Usage of Ansible in Debian

Christian Külker

2024-03-09

Contents

1 Introduction

Ansible supports Debian-specific methods for installing and configuring Debian GNU/Linux. Some aspects, such as installing and updating packages, are transparent, while others, like package configuration, are less so. This document aims to elucidate certain Debian-specific aspects of Ansible.

For Ansible to support Debian-specific tasks, the command-line `debconf` tools are a prerequisite and need to be installed.

2 Configuration of Packages

Packages can be configured at two different times:

1. Before installation
2. After installation

It is generally advisable to configure packages prior to installation.

2.1 Before Installation Configuration - BIC

The built-in Ansible module, `ansible.builtin.debconf`, facilitates pre-installation configuration. It serves as an interface to the **debconf** database, which a package consults for debconf questions (essentially configuration settings).

To determine a package's configurable options via the `debconf` database, use the `debconf-show` command. It queries the package database about specific packages.

Unfortunately, developing such Ansible playbooks requires **two** Debian installations: one (System D) where the package is manually installed, and another pristine (System A) where the package is installed with Ansible. While tools like `debconf-show` can be used on System D, they cannot be used on System A. Consequently, Ansible playbooks developed in this manner may fail on System D but likely succeed on System A. It should be noted that when utilizing such tools in this document, they should work on the Debian system (System D) not managed by Ansible. Additionally, when executing Ansible in this context, the package should not be installed prior to running the Ansible playbook. However, a second run of the Ansible playbook might not correctly reconfigure the package. This one-shot operation should ideally lead to success, but there are cases where it may not, such as if a local admin manually changes a value, potentially resetting the `debconf` database to the new value before the package is reconfigured.

Sysstat Example:

1. An Ansible playbook pre-configures the `debconf` database to enable the `sysstat` service of the `syssstat` package
2. Upon execution, Ansible installs `sysstat`, initializing it with the pre-seeded `debconf` settings to be enabled.
3. This results in ENABLED="true" in `/etc/default/sysstat`.

However, if a local system administrator sets `ENABLED="false"` and expects the same Ansible playbook to revert this using `debconf`, this is unlikely to occur. The package's configuration script, found at `/var/lib/dpkg/info/sysstat.config`, will overwrite the `debconf` value.

Thus, playbooks employing the before installation configuration technique **may not** or are even **likely not** idempotent.

However, let's continue with this approach. On System D, you can query the `debconf` database to determine which "debconf questions" are possible to set for a package. For example, for `sysstat`, you might find:

```
# As root on System D
debconf-show info sysstat
  sysstat/remove_files: true
  sysstat/enable: false
```

This reveals two supported questions:

1. `sysstat/remove_files`
2. `sysstat/enable`

This information helps in developing the playbook's initial part.

```
# sysstat_bic.yaml
# Quick Guide test for DevOps/Deployment/Ansible/debian.md
#   - "Before Installation Configuration - BIC"
#     -  __Order of install and pre-configure matters__
- hosts: role_client
  gather_facts: no
  become: yes
  vars:
      ns: sysstat
      packages:
          # SysStat (iostat, ...) System performance tools for Linux
        - sysstat
  tasks:
    - name: "{{ ns }}: Pre-configure sysstat package - BIC"
      ansible.builtin.debconf:
        name: sysstat
        question: sysstat/enable
        vtype: "boolean"
        value: "true"
      become: yes
      register: dpkg_reconfigure
    - name: "{{ ns }}: Install and update packages - BIC"
      apt:
        name: "{{ packages }}"
        state: latest
```

It can be observed that this Ansible playbook is concise.

```
# As root on System A - before running the Ansible playbook
debconf-show info sysstat
```

When Ansible is run with `-v` on System A, the setup step can be viewed as follows.

```
changed: [HOSTNAME] => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "current": {
        "sysstat/enable": "true"
    },
    "diff": {},
    "invocation": {
        "module_args": {
            "name": "sysstat",
            "question": "sysstat/enable",
            "unseen": false,
            "value": "true",
            "vtype": "boolean"
        }
    },
    "previous": {
        "sysstat/enable": ""
    }
}
```

After executing the playbook on System A, the `debconf` database should reflect the changes.

```
# As root on System A - after running the Ansible playbook
debconf-show info sysstat
* sysstat/enable: true
  sysstat/remove_files: true
```

The asterisk `*` indicates that a question has been asked. Finally, the defaults in `/etc/default/sysstat` should resemble:

```
#
# Default settings for /etc/init.d/sysstat, /etc/cron.d/sysstat
# and /etc/cron.daily/sysstat files
#

# Should sadc collect system activity informations? Valid values
# are "true" and "false". Please do not put other values, they
# will be overwritten by debconf!
ENABLED="true"
```

Additionally, the server should be running:

```
systemctl status sysstat
○ sysstat.service - Resets System Activity Logs
     Loaded: loaded (/lib/systemd/system/sysstat.service; enabled; preset:
      ↪  \
                    enabled)
     Active: inactive (dead)
       Docs: man:sa1(8)
             man:sadc(8)
             man:sar(1)
```

Therefore, the order of operations is crucial. If a local system administrator disables the service by setting `ENABLED="false"` in `/etc/default/sysstat`, this playbook will **not** capture or correct it. Furthermore, if `dpkg-reconfigure sysstat` is used, this playbook will **not** revert it.

2.2 After Installation Configuration

This approach doesn't assume prior installation of a package. However if not installed, it will be installed **first**, but in any case it will be reconfigured **after** installation.

We will use the `sysstat` package as an example. Note that if it is installed manually, the server is disabled via `/etc/default/sysstat`. This playbook aims to alter that configuration.

```
# On a sysstat preinstalled system:
systemctl status sysstat
○ sysstat.service - Resets System Activity Logs
     Loaded: loaded (/lib/systemd/system/sysstat.service; disabled;
      ↪  preset: \
                    enabled)
     Active: inactive (dead)
       Docs: man:sa1(8)
             man:sadc(8)
             man:sar(1)
```

The `ansible.builtin.debconf` module states that it is Debian policy-compliant to move `<package>.config` before executing `dpkg-reconfigure`.

```
# sysstat_aic.yaml
# Quick Guide test for DevOps/Deployment/Ansible/debian.md
#   - "After Installation Configuration - AIC"
#     -  __Order of install and pre-configure matters__
```

```yaml
- hosts: role_client
  gather_facts: no
  become: yes
  vars:
    ns: sysstat
    cfg_enabled: /var/lib/dpkg/info/sysstat.config
    cfg_disabled: /var/lib/dpkg/info/sysstat.config.DISABLED
    packages:
      # SysStat (iostat, ...) System performance tools for Linux
      - sysstat
  tasks:
    - name: "{{ ns }}: Install and update packages - AIC"
      apt:
        name: "{{ packages }}"
        state: latest
      register: package_installed
    - name: "{{ ns }}: Pre-configure sysstat package - AIC"
      ansible.builtin.debconf:
        name: sysstat
        question: sysstat/enable
        vtype: "boolean"
        value: "true"
      become: yes
      register: debconf_setting
      notify: reconfigure sysstat
    - name: "{{ ns }}: Move sysstat.config => sysstat.config.DISABLED -
↪  AIC"
      command: "mv {{cfg_enabled}} {{cfg_disabled}}"
      become: yes
      when: debconf_setting is changed
  handlers:
    - name: reconfigure sysstat
      command: dpkg-reconfigure -f noninteractive sysstat
      become: yes
      when: debconf_setting is changed
    - name: "{{ ns }}: Restore sysstat config file post-reconfiguration -
↪  AIC"
      command: "mv {{cfg_disabled}} {{cfg_enabled}}"
      when: debconf_setting is changed
      become: yes
      listen: "reconfigure sysstat"
```

After running this Ansible playbook for the first time, five tasks are executed: four are

'changed,' and one is 'ok.' Consequently, the server is enabled and running.

```
systemctl status sysstat
○ sysstat.service - Resets System Activity Logs
     Loaded: loaded (/lib/systemd/system/sysstat.service; enabled; preset:
      ↪  \
                    enabled)
     Active: inactive (dead)
       Docs: man:sa1(8)
             man:sadc(8)
             man:sar(1)
```

Furthermore, the package has the correct `debconf` values.

```
debconf-show info sysstat
* sysstat/enable: true
  sysstat/remove_files: true
```

A subsequent run will yield two 'ok' and one 'skipped' task. If the local system administrator disables the service by setting `ENABLED="false"` in `/etc/default/sysstat`, this playbook will **not** capture it. However, if local system administrator use `dpkg-reconfigure sysstat`, this playbook will revert the configuration.

For a probable non-Debian policy-compliant approach that catches changes in `/etc/default/sysstat`, refer to the "Ansible and Command-line" section of the SysStat document and compare the differences in methods. Combining both approaches may offer the most comprehensive solution for managing `sysstat`.

3 History

Version	Date	Notes
0.1.0	2024-03-09	Initial release (ansible.buildin.debconf)

4 Disclaimer of Warranty

5 Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE INFORMATION, DOCUMENTS OR PROGRAMS AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE INFORMATION, DOCUMENTS OR PROGRAMS (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE INFORMATION, DOCUMENTS OR PROGRAMS TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.