



Technical Report

# Deploying NetApp E-Series with Ansible

## Automating E-Series

Nathan Swartz, NetApp  
June 2022 | TR-4574

### Abstract

Ansible is a simple yet powerful orchestration tool that is sweeping the IT world. NetApp® E-Series has joined the Ansible community to provide you with a high-quality solution for managing your E-Series storage systems, regardless of scale. This document grounds you in the Ansible philosophy and provides the necessary information to get started. It walks you through how to define your storage infrastructure in Ansible inventories and how to apply policies for each target system. It explains the NetApp E-Series SANtricity® collection and provides a practical guide for you to get started with the SANtricity collection, which can facilitate your end-to-end deployments and ongoing system management.

## TABLE OF CONTENTS

<b>Introduction .....</b>	<b>3</b>
About NetApp .....	3
About Ansible .....	3
NetApp E-Series contribution .....	3
NetApp E-Series and Ansible Key Benefits .....	4
<b>Solution architecture .....</b>	<b>4</b>
Ansible architecture overview .....	4
Inventories .....	5
Playbooks .....	6
Roles .....	7
SANtricity collection .....	8
SANtricity host collection .....	9
Tags .....	10
<b>Getting started .....</b>	<b>10</b>
<b>Support .....</b>	<b>11</b>
<b>Where to find additional information .....</b>	<b>11</b>
<b>Version history .....</b>	<b>12</b>

## LIST OF FIGURES

Figure 1) Ansible core architecture.....	5
Figure 2) Inventory file with variables. ....	5
Figure 3) Ansible working directory with inventory. ....	6
Figure 4) Ansible playbook with E-Series modules. ....	7
Figure 5) Ansible playbook with an E-Series host role. ....	7
Figure 6) E-Series storage array host file example. ....	8
Figure 7) Typical playbook to provision storage and ensure E-Series volumes are accessible.....	9
Figure 8) Example of an E-Series storage system inventory file. ....	9
Figure 9) Example of a host inventory file. ....	9

# Introduction

Today more than ever, IT professionals face complex challenges that demand reliable and consistent infrastructure configuration. Ansible promises to meet these expectations by providing a simple, reliable, yet powerful orchestration tool, working in collaboration with a strong community of partners. NetApp has joined this community so that NetApp customers can reap the benefits that Ansible provides with data storage.

## About NetApp

NetApp creates innovative products, such as storage systems and software that helps customers around the world store, manage, protect, and retain one of their most precious corporate assets, data. We are recognized throughout the industry for continually pushing the limits of today's technology so that our customers don't have to choose between saving money and acquiring the capabilities that they need to be successful.

We are finding ways to enable our customers to do things that they couldn't do before, at a speed that they never thought possible. We partner with industry leaders around the world to create efficient and cost-effective solutions that are optimized for customer IT needs and to deliver and to support those solutions worldwide. Customers value our teamwork, expertise, and passion for helping them succeed now and into the future.

For more information, see <http://www.netapp.com>.

## About Ansible

Ansible is an open source orchestration tool for infrastructure automation with a focus on desired states to make complex and difficult tasks repeatable and less susceptible to errors. Ansible can manage servers, storage, networking infrastructure, Docker containers, and much more. Owned by Red Hat since 2015, Ansible has gained widespread adoption throughout the IT industry and enjoys the support of a large and growing community.

Ansible is a simple yet powerful tool that requires no proprietary remote agents to be installed on target hosts and can be executed from any Linux or Mac system that has Python installed. Users define information about their infrastructure in a simple and easy-to-define collection of YAML files, called an inventory. Next, users define what policies or states the infrastructure should have in playbooks. Together, the inventory and playbooks are used by Ansible to execute policies on target hosts to ensure that expected states are independently applied. This process means that regardless of the target's previous state, Ansible makes the necessary adjustments to ensure that all states are as expected.

This versatile tool scales to manage your infrastructure, whether it means five servers or 500. This ease of scaling is accomplished by execution of playbooks against targeted host groups at the same time, making Ansible both efficient and scalable.

Although the project is managed by Red Hat, a diverse community of companies and individuals regularly contribute to it. And now, with the shift to collections, the Ansible ecosystem can grow at an even faster pace.

For more information, see <https://www.ansible.com>.

## NetApp E-Series contribution

NetApp E-Series engineers began contributing to the Ansible project in 2017, supporting the project by producing high-quality modules for E-Series storage systems. Modules are Ansible's tools for ensuring that the expected state or policies are enacted on managed systems. Ansible passes information about infrastructure to modules; determines whether changes are required; and if so, makes the necessary changes. The modules produced by NetApp E-Series engineers integrate with existing infrastructure by

making inquiries and changes through the embedded NetApp SANtricity Web Services. Beginning with the E2800 system, Web Services is installed by default on all E-Series storage systems. For earlier storage systems, you can also take advantage of this Ansible solution by installing SANtricity Web Services Proxy on an external host.

In early 2020, all NetApp E-Series contributions were made available in the NetApp certified SANtricity collection. We have partnered with the Red Hat Ansible team to provide our E-Series customers with Ansible support that is geared toward them. You can access this support through the Ansible Automation Hub. Along with the switch to Ansible collections, we have significantly expanded our system coverage from earlier content releases.

Beginning with Ansible 2.10, all community-supported contributions will be removed from the core Ansible project. These contributions include the familiar `netapp_e*` modules, which will continue to be available in the 2.9 release and are also available in the NetApp SANtricity collection to aid in the transition to collections.

NetApp E-Series engineers are actively producing solution-based collections to improve our customers' experience. For example, our BeeGFS collection uses Ansible to set up the BeeGFS parallel file system with high availability (HA) across systems in your inventory. For more information, see [Ansible Galaxy](#) or [GitHub](#).

## NetApp E-Series and Ansible Key Benefits

Why should you choose Ansible and E-Series for your configuration management?

- Ansible is a simple, powerful tool that you can use to easily maintain your complete infrastructures.
- This tool does not require proprietary servers or agents to be installed on your systems.
- Ansible's focus on the desired state helps significantly reduce complexity and downtime.
- You can apply playbooks and roles to any number of managed systems for easy deployment, management, and scaling.
- Ansible integrates well with the existing SANtricity Web Services API, so no additional tools are required.
- Ansible enables simple, end-to-end deployment and management of E-Series systems.

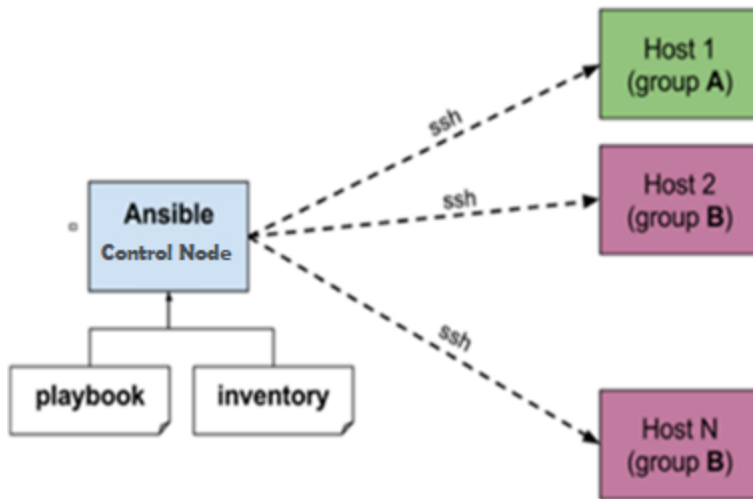
## Solution architecture

This section provides a helpful introduction into Ansible with E-Series and includes links to more information.

### Ansible architecture overview

Ansible uses information about your infrastructure in a collection of files that are called an inventory. This inventory is used in conjunction with modules to execute tasks on a selected host or group of hosts, sometimes called target nodes. These tasks enable you to enforce a policy or an expected state that the target should have. Ad hoc commands enable you to run a single task from the command line on multiple systems. More conveniently, you can combine related tasks into a single playbook and then execute it against target nodes. Figure 1 demonstrates this process of combining the playbook with inventory information so that Ansible can apply the expected policies on each targeted system.

Figure 1) Ansible core architecture.



Unless you specify otherwise, all playbook tasks are executed on each target by using the Secure Shell (SSH) protocol. Ansible has a wide range of connection plug-ins that you can use instead of SSH, such as Microsoft Windows Remote Management (WinRM), Docker, and kubectl. Ansible can also execute tasks on multiple systems concurrently, reducing the playbook execution time. For more information about connection plug-in options, see the [Ansible Connection Plugins](#) documentation page.

Ansible roles are a helpful extension to a playbook. A role is a collection of related tasks, variables, templates, files, plug-ins, and custom modules that are grouped purposefully for a specific function. An example is an Apache `webserver` role that, when applied to a target, makes the host a functioning Apache web server according to the details in the inventory. Roles are also useful for creating reusable task groupings that you commonly need across systems.

## Inventories

You use inventories to define information about target hosts and host groups. For readability, you should define these documents in YAML format. Figure 2 gives an example of an inventory that defines hosts and how they are related.

Figure 2) Inventory file with variables.

```
all:
  hosts:
    children:
      servers:
        webserver:
          ansible_host: 192.168.1.100
          ansible_ssh_user: admin
          ansible_become_pass: admin_pass
        database:
          ansible_host: 192.168.1.200
          ansible_ssh_user: admin
          ansible_become_pass: admin_pass
      eseries_arrays:
        hosts:
          metadata:
            eseries_system_serial: "012345678901"
            eseries_system_password: admin_password
            eseries_proxy_api_url: https://192.168.1.100:8443/devmgr/v2/
            eseries_proxy_api_password: admin_password
          storage:
            eseries_system_api_url: https://192.168.1.200:8443/devmgr/v2/
```

```
eseries_system_password: admin_password
```

This inventory also specifies variables that you can use directly in playbooks and roles. The default location for this file is `/etc/ansible/host` on the Ansible control node. However, you can use host files from anywhere by specifying the file in Ansible commands (see Step 5 in the “Getting started” section). This option can be useful for many reasons, such as in production and staging inventories or for logically separating infrastructure.

Although the example in Figure 2 works well for small inventories, it does not scale well. A better practice is to use the host inventory files to define only hosts and group relationships and then place the host and group variables into external files. Fortunately, Ansible makes it very easy for you to apply this best practice. You simply create two directories, `host_vars` and `group_vars`, and place in them YAML files that correspond to the name of the host or group as defined in the inventory file. You can find these directories in the same directory as the inventory file.

Figure 3 shows an example directory structure for the inventory from Figure 2.

**Figure 3) Ansible working directory with inventory.**

```
.
├── host.yml                # main inventory file
├── group_vars              # group files directory
│   ├── all.yml
│   ├── servers.yml
│   └── eseries_arrays.yml
├── host_vars               # host files directory
│   ├── database.yml
│   ├── metadata.yml
│   ├── storage.yml
│   └── webserver.yml
└── playbook.yml
```

The `group_vars` file names reflect each group name, and `host_vars` file names reflect the hosts that are specified. As a best practice, maintain version control on your inventory files to ensure that all changes are recorded and can be rolled back if necessary.

The `ansible-inventory` command facilitates Ansible inventory encryption to protect sensitive information at the variable or file level. For more information, see [Encrypting content with Ansible Vault](#).

For more information about how to build an inventory, see [How to build your inventory](#).

## Playbooks

Playbooks are Ansible’s orchestration language, written entirely in YAML, and are designed to be easy to understand, even for people who are new to Ansible. Each playbook starts by specifying the target host or group and then lists a series of tasks that defines what desired states or policies must be applied to each target. Although playbooks can be incredibly simple when you use well-made modules, they can quickly become unnecessarily complex when you design the playbook with intertwined shell commands. When you design Ansible playbooks, you should always prefer simplicity because complexity quickly becomes difficult to maintain (even for the original author). So, in general, avoid using the shell module when an existing module will suffice. This approach helps you avoid unnecessary complications and can simplify debugging when issues arise.

Figure 4 provides a simple playbook that targets the `eseries_arrays` inventory group. The playbook creates policies for each array to have a single volume group, with one volume with its write cache disabled. Each time this playbook is executed, Ansible uses the `na_santricity_storagepool` and `na_santricity_volume` modules to enforce these policies. Therefore, if the write cache option has been enabled inadvertently, when the playbook is run again, the change is recognized and Ansible disables the option. Also, note that the structure “`{...}`”, found on many lines, indicates how values are used from the inventory within tasks.

**Figure 4) Ansible playbook with E-Series modules.**

```
- host: eseries_arrays
gather_facts: false # Forces Ansible not to collect facts about target nodes
connection: local   # Forces Ansible to execute commands from the localhost
collections:
  - netapp_eseries.santricity
tasks:
  - name: Create a RAID6 volume group
    na_santricity_storagepool:
      ssid: "{{ eseries_ssid }}"
      api_url: "{{ eseries_api_url }}"
      api_username: "{{ eseries_api_username }}"
      api_password: "{{ eseries_api_password }}"
      validate_certs: false
      state: present
      name: example_volume_group
      raid_level: raid6
      criteria_drive_count: 10
  - name: Create volume in volume group
    na_santricity_volume:
      ssid: "{{ eseries_ssid }}"
      api_url: "{{ eseries_api_url }}"
      api_username: "{{ eseries_api_username }}"
      api_password: "{{ eseries_api_password }}"
      validate_certs: false
      state: present
      name: example_volume
      storage_pool_name: example_volume_group
      size: 100
      size_unit: gb
      write_cache_enable: false
```

To prevent Ansible from attempting to contact the storage system by using SSH, execute all E-Series modules locally on the control node by adding `connection: local` to the playbook, as shown in Figure 4, or by adding `ansible_connection: local` to the group or host inventory files. To prevent unnecessary fact collecting for each targeted storage system, disable the Ansible fact gathering by adding `gather_facts: false`, as shown in Figure 4.

Ansible provides documentation for available modules through its website or through the `ansible-doc` command. The `ansible-doc` command provides documentation that is specific to your modules. The `-s` or `--snippet` flag for the `ansible-doc` command is particularly helpful for playbook development; the output provides cut-and-paste YAML snippets for your playbook complete with comments that describe each option.

For more information, see [Intro to Playbooks](#) in the Ansible User Guide.

## Roles

Ansible roles are a collection of tasks, files, templates, variables, plug-ins, and modules that are used together to perform a complex task. Roles are called from a playbook as shown in Figure 5.

**Figure 5) Ansible playbook with an E-Series host role.**

```
- hosts: eseries_storage_arrays
gather_facts: false
collections:
  - netapp_eseries.santricity
tasks:
  - name: Provision E-Series storage
    import_role:
      name: nar_santricity_host
```

In this playbook, the role `nar_santricity_host` is called, which enacts policies for storage pools, volumes, hosts and host groups, volume-to-host mappings, and port interfaces. This role uses specific variables in the `eseries_storage_arrays` inventory as outlined by the role's documentation. Figure 6

shows an example of the inventory file that supports the role in creating policies such as the number of drives in the storage pool, volume sizes, and where to map the volumes.

**Figure 6) E-Series storage array host file example.**

```
eseries_system_api_url: https://192.168.1.200:8443/devmgr/v2/
eseries_system_password: admin_password
eseries_validate_certs: false

eseries_system_name: my_eseries_array
eseries_initiator_protocol: fc

eseries_management_interfaces:
  config_method: static
  subnet_mask: 255.255.255.0
  controller_a:
    - address: 192.168.1.100
  controller_b:
    - address: 192.168.1.102

eseries_storage_pool_configuration:
  - name: vg_example
    raid_level: raid6
    criteria_drive_count: 10
    volumes:
      - name: vol_example_[1-2]
        host: servers
        size: 2
        size_unit: tb
```

The use of roles is also an efficient way to build complete policy definitions, to apply best practices, and to enforce consistency in your inventory files.

## SANtricity collection

Collections are new with Ansible 2.9 and are a necessary result of the project's growth and success. Collections give contributors the means to release content on the contributors' timeline without waiting for the next Ansible release. Contributors can therefore be more nimble in supporting their content users.

The NetApp E-Series SANtricity collection contains the latest Ansible content from E-Series. The collection is certified by the Red Hat Ansible team, and support is available on the Ansible Automation Hub. The collection includes all modules, documentation, utilities, and roles. To help users migrate to the SANtricity collection with minimal changes to their playbooks, `netapp_e` prefixed modules are also included. However, it should be noted that going forward `netapp_e` modules are deprecated and must be replaced with their `na_santricity` prefixed counterparts because they will be removed in a future release.

This collection provides NetApp E-Series customers with a wide range of configuration options through the collection's modules. However, the real benefit of using the SANtricity collection is in the host and management roles. These roles provide ready-made, policy-based orchestration for E-Series systems that you enable simply by defining role variables.

After you have installed the physical hardware, the SANtricity roles can discover the Dynamic Host Configuration Protocol (DHCP)–assigned addresses. Use these addresses to set credentials and management interfaces so that your automation can perform full deployments for you without logging directly in to the devices. Moreover, the management role also confirms that alerts, the NetApp AutoSupport feature, logging, LDAP, firmware, and other components are configured as expected. The host role sets up host interfaces and provisions and maps storage, and if your servers are defined in your inventory, it correctly populates E-Series hosts and host groups automatically. With NetApp integration, you can use Ansible for all aspects of deploying and configuring your E-Series storage systems.



For more information about the NetApp E-Series SANtricity collection, see the NetApp E-Series SANtricity Collection on [Ansible Galaxy](#).

## SANtricity host collection

The NetApp SANtricity host collection maintains a number of roles that allow your host systems to access their E-Series storage. These roles can discover volumes mapped from any number of E-Series storage systems, install and configure the necessary communication protocols, and ensure that the volumes are formatted with persistent mount points. The roles can also clean up after themselves, leaving hosts in the same state they were before. For a complete inventory and playbook example for provisioning storage and making it accessible on the host by using the iSCSI protocol to communicate with the E-Series storage system, see Figure 7 through Figure 9.

**Figure 7) Typical playbook to provision storage and ensure E-Series volumes are accessible.**

```
- hosts: example_eseries_storage
  connection: local
  collections:
    - netapp_eseries.santricity
  tasks:
    - name: Provision and map E-Series storage to hosts.
      import_role:
        name: nar_santricity_host

- hosts: example_host
  collections:
    - netapp_eseries.host
  tasks:
    - name: Ensure any communication protocols are setup.
      import_role:
        name: storage_setup
    - name: Format and create persistent mount points for volumes.
      import_role:
        name: mount
```

**Figure 8) Example of an E-Series storage system inventory file.**

```
eseries_system_api_url: https://192.168.1.10:8443/devmgr/v2/
eseries_system_password: adminpass
eseries_initiator_protocol: iscsi
eseries_storage_pool_configuration:
  - name: vg1
    raid_level: raid6
    criteria_drive_count: 10
    common_volume_configuration:
      host_type: linux dm-mp
    volumes:
      - name: example_volume
        host: example_host
        size: 10240
        volume_metadata:
          format_type: ext4
          format_options: "-i 2048 -I 512 -J size=400 -Odir_index,filetype"
          mount_options: "sync,noatime,nodiratime,nobarrier"
          mount_dir: /mnt/
```

**Figure 9) Example of a host inventory file.**

```
ansible_host: 192.168.1.100
ansible_ssh_user: admin
ansible_become_password: adminpass

eseries_iscsi_interfaces:
  - name: eth1
    address: 192.168.2.100/24
  - name: eth2
```

```
address: 192.168.3.100/24
```

The first half of the playbook in Figure 7 does all that is necessary to provision and map volumes to the example host based on the `eseries_storage_pool_configuration` variable found in Figure 8. The second half of the playbook found in Figure 7 ensures that the host has the `open-iscsi` package installed, the interfaces are configured, the iSCSI sessions with storage are established, and the `example_volume` is formatted and mounted.

For more information about the NetApp E-Series SANtricity host collection, see the NetApp E-Series SANtricity host collection on [Ansible Galaxy](https://galaxy.ansible.com/netapp/eseries/santricity).

## Tags

Tags give you the ability to choose which policies to update or to check. For example, to execute only the policies for E-Series storage system interfaces, add `--tags interface`. You can skip specific groups of tags by using the `--skip-tags` flag.

For more information, see [https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_tags.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_tags.html).

## Getting started

Now that you have a foundation for Ansible with E-Series, here is a quick-start guide. To use Ansible with the NetApp E-Series SANtricity collection, follow these steps:

1. Install Ansible. For more information about how to install Ansible on your specific Linux distribution, see [Installing Ansible](#).
2. Install the SANtricity collection by using the `ansible-galaxy collection install netapp_eseries.santricity` command. If you need to install the collection offline, go to [https://galaxy.ansible.com/netapp\\_eseries/santricity](https://galaxy.ansible.com/netapp/eseries/santricity), download the tarball file, then install locally with the `ansible-galaxy collection install netapp_eseries-santricity-x.x.x.tar.gz` command.
3. Create project directories.

```
mkdir -p ~/ansible_project/{host_vars,group_vars}.
```

4. Create project inventory files for your project. Following are examples for you to copy, paste, and update to reflect your inventory. To see more variable options and details, click the Read Me button on the SANtricity home page ([https://galaxy.ansible.com/netapp\\_eseries/santricity](https://galaxy.ansible.com/netapp_eseries/santricity)).

```
# ~/ansible_project/hosts.yml
all:
  children:
    eseries_arrays:
      hosts:
        storage1:
        storage2:
    database_servers:
      hosts:
        storage_server:
```

```
# ~/ansible_project/host_vars/storage1.yml
eseries_system_name: my_eseries_array
eseries_system_api_url: https://192.168.1.200:8443/devmgr/v2/
eseries_system_password: admin_password
eseries_validate_certs: false
eseries_system_serial: 012345678901 # Use eseries_system_serial and eseries_subnet when dhcp is
eseries_subnet: 192.168.0.0/22      # available and system have not been configured. This will
                                   # allow the collection to discover and set the password.
eseries_initiator_protocol: fc
```

```

eseries_management_interfaces:
  config_method: static
  subnet_mask: 255.255.255.0
  gateway: 192.168.1.1
  controller_a:
    - address: 192.168.1.100
  controller_b:
    - address: 192.168.1.102
eseries_storage_pool_configuration:
  - name: vg_example
    raid_level: raid6
    criteria_drive_count: 10
    volumes:
      - name: vol_example_[1-2]
        host: servers
        size: 2048

```

5. Create a playbook in your project directory called `eseries_playbook.yml`. Copy the following playbook into your playbook.

```

# ~/ansible_project/eseries_playbook.yml
- hosts: eseries_arrays
  gather_facts: false
  collections:
    - netapp_eseries.santricity
  tasks:
    - name: Apply management-related tasks to E-Series storage systems
      import_role:
        name: nar_santricity_management
    - name: Apply host-related tasks to E-Series storage systems
      import_role:
        name: nar_santricity_host

```

6. Execute your Ansible playbook.

```
ansible-playbook -i hosts.yml eseries_playbook.yml
```

## Support

The NetApp E-Series team strives to produce high-quality Ansible modules and roles, which undergo specific testing to confirm that they meet expectations. For help with issues that you encounter, visit the respective collection's Ansible Galaxy homepage and click Issue Tracker, found on the top right of the page. See "Where to find additional information" for links to our E-Series Ansible collections found on Ansible Galaxy.

And for any of your Ansible questions, you also can join the discussion on the NetApp official Slack `#configurationmgmt` channel. To interact directly with the NetApp community, go to <https://netapp.io/slack/>.

## Where to find additional information

To learn more about the information that is described in this document, review the following documents and websites:

- NetApp E-Series SANtricity Collection  
[https://galaxy.ansible.com/netapp\\_eseries/santricity](https://galaxy.ansible.com/netapp_eseries/santricity)
- NetApp E-Series SANtricity Host Collection  
[https://galaxy.ansible.com/netapp\\_eseries/host](https://galaxy.ansible.com/netapp_eseries/host)
- NetApp E-Series BeeGFS Collection  
[https://galaxy.ansible.com/netapp\\_eseries/beegfs](https://galaxy.ansible.com/netapp_eseries/beegfs)

- Ansible Installation Guide—Linux distribution—specific installation guide  
[https://docs.ansible.com/ansible/latest/installation\\_guide/intro\\_installation.html](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html)
- Ansible Documentation—Detailed documentation on Ansible  
<https://docs.ansible.com/ansible/latest/index.html>
- Ansible Best Practices—Learn from the mistakes of others in this useful best practices guide  
[https://docs.ansible.com/ansible/latest/user\\_guide/playbooks\\_best\\_practices.html](https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html)
- Ansible Resources—Generic landing page for Ansible resources  
<https://www.ansible.com/resources>
- Ansible Galaxy—Search for available roles to use in your playbooks, including NetApp roles  
<https://galaxy.ansible.com/>
- Ansible Tower User Guide Overview—Overview of Ansible Tower, which is Ansible’s UI and REST API  
<http://docs.ansible.com/ansible-tower/latest/html/userguide/overview.html>

## Version history

Version	Date	Document version history
Version 4.0	June, 2022	Updates to fix obsolete links and various updates.
Version 3.0	November 2020	Update with SANtricity host collection.
Version 2.0	April 2020	Update with SANtricity collection.
Version 1.0	June 2019	Initial release.

Refer to the [Interoperability Matrix Tool \(IMT\)](#) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

### **Copyright Information**

Copyright © 2022 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. The U.S. Government has a non-exclusive, non-transferrable, non-sublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b).

### **Trademark Information**

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.

TR-4574-0622