_map: {}
```

roundcube__host_local_config_map

Custom configuration values which can be defined ona host level and eventually are merged with *roundcube__local_config_map*.

```
roundcube__host_local_config_map: {}
```

roundcube__default_plugins

List of plugins shipped and enabled by default with Roundcube.

```
roundcube__default_plugins: [ 'archive', 'filesystem_attachments', 'jqueryui',
↪'zipdownload' ]
```

roundcube__plugins

Additional Roundcube plugins to enable. Check the `plugins/` folder for the plugins shipped by default.

```
roundcube__plugins: []
```

Other variables

roundcube__max_file_size

Maximum upload size, in MB.

```
roundcube__max_file_size: '30'
```

Role-dependent configuration

roundcube__nginx__dependent_servers

nginx server configuration managed by the debops.nginx role.

```
roundcube__nginx__dependent_servers:

  - name: '{{ roundcube__domain }}'
    filename: 'debops-contrib.roundcube'
    by_role: 'debops-contrib.roundcube'
    type: 'php'
    default: False
    root: '{{ roundcube__git_checkout }}'
    access_policy: '{{ roundcube__nginx_access_policy }}'
    index: 'index.php'

    options: |
      autoindex off;
      client_max_body_size {{ roundcube__max_file_size }}M;
      client_body_buffer_size 128k;

    location_list:
      - pattern: '/'
        options: |
          try_files $uri $uri/ @roundcube;

      - pattern: '@roundcube'
        options: |
          rewrite ^/favicon\.ico$ skins/larry/images/favicon.ico last;

      - pattern: '~ ^/?(installer|[A-Z0-9]+$)'
        options: |
          deny all;

      - pattern: '~ ^/?(\.git|\.tx|SQL|bin|config|logs|temp|tests|program\/
↪(include|lib|localization|steps))'
        options: |
          deny all;

      - pattern: '~ /(README\.md|composer\.json-dist|composer\.json|package\.
↪xml|Dockerfile)$'
        options: |
          deny all;

    php_options: |
      fastcgi_intercept_errors        on;
      fastcgi_ignore_client_abort     off;
      fastcgi_connect_timeout         60;
      fastcgi_send_timeout            180;
      fastcgi_read_timeout            180;
      fastcgi_buffer_size             128k;
      fastcgi_buffers               4 256k;
      fastcgi_busy_buffers_size       256k;
      fastcgi_temp_file_write_size    256k;

    php_upstream: 'php_roundcube'
```

roundcube__nginx_access_policy

Name of the "nginx access policy" for Roundcube webpage. See debops.nginx Ansible role for more details.

```
roundcube__nginx_access_policy: ''
```

roundcube__nginx__dependent_upstreams

PHP upstream server configuration managed by the debops.nginx role.

```
roundcube__nginx__dependent_upstreams:

  - name: 'php_roundcube'
    by_role: 'debops-contrib.roundcube'
    enabled: True
    type: 'php'
    php_pool: 'roundcube'
```

roundcube__php__dependent_packages

List of PHP packages to install using the debops.php role.

```
roundcube__php__dependent_packages:
  - '{{ roundcube__base_php_packages }}'
  - '{{ roundcube__optional_php_packages }}'
  - '{{ roundcube__custom_php_packages }}'
```

roundcube__php__dependent_pools

PHP pools managed by the debops.php role.

```
roundcube__php__dependent_pools:
  name: 'roundcube'
  by_role: 'debops-contrib.roundcube'
  user: '{{ roundcube__user }}'
  group: '{{ roundcube__group }}'

  php_values:
    ## https://secure.php.net/manual/en/info.configuration.php#ini.upload-max-filesize
    upload_max_filesize:        '{{ roundcube__max_file_size }}M'

    ## https://secure.php.net/manual/en/ini.core.php#ini.post-max-size
    post_max_size:              '{{ roundcube__max_file_size }}M'

    ## https://github.com/roundcube/roundcubemail/wiki/Install-Requirements
    file_uploads:               'on'
    mbstring.func_overload:     'off'
    memory_limit:               '64M'
    magic_quotes_runtime:       'off'
    magic_quotes_sybase:        'off'
    session.auto_start:         'off'
    suhosin.session.encrypt:    'off'
```

Guides and examples

Enable spell checking with aspell

To enable local spell checking of your email content, you have to install `php5-enchant` and `aspell` together with the according language dictionaries. For example for english and french spell checking, you would add the following packages to your Roundcube role configuration:

```
roundcube__packages: [ 'php5-enchant', 'aspell', 'aspell-en', 'aspell-fr' ]
```

Additionally you have to tell Roundcube that you want to use the local spell checking library:

```
roundcube__local_config_map:
  spellcheck_engine: 'enchant'
  spellcheck_languages: "array('en', 'fr')"
```

Of course, many more languages are supported. You can find more information about the required packages and configuration in the Roundcube Aspell-Howto.

Copyright

```
debops-contrib.roundcube - Manage Roundcube Web mail with Ansible

Copyright (C) 2016-2017 Reto Gantenbein <reto.gantenbein@linuxmonk.ch>
Copyright (C) 2016-2017 DebOps https://debops.org/

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see https://www.gnu.org/licenses/
```

Changelog

debops-contrib.roundcube

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ganto.

Refer to the *Upgrade notes* when you intend to upgrade to a new release of this role.

debops-contrib.roundcube master - unreleased

debops-contrib.roundcube v0.2.0 - 2017-08-28

Added

- Added new soft dependency on debops.ferm to the example playbook. [ganto]
- Added new soft dependency on debops.apt_preferences to the example playbook to satisfy possible package pinning requirements of the debops.nginx and debops.php roles. [ganto]
- Added new soft dependency on debops.logrotate to the example playbook to handle logfile rotation of PHP-FPM. [ganto]
- New configuration variables `roundcube__shell` and `roundcube__comment` to customize the Roundcube system account. [ganto]

- New configuration variables *roundcube__database_password_path* and *roundcube__database_name* for easier customization of the database setup. [ganto]
- Install PHP packages which cannot be satisfied by the APT package manager via PHP's own **composer** dependency manager. [ganto]
- Run post-install script provided by upstream which downloads the required Javascript libraries served to the Web browsers. [ganto]

Changed

- Set default Roundcube version to 1.3.0. [ganto]
- Adjusted the debops.nginx configuration to make use of the role's dependent variables which required minor format changes and variable name adjustments to correspond to the DebOps naming conventions: `roundcube__nginx_server` → *roundcube__nginx__dependent_servers* `roundcube__nginx_upstream_php5` → *roundcube__nginx__dependent_upstreams* [ganto]
- Make use of the debops.mariadb dependent variables in the example playbook. [ganto]
- Updated PHP role dependency from `debops.php5` to the more capable debops.php. This changed the format and name of the following variables: `roundcube__php5_packages` → *roundcube__php__dependent_packages* `roundcube__php5_pool` → *roundcube__php__dependent_pools* [ganto]
- Renamed `roundcube__extra_packages` to *roundcube__packages* to be consistent with other DebOps roles. [ganto]
- Changed default configuration of *roundcube__www* from `/srv/www/{ roundcube__user }` to `/srv/www` to be more consistent with other system-wide Web applications. [ganto]

Fixed

- Fixed definition of *roundcube__home* and *roundcube__src* in cases where the local facts defined by debops.core are not available. [ganto]

Removed

- Remove support for Debian (oldoldstable) wheezy. [ganto]

debops-contrib.roundcube v0.1.3 - 2017-07-26

Changed

- Set default version to 1.1.9. [ganto]

Fixed

- Fix documentation build error due to deleted link definition to deprecated `debops.php5` role repository. [ganto]

- Probe if *roundcube__domain* is a string and construct *roundcube__git_checkout* accordingly. [cultcom]

debops-contrib.roundcube v0.1.2 - 2017-03-09

Changed

- Set default version to 1.1.7. [ganto]
- Moved all variable definitions to `defaults/main.yml` for better configurability. Restructured defaults configuration file. [ganto]

Fixed

- Properly pass `password` parameter to debops.mariadb role dependency. [cultcom]
- Fix `login_host` definition in database schema import. [cultcom]
- Fix syntax error in *roundcube__database_schema* variable definition. [cultcom]
- Fix MySQL database schema setup when using remote database by adjusting indentation of example playbook. [ganto]

debops-contrib.roundcube v0.1.1 - 2016-08-03

Changed

- Introduced playbook-based role dependencies and removed hard-dependencies on optional roles. For this reason the role variable `roundcube__dependencies` was removed too. If no or only individual dependencies are required simply adjust the playbook accordingly. [ganto]
- Converted documentation/Changelog to a new format. [ganto]

debops-contrib.roundcube v0.1.0 - 2016-06-14

Added

- Initial release of Roundcube 1.1.5 with SQLite and MySQL support. [ganto]

Upgrade notes

The upgrade notes only describe necessary changes that you might need to make to your setup in order to use a new role release. Refer to the *Changelog* for more details about what has changed.

From v0.1.3 to v0.2.0

Due to changes in the role dependencies and some adjustments in the role's default values, your setup is likely to break if you simply execute the updated role. To avoid this, take care of the following issues:

- If you are using a custom playbook, make sure to review the changes in the *Example playbook*.
- The following variables were replaced and therefore are not defined anymore in the default variables:

  - roundcube__nginx_server
  - roundcube__nginx_upstream_php5
  - roundcube__php5_packages
  - roundcube__php5_pool
  - roundcube__extra_packages

  In case your playbook is referencing one of them, make sure they are properly defined in your inventory or update your playbook. If you are using the example playbook but customized one of those variables in your Ansible inventory update the definition accordingly.

- The default installation path defined in *roundcube__www* changed. If you didn't customize its value the Roundcube installation will be under a new file system path after the installation.

Upgrade procedure

The following procedure is valid if you are using the role dependencies as defined in the example playbook.

1. Make sure you have the latest version of the DebOps roles.

```
$ debops-update
```

2. Make sure you have the lastest version of the debops-contrib.roundcube role. In your DebOps project directory run:

```
$ ansible-galaxy install --force --no-deps --roles-path=ansible/roles debops-
↪contrib.roundcube
```

2. Review the *Changelog* and make sure your Ansible inventory is adjusted to the variable changes (if necessary).
3. Remove the nginx virtual host and PHP definitions created by the debops.nginx role from the Roundcube server:

```
# rm /etc/nginx/{sites-available,sites-enabled}/roundcube.example.com.conf
# rm /etc/nginx/conf.d/upstream_php5_roundcube.conf
```

4. Run the role (e.g. via example playbook):

```
$ debops ansible/roles/debops-contrib.roundcube/docs/playbooks/roundcube.yml
```

5. In case you are using the default configuration copy the Roundcube SQLite database containing the user settings to the new installation path.

```
$ cp /srv/www/roundcube/sites/roundcube.example.com/public/db/roundcube.db \
  /srv/www/sites/roundcube.example.com/public/db
```

6. In case Roundcube was installed into a new directory but you didn't use the default *roundcube__www* configuration before the update or you experience SQL schema issues, you need to manually run the upstream post update script on the Roundcube server.

```
# su roundcube -s /bin/bash \
  -c "php /srv/www/sites/roundcube.example.com/public/bin/updatedb.sh \
  --package=roundcube --dir=/srv/www/sites/roundcube.example.com/public/SQL"
```

7. If you manually installed some additional plugins you might need to reinstall or update them for the new Roundcube version.

Ansible role: debops-contrib.snapshot_snapper

Introduction

Snapper can manage snapshots for the following filesystems and volume managers:

- btrfs
- ext4
- lvm2

This role allows to setup and configure snapper with Ansible.

Installation

This role requires at least Ansible `v2.1.3`. To install it, run:

```
ansible-galaxy install debops-contrib.snapshot_snapper
```

Getting started

Example inventory

To configure volume snapshots on host given in `debops_service_snapshot_snapper` Ansible inventory group:

```
[debops_service_snapshot_snapper]
hostname
```

Example playbook

Here's an example playbook that uses the `debops-contrib.snapshot_snapper` role:

```
---

- name: Configure volume snapshots with snapper
  hosts: [ 'debops_service_snapshot_snapper' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:
```

```
- role: debops-contrib.snapshot_snapper
  tags: [ 'role::snapshot_snapper' ]
```

The playbooks is shipped with this role under `./docs/playbooks/snapshot_snapper.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::snapshot_snapper` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::snapshot_snapper:reinit` Execute tasks related to automatic reinitialization of volume snapshots.

debops-contrib.snapshot_snapper default variables

Sections

Required packages

`snapshot_snapper__base_packages`

List of base packages to install.

```
snapshot_snapper__base_packages:
  - 'snapper'
```

`snapshot_snapper__packages`

List of optional packages to install. If the `mlocate` package is listed, the *`snapshot_snapper__directory`* will be excluded from mlocate. (Currently `mlocate` is required to be installed for this role).

```
snapshot_snapper__packages:
  - 'mlocate'
```

Snapper templates

`snapshot_snapper__templates`

Sets the global default settings for snapper. If empty, the default of snapper will be left unchanged.

Example:

```
snapshot_snapper__templates:
  default:
    TIMELINE_LIMIT_HOURLY: 12
    TIMELINE_LIMIT_DAILY: 10
    TIMELINE_LIMIT_MONTHLY: 6
    TIMELINE_LIMIT_YEARLY: 0
```

```
snapshot_snapper__templates: {}
```

snapshot_snapper__host_group_templates

Sets the host group default settings for snapper.

```
snapshot_snapper__host_group_templates: {}
```

snapshot_snapper__host_templates

Sets the host default settings for snapper.

```
snapshot_snapper__host_templates: {}
```

Volume configuration

snapshot_snapper__volumes

"Global" list of volumes to snapshot.

path String, required. Path of the volume.

name String, required. Name of the volume. Only used by snapper.

template String, optional. Defaults to `default` . Name of the template to base the configuration on.

config Dictionary of strings, optional. Allows you to overwrite a setting from the template.

state String, optional. Defaults to `present` . Choices `present` , `absent` .

Example:

```
snapshot_snapper__volumes:
  - path: '/'
    name: 'root'
    template: 'common'
    config:
      TIMELINE_LIMIT_MONTHLY: 9
      TIMELINE_LIMIT_YEARLY: 2
```

```
snapshot_snapper__volumes: []
```

snapshot_snapper__host_group_volumes

"Host group" list of volumes to snapshot.

```
snapshot_snapper__host_group_volumes: []
```

snapshot_snapper__host_volumes

"Host" list of volumes to snapshot.

```
snapshot_snapper__host_volumes: []
```

snapshot_snapper__auto_reinit

Automatically reinitialize the snapshots for a volume if the directory containing the actual snapshots has vanished. This can be useful if the volume has been reformatted but the old snapper configuration is still in place. All what snapper does in such case is to return "IO Error." when trying to work with this volume configuration.

When this option is set to `True` , the role will do some manually intervention to automatically fix this if necessary.

```
snapshot_snapper__auto_reinit: True
```

Role internal configuration

snapshot_snapper__directory

Name of the directory in the root of the volume containing the snapshots and metadata.

```
snapshot_snapper__directory: '.snapshots'
```

snapshot_snapper__divert_files

List of files which the role will divert.

```
snapshot_snapper__divert_files:
  - '/etc/updatedb.conf'
  - '/etc/snapper/config-templates/default'
```

Copyright

```
debops-contrib.snapshot_snapper - Configure snapshots with snapper

Copyright (C) 2015-2017 Robin Schneider <ypid@riseup.net>
Copyright (C) 2016-2017 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.snapshot_snapper

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.snapshot_snapper v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]
- Wrote initial documentation. [ypid]
- Implemented automatic reinitialization of volume snapshots after a volume has been reformatted. [ypid]

Changed

- Moved to DebOps Contrib (the role is still available under `ypid.snapshot_snapper` until it has been fully renamed). [ypid]
- Changed namespace from `snapshot_snapper_` to `snapshot_snapper__`. `snapshot_snapper_[^_]` variables are hereby deprecated and you might need to update your inventory. This oneliner might come in handy to do this.

```
git ls-files -z | find -type f -print0 | xargs --null sed --in-place --regexp-
↪extended 's/(snapshot_snapper)_([^_])/\1__\2/g'
```

  [ypid]
- Use the loop control feature of Ansible 2.1 and thus require Ansible 2.1. [ypid]
- Include the `mlocate` package in the default package list as the role requires it currently. More rework is needed. [ypid]

Fixed

- Fixed recognition of empty `SNAPPER_CONFIGS` set in `/etc/default/snapper` and don't write a second `SNAPPER_CONFIGS` variable in this case. Previous to this fix, snapshots where not automatically created because a second `SNAPPER_CONFIGS` (empty) set was added to the file. [ypid]

Ansible role: debops-contrib.volkszaehler

Introduction

The `debops-contrib.volkszaehler` role allows to setup your own volkszaehler.org instance. volkszaehler.org is a free smart meter implementation with focus on data privacy.

A volkszaehler instance consists of a middleware written in PHP which is backed by a SQL database. The middleware provides an API for frontends to query data and for controllers to insert data.

This role installs the default middleware together with the default/bundled HTML/JavaScript frontend.

Installation

This role requires at least Ansible `v2.1.5`. To install it, run:

```
ansible-galaxy install debops-contrib.volkszaehler
```

Getting started

Database support

It is recommended that you install a database server. You can install one on the same host as volkszaehler or choose a different host:

```
[debops_service_mariadb_server]
hostname
```

In case you chose a different host, you will need to specify which of your database servers the volkszaehler instance should use by specifying the database server host as *`volkszaehler__database_server`*.

Webserver support

The following two webservers are supported by the role:

- Apache using debops.apache.
- Nginx using debops.nginx.

The role maintainer has chosen Nginx as webserver for his deployments. He added Apache support because he is very familiar with debops.apache (author). Note that integration testing is done with debops.nginx only, currently.

In case you chose Apache, you don't need PHP FPM which debops.php might install by default. To ensure that FPM is not going to be installed, add the following to your inventory:

```
php__server_api_packages:
  - 'cli'
```

Example inventory

To manage volkszaehler on a given host or set of hosts, they need to be added to the `[debops_service_volkszaehler_${webserver}]` Ansible group in the inventory:

```
[debops_service_volkszaehler_apache]
hostname

[debops_service_mariadb_server]
hostname

[debops_service_volkszaehler_nginx]
hostname2
```

Example playbook

Ansible playbook that uses the `debops-contrib.volkszaehler` role together with debops.apache:

```
---

- name: Setup and manage volkszaehler with Apache as webserver
  hosts: [ 'debops_service_volkszaehler_apache' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops-contrib.volkszaehler/env
      tags: [ 'role::volkszaehler', 'role::volkszaehler:env', 'role::mariadb' ]

    - role: debops.apache/env
      tags: [ 'role::apache', 'role::apache:env' ]

    - role: debops.php/env
      tags: [ 'role::php', 'role::php:env','role::apt_preferences', 'role::logrotate'␣
↪]

    - role: debops.apt_preferences
      tags: [ 'role::apt_preferences' ]
      apt_preferences__dependent_list:
        - '{{ php__apt_preferences__dependent_list }}'

    - role: debops.ferm
      tags: [ 'role::ferm' ]
      ferm__dependent_rules:
        - '{{ apache__ferm__dependent_rules }}'

    - role: debops.mariadb
      tags: [ 'role::mariadb' ]
      mariadb__dependent_databases: '{{ volkszaehler__mariadb__dependent_databases }}'
      mariadb__dependent_users: '{{ volkszaehler__mariadb__dependent_users }}'
      when: (volkszaehler__database == 'mariadb')

    - role: debops.php
      tags: [ 'role::php' ]
      php__dependent_packages:
        - '{{ volkszaehler__php__dependent_packages }}'
      php__dependent_pools:
        - '{{ volkszaehler__php__dependent_pools }}'
```

```
    - role: debops.logrotate
      tags: [ 'role::logrotate' ]
      logrotate__dependent_config:
        - '{{ php__logrotate__dependent_config }}'

    - role: geerlingguy.composer
      tags: [ 'role::composer' ]

    - role: debops.apache
      tags: [ 'role::apache' ]
      apache__dependent_vhosts:
        - '{{ volkszaehler__apache__dependent_vhosts }}'

    - role: debops-contrib.volkszaehler
      tags: [ 'role::volkszaehler' ]
```

Ansible playbook that uses the `debops-contrib.volkszaehler` role together with debops.nginx:

```
---

- name: Setup and manage volkszaehler with Nginx as webserver
  hosts: [ 'debops_service_volkszaehler_nginx' ]
  become: True

  environment: '{{ inventory__environment | d({})
                   | combine(inventory__group_environment | d({}))
                   | combine(inventory__host_environment  | d({})) }}'

  roles:

    - role: debops-contrib.volkszaehler/env
      tags: [ 'role::volkszaehler', 'role::volkszaehler:env', 'role::mariadb' ]

    - role: debops.php/env
      tags: [ 'role::php', 'role::php:env','role::apt_preferences', 'role::logrotate'␣
↪]

    - role: debops.apt_preferences
      tags: [ 'role::apt_preferences' ]
      apt_preferences__dependent_list:
        - '{{ nginx__apt_preferences__dependent_list }}'
        - '{{ php__apt_preferences__dependent_list }}'

    - role: debops.ferm
      tags: [ 'role::ferm' ]
      ferm__dependent_rules:
        - '{{ nginx__ferm__dependent_rules }}'

    - role: debops.mariadb
      tags: [ 'role::mariadb' ]
      mariadb__dependent_databases: '{{ volkszaehler__mariadb__dependent_databases }}'
      mariadb__dependent_users: '{{ volkszaehler__mariadb__dependent_users }}'
      when: (volkszaehler__database == 'mariadb')

    - role: debops.php
      tags: [ 'role::php' ]
      php__dependent_packages:
```

```
      - '{{ volkszaehler__php__dependent_packages }}'
    php__dependent_pools:
      - '{{ volkszaehler__php__dependent_pools }}'

  - role: debops.logrotate
    tags: [ 'role::logrotate' ]
    logrotate__dependent_config:
      - '{{ php__logrotate__dependent_config }}'

  - role: geerlingguy.composer
    tags: [ 'role::composer' ]

  - role: debops.nginx
    tags: [ 'role::nginx' ]
    nginx__dependent_upstreams:
      - '{{ volkszaehler__nginx__dependent_upstreams }}'
    nginx__dependent_servers:
      - '{{ volkszaehler__nginx__dependent_servers }}'

  - role: debops-contrib.volkszaehler
    tags: [ 'role::volkszaehler' ]
```

These playbooks are shipped with this role under `./docs/playbooks/` from which you can symlink them to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::volkszaehler:env` Environment role tag, should be used in the playbook to execute a special environment role contained in the main role. The environment role prepares the environment for other dependency roles to work correctly.

`role::volkszaehler` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

`role::volkszaehler:pkgs` Tasks related to system package management like installing or removing packages.

debops-contrib.volkszaehler default variables

Sections

System packages

volkszaehler__base_packages

List of base packages to install.

```
volkszaehler__base_packages:
  - 'git-core'

  - '{{ [ "php-xml", "php-mbstring" ]
        if (ansible_local|d() and ansible_local.php|d() and
            ansible_local.php.version|d() and ansible_local.php.version|version_
↪compare("7", ">="))
        else [] }}'
```

volkszaehler__optional_packages

List of optional packages to install.

```
volkszaehler__optional_packages:
  # Server-side chart generation for volkszaehler.
  - 'libphp-jpgraph'
```

volkszaehler__packages

List of additional packages to install as configured by the system administrator.

```
volkszaehler__packages: []
```

volkszaehler__deploy_state

What is the desired state which this role should achieve? Possible options:

present Default. Ensure that volkszaehler is installed and configured as requested.

absent Ensure that volkszaehler is uninstalled and it's configuration is removed.

purged Same as `absent` but additionally also ensures that the database and other persistent data is removed.

```
volkszaehler__deploy_state: 'present'
```

FQDN and DNS addresses

volkszaehler__fqdn

The Fully Qualified Domain Name of the volkszaehler instance. This address is used to configure the webserver frontend.

```
volkszaehler__fqdn: 'vz.{{ volkszaehler__domain }}'
```

volkszaehler__domain

Domain that will be configured for the volkszaehler instance.

```
volkszaehler__domain: '{{ ansible_local.core.domain
                          if (ansible_local|d() and ansible_local.core|d() and
                              ansible_local.core.domain|d())
                          else (ansible_domain if ansible_domain else ansible_
↪hostname) }}'
```

Database configuration

volkszaehler__database

Autodetected variable containing the database management system which should be used. The supported and tested option is `mariadb` .

Refer to *Getting started* for details.

```
volkszaehler__database: '{{ ansible_local.volkszaehler.database
                            if (ansible_local|d() and ansible_local.volkszaehler|d()␣
↪and
                                ansible_local.volkszaehler.database|d())
                            else ("mariadb"
                                  if (ansible_local|d() and ansible_local.mariadb is␣
↪defined)
                                  else ("postgresql"
                                        if (ansible_local|d() and ansible_local.
↪postgresql is defined)
                                        else "no-database-detected")) }}'
```

volkszaehler__database_doctrine_map

Database name mapping from the names as used in DebOps to doctrine database driver names.

```
volkszaehler__database_doctrine_map:
  'mariadb': 'pdo_mysql'
  'postgresql': 'pdo_pgsql'
  'sqlite': 'pdo_sqlite'

  # Legacy:
  'mysql': 'pdo_mysql'
```

volkszaehler__database_server

FQDN of the database server. It will be configured by the debops.mariadb or debops.postgresql role.

```
volkszaehler__database_server: '{{ ansible_local[volkszaehler__database].server }}'
```

volkszaehler__database_port

Port database is listening on.

```
volkszaehler__database_port: '{{ ansible_local[volkszaehler__database].port }}'
```

volkszaehler__database_name

Name of the database to use for volkszaehler.

```
volkszaehler__database_name: 'volkszaehler'
```

volkszaehler__database_user

Database user to use for volkszaehler.

```
volkszaehler__database_user: 'volkszaehler'
```

volkszaehler__database_password_path

Path to database password file.

```
volkszaehler__database_password_path: '{{ secret + "/" + volkszaehler__database + "/"
                                        + ansible_local[volkszaehler__database].
↪delegate_to
                                        + (("/" + ansible_local[volkszaehler__
↪database].port)
                                           if (volkszaehler__database == "postgresql
↪")
                                           else "")
                                        + "/credentials/" + volkszaehler__database_
↪user + "/password" }}'
```

volkszaehler__database_password

Database password for volkszaehler.

```
volkszaehler__database_password: '{{ lookup("password", volkszaehler__database_
↪password_path + " length=48 chars=ascii_letters,digits,.:-_") }}'
```

volkszaehler__database_user_priv

Privileges of the *volkszaehler__database_user*.

```
volkszaehler__database_user_priv: |
  {{
    [
      volkszaehler__database_name + ".*:USAGE",
      volkszaehler__database_name + ".*:SELECT,UPDATE,INSERT",
    ] + ([
          volkszaehler__database_name + ".*:DELETE",
      ] if (volkszaehler__allow_channel_deletion|bool)
        else [
          volkszaehler__database_name + ".entities_in_aggregator:CREATE,DELETE",
          volkszaehler__database_name + ".properties:CREATE,DELETE",
        ]
    )
  }}
```

volkszaehler__database_demo_insert

Insert demo data in to database?

```
volkszaehler__database_demo_insert: False
```

volkszaehler__allow_channel_deletion

Allow channel deletion? Note that you might not be able to change this after the database user has been created. You can drop the database user manually and let the role re-create the user to enforce new privileges.

```
volkszaehler__allow_channel_deletion: False
```

PHP configuration

volkszaehler__base_php_packages

List of base PHP packages required by volkszaehler.

```
volkszaehler__base_php_packages:
  - 'doctrine-orm'
  - 'doctrine-dbal'
  - 'symfony-console'
  - '{{ [ "symfony-http-foundation" ] if (not (ansible_distribution == "Ubuntu" and␣
↪ansible_distribution_release in ["trusty"])) else [] }}'
  - '{{ [ "symfony-http-kernel" ] if (not (ansible_distribution == "Ubuntu" and␣
↪ansible_distribution_release in ["trusty"])) else [] }}'
  - 'symfony-routing'

  - '{{ [ "mysql" ] if (volkszaehler__database in [ "mariadb", "mysql" ]) else [] }}'
  - '{{ [ "pgsql" ] if (volkszaehler__database in [ "postgresql" ]) else [] }}'
  - '{{ ["libapache2-mod-php"] if (volkszaehler__webserver == "apache") else [] }}'

  ## Included in normal PHP installations but require it here because it is
  ## used internally by the role:
  - 'json'
```

volkszaehler__optional_php_packages

List of optional PHP packages for volkszaehler.

```
volkszaehler__optional_php_packages:
  # Server-side chart generation for volkszaehler.
  - 'libphp-jpgraph'

  - 'apcu'
```

volkszaehler__max_file_size

Maximum upload size.

```
volkszaehler__max_file_size: '1M'
```

Webserver configuration

volkszaehler__webserver

Autodetected variable containing the webserver which should be used.

Refer to *Getting started* for details.

```
volkszaehler__webserver: '{{ ansible_local.volkszaehler.webserver
                             if (ansible_local|d() and ansible_local.volkszaehler|d()␣
↪and
                                 ansible_local.volkszaehler.webserver|d())
                             else ("apache"
                                   if (ansible_local|d() and ansible_local.apache|d()␣
↪and ansible_local.apache.enabled|d()|bool)
```

```
                                    else ("nginx"
                                          if (ansible_local|d() and ansible_local.
↪nginx|d() and ansible_local.nginx.enabled|d()|bool)
                                          else "no-webserver-detected")) }}'
```

volkszaehler__webserver_http_methods

List of allowed HTTP methods.

```
volkszaehler__webserver_http_methods: |
  {{ [
       'GET',
       'HEAD',
       'POST',
       'PATCH',
     ] + ([ 'DELETE' ]
          if (volkszaehler__allow_channel_deletion|bool)
          else [])
  }}
```

volkszaehler__apache_modules

Dict of required Apache modules.

```
volkszaehler__apache_modules:
  'rewrite': {}
```

Directory paths

volkszaehler__home_path

The volkszaehler system account home directory.

```
volkszaehler__home_path: '{{ (ansible_local.nginx.www
                              if (ansible_local|d() and ansible_local.nginx|d()
                                  and ansible_local.nginx.www|d())
                              else "/srv/www") + "/" + volkszaehler__user }}'
```

volkszaehler__www_path

Base web root directory for volkszaehler.

```
volkszaehler__www_path: '{{ volkszaehler__git_dest + "/htdocs" }}'
```

System user and group

volkszaehler__user

System UNIX account used by the volkszaehler middleware and for application deployment.

```
volkszaehler__user: 'volkszaehler'
```

volkszaehler__group

System UNIX group used by the volkszaehler middleware.

```
volkszaehler__group: 'volkszaehler'
```

volkszaehler__gecos

Contents of the GECOS field set for the volkszaehler account.

```
volkszaehler__gecos: 'volkszaehler.org'
```

volkszaehler__shell

The default shell set on the volkszaehler account.

```
volkszaehler__shell: '/usr/sbin/nologin'
```

Volkszaehler sources and deployment

volkszaehler__git_repo

The URI of the volkszaehler git source repository.

```
volkszaehler__git_repo: 'https://github.com/volkszaehler/volkszaehler.org.git'
```

volkszaehler__git_version

The git branch or tag which will be installed. Defaults to the commit hash of latest master as the role was written. This is done because volkszaehler development is not cryptographically signed and this role wants to comply with the DebOps Software Source Policy.

```
volkszaehler__git_version: 'fadb821555527d0fb4d729a3f62e238cde10f168'
```

volkszaehler__git_dest

Path where the volkszaehler sources will be checked out (installation path).

```
volkszaehler__git_dest: '{{ volkszaehler__home_path + "/volkszaehler.org" }}'
```

volkszaehler__git_recursive

Should the git repository be cloned recursively?

```
volkszaehler__git_recursive: False
```

volkszaehler__git_update

Should new revisions be retrieved from the origin repository?

```
volkszaehler__git_update: True
```

Volkszaehler configuration

volkszaehler__config_user

The system owner of the `etc/volkszaehler.conf.php` file.

```
volkszaehler__config_user: '{{ volkszaehler__user
                               if (volkszaehler__webserver in ["apache"])
                               else "root" }}'
```

volkszaehler__config_group

The system group of the `etc/volkszaehler.conf.php` file.

```
volkszaehler__config_group: '{{ (ansible_local.apache.user
                                  if (ansible_local|d() and ansible_local.apache|d()␣
↪and
                                      ansible_local.apache.user|d())
                                 else "www-data")
                                if (volkszaehler__webserver in ["apache"])
                                else volkszaehler__user }}'
```

volkszaehler__locale

The default locale to use, ordered by preference. See setlocale for details.

```
volkszaehler__locale:
  - 'en_US'
  - 'de_DE'
  - 'C'
```

volkszaehler__upstream_config

Configuration as defined by upstream volkszaehler in `volkszaehler.conf.template.php`.

```
volkszaehler__upstream_config:

  push:
    # Set to True to enable push updates.
    enabled: False
    server: 5582
    broadcast: 8082
    routes:
      wamp:
        - '/'
        - '/ws'
      websocket: []

  security:
    maxbodysize: False

  locale:
    - 'en_US'
    - 'de_DE'
    - 'C'

  # Only used by jpGraph for server-side plotting!
  colors:
    - '#83CAFF'
    - '#7E0021'
    - '#579D1C'
    - '#FFD320'
    - '#FF420E'
    - '#004586'
    - '#0084D1'
    - '#C5000B'
    - '#FF950E'
    - '#4B1F6F'
    - '#AECF00'
    - '#314004'
```

```
  devmode: False
  cache:
    # Only used if devmode == False
    ttl: 3600

  debug: 0
```

volkszaehler__role_config

This dict is managed by the role itself, controlled by other default variables.

```
volkszaehler__role_config:

  db:
    driver: '{{ volkszaehler__database_doctrine_map[volkszaehler__database] }}'
    host: '{{ volkszaehler__database_server }}'
    user: '{{ volkszaehler__database_user }}'
    password: '{{ volkszaehler__database_password }}'
    dbname: '{{ volkszaehler__database_name }}'
    charset: 'UTF8'

  locale: '{{ volkszaehler__locale }}'

  security:
    maxbodysize: '{{ volkszaehler__max_file_size }}'
```

volkszaehler__config

This dict is intended to be used in Ansible's global inventory as needed.

```
volkszaehler__config: {}
```

volkszaehler__group_config

This dict is intended to be used in a host inventory group of Ansible (only one host group is supported) as needed.

```
volkszaehler__group_config: {}
```

volkszaehler__host_config

This dict is intended to be used in the inventory of hosts as needed.

```
volkszaehler__host_config: {}
```

volkszaehler__combined_config

The configuration written to `etc/volkszaehler.conf.php`.

```
volkszaehler__combined_config: '{{ volkszaehler__upstream_config
                                   | combine(
                                       volkszaehler__role_config,
                                       volkszaehler__config,
                                       volkszaehler__group_config,
                                       volkszaehler__host_config) }}'
```

Configuration for other Ansible roles

volkszaehler__mariadb__dependent_databases

Configuration of the volkszaehler database managed by the debops.mariadb role.

```
volkszaehler__mariadb__dependent_databases:

  - database: '{{ volkszaehler__database_name }}'
    state: '{{ "present" if (volkszaehler__deploy_state != "purged") else "absent" }}'
```

volkszaehler__mariadb__dependent_users

Configuration of the volkszaehler database user managed by the debops.mariadb role.

```
volkszaehler__mariadb__dependent_users:

  - database: '{{ volkszaehler__database_name }}'
    state: '{{ "present" if (volkszaehler__deploy_state == "present") else "absent" }}
↪'
    user: '{{ volkszaehler__database_user }}'
    owner: '{{ volkszaehler__user }}'
    group: '{{ volkszaehler__group }}'
    home: '{{ volkszaehler__home_path }}'
    system: True
    password: '{{ volkszaehler__database_password }}'
    priv_default: False
    priv_aux: False
    priv: '{{ volkszaehler__database_user_priv }}'
```

volkszaehler__php__dependent_packages

List of PHP packages to install using the debops.php role.

```
volkszaehler__php__dependent_packages:
  - '{{ volkszaehler__base_php_packages }}'
  - '{{ volkszaehler__optional_php_packages }}'
```

volkszaehler__php__dependent_pools

Configuration of the volkszaehler PHP-FPM pool managed by the debops.php role.

```
volkszaehler__php__dependent_pools:

  - name: 'volkszaehler'
    user: '{{ volkszaehler__user }}'
    group: '{{ volkszaehler__group }}'
    state: '{{ "present" if (volkszaehler__deploy_state == "present") else "absent" }}
↪'

    php_admin_values:
      post_max_size:       '{{ volkszaehler__max_file_size }}'
      upload_max_filesize: '{{ volkszaehler__max_file_size }}'
```

volkszaehler__nginx__dependent_upstreams

Configuration of the volkszaehler nginx upstream, used by the debops.nginx Ansible role.

```
volkszaehler__nginx__dependent_upstreams:

  - name: 'php_volkszaehler'
    type: 'php'
    php_pool: 'volkszaehler'
    state: '{{ "present" if (volkszaehler__deploy_state == "present") else "absent" }}
↪'
```

volkszaehler__nginx__dependent_servers

Configuration of the volkszaehler nginx server, used by the debops.nginx Ansible role.

```
volkszaehler__nginx__dependent_servers:

  - name: '{{ volkszaehler__fqdn }}'
    filename: 'debops.volkszaehler'
    by_role: 'debops-contrib.volkszaehler'
    state: '{{ "present" if (volkszaehler__deploy_state == "present") else "absent" }}
↪'
    type: 'php'
    root: '{{ volkszaehler__www_path }}'
    php_upstream: 'php_volkszaehler'
    csp: "default-src 'self'; connect-src * ws: wss: http: https:; script-src 'self'
↪'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline';"
    csp_enabled: True
    php_limit_except: '{{ volkszaehler__webserver_http_methods }}'

    options: |
      client_max_body_size {{ volkszaehler__max_file_size }};
      client_body_buffer_size 128k;

    location:
      '/': |
        rewrite ^/middleware/(.*)  /middleware.php/$1  last;
        rewrite ^/frontend/(.*)    /$1                 last;
```

volkszaehler__apache__dependent_vhosts

Apache virtual host managed by the debops.apache role.

```
volkszaehler__apache__dependent_vhosts:

  - type: 'default'
    name: '{{ volkszaehler__fqdn }}'
    filename: 'debops.volkszaehler'
    by_role: 'debops-contrib.volkszaehler'
    state: '{{ "present" if (volkszaehler__deploy_state == "present") else "absent" }}
↪'
    root: '{{ volkszaehler__www_path }}'
    options: 'Indexes FollowSymLinks MultiViews'
    allow_override: 'FileInfo Limit Options Indexes AuthConfig'
```

Copyright

```
DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.volkszaehler

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.volkszaehler master - unreleased

debops-contrib.volkszaehler v0.1.0 - 2017-04-17

Added

- Initial coding and design. [ypid]

Ansible role: debops-contrib.x2go_server

Introduction

The `debops-contrib.x2go_server` role allows you to setup and mange X2Go on the server-side. X2Go enables you to access a graphical desktop of a computer over a low bandwidth (or high bandwidth) connection.

Installation

This role requires at least Ansible `v2.1.3`. To install it, run:

```
ansible-galaxy install debops-contrib.x2go_server
```

Getting started

- *Example inventory*
- *Example playbook*
- *Ansible tags*

Example inventory

To setup and manage the X2Go server, add the hosts to the `debops_service_x2go_server` Ansible inventory host group:

```
[debops_service_x2go_server]
hostname
```

If you are using debops.sshd for configuring your OpenSSH server, you will need to adopt some of the defaults of this role to allow X2Go clients to connect to the X2Go server via SSH. The recommended way to do those adoptions is to symlink the `docs/inventory/debops_service_x2go_server_global_role_vars` file shipped with this role into your inventory under `ansible/inventory/group_vars/debops_service_x2go_server_global_role_vars` and include all hosts from the `debops_service_x2go_server` in the `debops_service_x2go_server_global_role_vars` host group by adding this:

```
[debops_service_x2go_server_global_role_vars:children]
debops_service_x2go_server
```

into your host inventory which makes the following adjustments to the defaults variables of other roles:

```
---
# .. vim: foldmarker=[[[,]]]:foldmethod=marker

# Required configuration for debops.sshd [[[

# Require the following cryptography methods as X2Go seems to only support a
# subset of which OpenSSH does support.
sshd__ciphers_additional: '{{ [ "aes256-ctr" ] if (x2go_server__deploy_state|d(
 ↪"present") == "present") else [] }}'
sshd__kex_algorithms_additional: '{{ [ "curve25519-sha256@libssh.org" ] if (x2go_
 ↪server__deploy_state|d("present") == "present") else [] }}'
sshd__macs_additional: '{{ [ "hmac-sha1" ] if (x2go_server__deploy_state|d("present")␣
 ↪== "present") else [] }}'

# ]]]

# Optional configuration for debops.sshd [[[

# Enabled for performance reasons. X11 forwarding over SSH is not directly used.
# http://wiki.x2go.org/doku.php/doc:faq:start#why_am_i_told_to_enable_x11_forwarding_
 ↪with_x2go_i_thought_that_x2go_uses_nx-libs_instead_of_x11_forwarding
# https://www.nomachine.com/AR05D00391
sshd__x11_forwarding: 'yes'

# ]]]
```

Example playbook

Here's an example playbook that can be used to setup and manage X2Go server:

```
---

- name: Setup and manage the server-side of X2Go
  hosts: [ 'debops_service_x2go_server' ]
  become: True
```

```
environment: '{{ inventory__environment | d({})
                 | combine(inventory__group_environment | d({}))
                 | combine(inventory__host_environment  | d({})) }}'

roles:

  - role: debops-contrib.x2go_server
    tags: [ 'role::x2go_server' ]
```

This playbooks is shipped with this role under `docs/playbooks/x2go_server.yml` from which you can symlink it to your playbook directory. In case you use multiple DebOps Contrib roles, consider using the DebOps Contrib playbooks.

Ansible tags

You can use Ansible `--tags` or `--skip-tags` parameters to limit what tasks are performed during Ansible run. This can be used after a host was first configured to speed up playbook execution, when you are sure that most of the configuration is already in the desired state.

Available role tags:

`role::x2go_server` Main role tag, should be used in the playbook to execute all of the role tasks as well as role dependencies.

debops-contrib.x2go_server default variables

Sections

Packages and installation

`x2go_server__base_packages`

List of base packages to install. You can checkout the Ansible role ypid.packages which can install additional packages to support your desktop environment better.

```
x2go_server__base_packages:
  - '{{ [ "x2go-keyring" ] if (ansible_distribution in [ "Debian" ]) else [] }}'
  - 'x2goserver'
  - 'x2goserver-xsession'
```

`x2go_server__deploy_state`

What is the desired state which this role should achieve? Possible options:

`present` Default. Ensure that X2Go is installed and configured as requested.

`absent` Ensure that X2Go is uninstalled and it's configuration is removed.

```
x2go_server__deploy_state: 'present'
```

Software sources

x2go_server__apt_repo_key_fingerprint_map

APT PGP key fingerprint used to sign the upstream X2Go repository and it's packages per distributions corresponding to `ansible_distribution`. Use `{{ omit }}` to omit the key fingerprint fetch all together. For example, using PPAs does automatically fetch the correct PGP public key somehow. Refer to the official X2Go Dokumentation for details.

```
x2go_server__apt_repo_key_fingerprint_map:
  'Ubuntu': '{{ omit }}'
  'Linuxmint': '{{ omit }}'
  'default': '972FD88FA0BAFB578D0476DFE1F958385BFE2B6E'
```

x2go_server__upstream_release_channel

Release channel to use. Choices:

main Release builds, default.

heuler Nightly builds.

```
x2go_server__upstream_release_channel: 'main'
```

x2go_server__upstream_mirror_url

URL of the X2Go upstream APT repository.

```
x2go_server__upstream_mirror_url: 'http://packages.x2go.org/{{ ansible_distribution |␣
↪lower }}/'
```

x2go_server__upstream_repository_map

APT repository definition for using the X2Go upstream repository per distribution corresponding to `ansible_distribution`.

```
x2go_server__upstream_repository_map:
  'Ubuntu': 'ppa:x2go/{{ x2go_server__ppa_release_channel_map[x2go_server__upstream_
↪release_channel] }}'
  'Linuxmint': 'ppa:x2go/{{ x2go_server__ppa_release_channel_map[x2go_server__
↪upstream_release_channel] }}'
  'default': 'deb {{ x2go_server__upstream_mirror_url }} {{ ansible_distribution_
↪release }} {{ x2go_server__upstream_release_channel }}'
```

x2go_server__ppa_release_channel_map

Mapping from *x2go_server__upstream_release_channel* to the names used in PPAs.

```
x2go_server__ppa_release_channel_map:
  'main': 'stable'
```

Copyright

```
debops-contrib.x2go_server - Setup and manage the server-side of X2go

Copyright (C) 2016 Robin Schneider <ypid@riseup.net>
Copyright (C) 2016 DebOps https://debops.org/

This Ansible role is part of DebOps.

DebOps is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License version 3, as
published by the Free Software Foundation.

DebOps is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with DebOps. If not, see https://www.gnu.org/licenses/.
```

Changelog

debops-contrib.x2go_server

This project adheres to Semantic Versioning and human-readable changelog.

The current role maintainer is ypid.

debops-contrib.x2go_server v0.1.0 - unreleased

Added

- Initial coding and design. [ypid]

Changed

- Rename `x2go_server__apt_repo_key_fingerprint_override_map` to *x2go_server__apt_repo_key_fingerprint_map*

  Rename `x2go_server__upstream_repository_override_map` to *x2go_server__upstream_repository_map*.

  And just use a default entry. [ypid]

- Require Ansible v2.1.3 to mitigate CVE-2016-8614:

  ```
  apt_key module not properly validating keys in some situations - resolved in␣
  ↪Ansible 2.1.3/2.2.
  ```

  Refer to Ansible Security and apt_key module does not verify key fingerprints for details.

  Note that Ansible currently does not check min version of roles (Tracked upstream). Please only use v2.1.3 or above to run this role! [ypid]

F

G

K

R

S

V

X