Technical Report

Deploying NetApp E-Series with Ansible

Automating E-Series

Nathan Swartz, NetApp

May 2019 | 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 quality solution for managing your E-Series storage systems. 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, apply policies for each target system, and how to use E-Series modules and roles. Finally, we introduce the `netapp_eseries_host` role which enforces a set of policies for provisioning storage, creating hosts and host groups, mapping volumes, and configuring ports.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Today more than ever IT professionals are faced with complex challenges that demand reliable and expected performance with fast recovery should the unforeseeable happen. 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 to ensure our customers can enjoy the benefits Ansible provides with data storage.

1.1 About NetApp

NetApp creates innovative products, such as, storage systems and software that help customers around the world store, manage, protect, and retain one of their most precious corporate assets, their data. We are recognized throughout the industry for continually pushing the limits of today's technology so that our customers never have to choose between saving money and acquiring the capabilities they need to be successful.

We are finding ways to enable our customers to do things they couldn't do before at a speed they never thought possible. We partner with industry leaders to create the most efficient and cost-effective solutions optimized for their IT needs and to deliver and support them worldwide. Leading organizations worldwide count on NetApp for software, systems, and services to manage and store their data. Customers value our teamwork, expertise, and passion for helping them succeed now and into the future.

For more information, visit http://www.netapp.com.

1.2 About Ansible

Ansible is an open source orchestration tool for infrastructure automation with a focus on desired state 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 RedHat since 2015, Ansible has gained widespread adoption throughout the IT industry and enjoys a large support 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 host that has Python installed. Users define their infrastructure in a simple and easy-to-define collection of YAML or INI files called an inventory. Next, users define what policies or states the infrastructure should have in playbooks. Together, the inventory and playbook are used by Ansible to execute policies on target hosts to ensure that expected states are independently applied. This means that regardless of the target's previous state, Ansible makes the necessary adjustments to ensure they are as expected. This versatile tool will scale to manage your infrastructure, whether that means five servers or five hundred. This ease of scaling is accomplished by executing commands on target groups at the same time, making Ansible both efficient and scalable.

While the project is managed by RedHat, a diverse community of companies and individuals regularly contribute to the project. These contributions include new modules for variety of hosts and devices, bug fixes, and other core functionalities.

For more information, visit https://www.ansible.com.

1.3 NetApp E-Series Contribution

NetApp E-Series began contributing to the Ansible project in 2017, supporting the project by producing quality modules for their storage systems. Modules are Ansible's tools for ensuring the expected state or policies are enacted on managed systems. Ansible passes inventory information about the target to the modules, determines whether changes are required, and if so, makes the necessary changes. The modules produced by NetApp E-Series integrate well with existing infrastructure by making inquiries and changes through the NetApp SANtricity® Web Services Embedded that comes installed by default on all storage systems starting with the E2800 storage array. Older storage systems can also take advantage of this Ansible solution by installing SANtricity Web Services Proxy on an external host.

Customers can find the NetApp E-Series Ansible modules on Ansible's website with the prefix "netapp_e". These modules help our customers deploy new systems, setup host connections, provision storage, and other common management tasks. There is even an E-Series host role (more on roles later) that manages storage groups and volumes, hosts and host groups, interface ports, and storage-to-host mapping.

1.4 NetApp E-Series and Ansible Key Benefits

Why Choose the Ansible Orchestration Solution?

- Ansible is a simple, powerful, agentless tool that can be used to maintain complete infrastructures.
- Ansible's focus on the desired state helps significantly reduce complexity and downtime.
- Customers 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 there are no additional tools required.

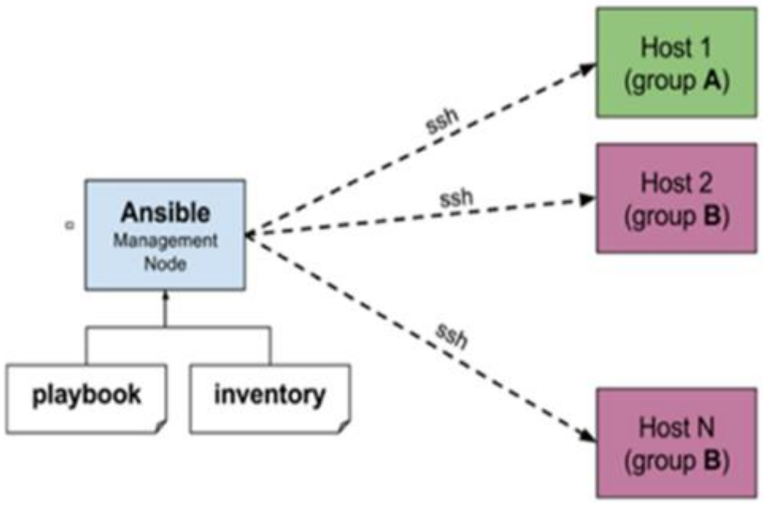
2 Solution Architecture

This section provides a helpful introduction into Ansible with E-Series, as well as links to more information.

2.1 Ansible Architecture Overview

Ansible uses information about your infrastructure in a collection of files referred to as an inventory. This inventory is used in conjunction with modules to execute tasks on a select host or group of hosts sometimes called target nodes. These tasks allow you to enforce a policy or expected state that the target should have. Ad-hoc commands allow 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 against target nodes. Figure 1 demonstrates this process of combining the playbook with inventory information so Ansible can apply the expected policies on each targeted system.

Figure 1) Ansible core architecture.



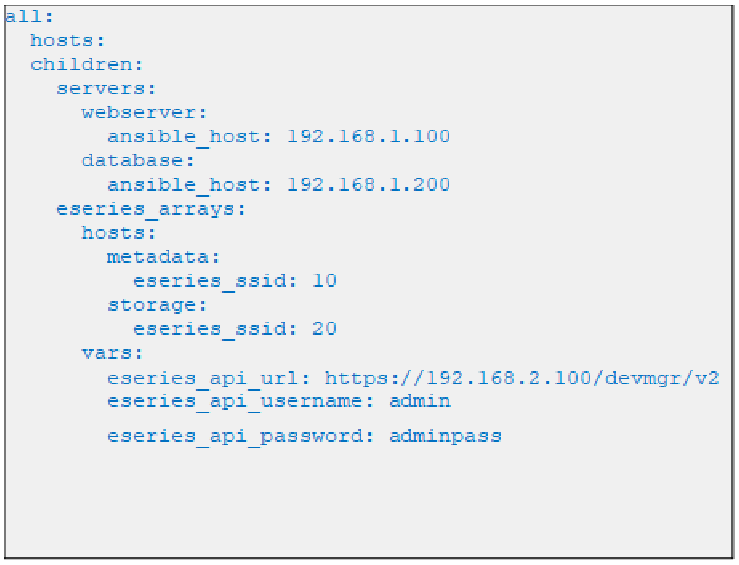
Unless you specify otherwise, all playbook tasks are executed on each target using the SSH protocol. Ansible has a wide range of connection plugins that can be used instead of SSH, such as WinRM, docker, and kubectl. Additionally, Ansible can execute tasks on multiple systems concurrently, reducing playbook execution time. For more information on connection plugin options, visit https://docs.ansible.com/ansible/latest/plugins/connection.html.

A helpful extension to a playbook is Ansible roles. A role is a collection of related tasks, variables, templates, files, plugins and custom modules that are grouped purposefully for a specific function. An example of this would be an apache webserver role that, when applied to a target, makes the host a functioning apache webserver according to details provided in the inventory.

2.2 Inventory

Inventories are used to define information about target hosts and host groups. These documents should be defined in YAML format for readability. Figure 2 gives an example of an inventory that defines hosts and how they are related.

Figure 2) Inventory file with variables.



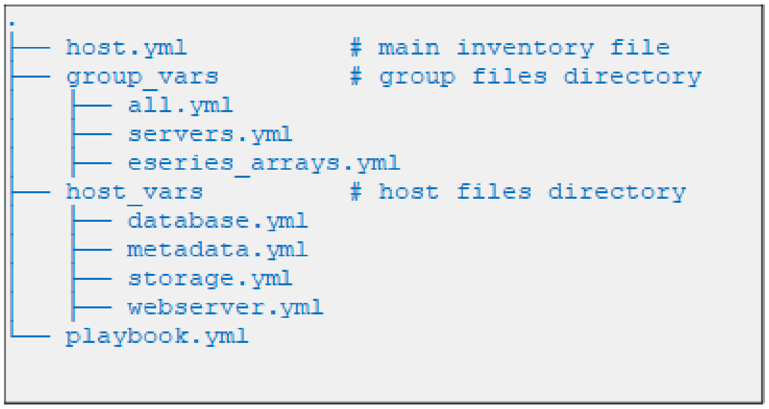
```
all:
  hosts:
  children:
    servers:
      webserver:
        ansible_host: 192.168.1.100
      database:
        ansible_host: 192.168.1.200
    eseries_arrays:
      hosts:
        metadata:
          eseries_ssid: 10
        storage:
          eseries_ssid: 20
      vars:
        eseries_api_url: https://192.168.2.100/devmgr/v2
        eseries_api_username: admin

        eseries_api_password: adminpass
```

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

While 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 only define hosts and groups and then place group and host variables into external files. Fortunately, Ansible makes applying this best practice very easy by creating two directories, `host_vars` and `group_vars` and placing in them YAML files that correspond to the name of the host or group. These directories are found in the same root directory as the inventory file. Figure 4 gives 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
```

In the example, the `group_vars'` files reflect each group name and host_vars' files reflect the hosts 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 is particularly helpful for viewing the complete inventory related to a specific host or group, given that common variables can be inherited from multiple parent groups and other Ansible sources. Variables in different locations have varying levels of priority. For instance, variables found in a host file take precedence over the same variables found in its group file. This is helpful for defining common values for your infrastructure.

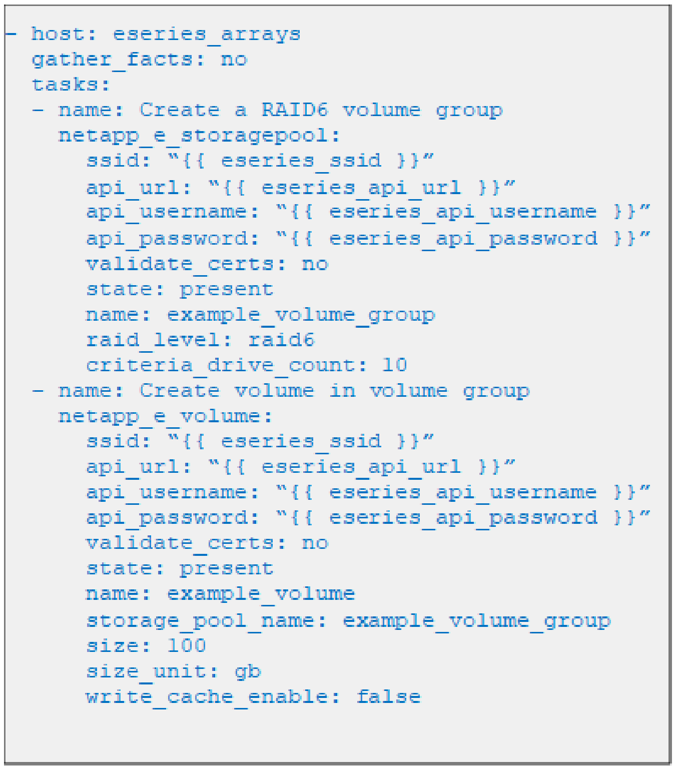
For more information, visit https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

2.3 Playbooks

Playbooks are Ansible's orchestration language written entirely in human-readable YAML. Each playbook starts by specifying the target host or group and then a series of tasks that defines the desired states or policies for each target. While playbooks can be incredibly simple when leveraging well-made modules, they can quickly become unnecessarily complex when designing the playbook with intertwined shell commands. When designing Ansible playbooks, simplicity should always be preferred since complexity will quickly become difficult to maintain (even for the original author). So, in general, avoid using the shell module when an existing module suffices. Doing so will help you avoid unnecessary complications and can simplify debugging when issues arise.

Figure 4 provides a simple playbook that targets the `eseries_arrays` 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 netapp_e_storagepool and netapp_e_volume modules to ensure that these policies are enforced. This means that if the write cache option has been enabled inadvertently, when the playbook is run again, the change will be recognized and Ansible will disable the option. Also, notice the structure, "{{… }}", found on many lines, this is how values are used from the inventory within tasks.

Figure 4) Ansible playbook with E-Series modules.



```
- host: eseries_arrays
  gather_facts: no
  tasks:
  - name: Create a RAID6 volume group
    netapp_e_storagepool:
      ssid: “{{ eseries_ssid }}”
      api_url: “{{ eseries_api_url }}”
      api_username: “{{ eseries_api_username }}”
      api_password: “{{ eseries_api_password }}”
      validate_certs: no
      state: present
      name: example_volume_group
      raid_level: raid6
      criteria_drive_count: 10
  - name: Create volume in volume group
    netapp_e_volume:
      ssid: “{{ eseries_ssid }}”
      api_url: “{{ eseries_api_url }}”
      api_username: “{{ eseries_api_username }}”
      api_password: “{{ eseries_api_password }}”
      validate_certs: no
      state: present
      name: example_volume
      storage_pool_name: example_volume_group
      size: 100
      size_unit: gb
      write_cache_enable: false
```

Ansible provides documentation for available modules through their website or 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 since the output provides cut-and-paste YAML snippet for your playbook, complete with comments describing each option.

For more information, visit https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html

2.4 Roles

Ansible roles are a collection of tasks, files, templates, variables, plugins and modules that are used together to perform a complex task. Roles are called from a playbook as demonstrated in Figure 5. In this playbook, a NetApp E-Series developed role 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_array` inventory as outlined by the role’s documentation. Figure 5 gives an example `eseries_storage_array` host inventory file that supports the role in creating policies such as number of drives in the storage pool, volume sizes, where to map the volumes and many others.

Figure 5) Ansible playbook with E-Series iSCSI role.

```
- hosts: eseries_storage_arrays
  gather_facts: false
  connection: local
  tasks:
  - name: Configure E-Series storage arrays
    include_role:
      name: netapp-eseries. netapp_eseries_host
    tags:
      - storage_pools
      - volumes
      - initiators
      - lun_mappings
      - ports
```

Figure 6) E-Series Storage Array host file example.

```
# Web Services Proxy credentials
eseries_ssid: 1
eseries_api_url: https://192.168.2.100:8443/devmgr/v2
eseries_api_username: admin
eseries_api_password: adminpass
eseries_validate_certs: no


# Storage HIC protocol
eseries_initiator_protocol: iscsi

# Controller port definitions
eseries_controller_port:
  controller_a:
    common_port_definitions:
      config_method: static
      gateway: 10.10.11.1
      subnet_mask: 255.255.255.0
    ports:
      - channel: 1
        address: 10.10.11.110
  controller_b:
    common_port_definitions:
      config_method: static
      gateway: 10.10.10.1
      subnet_mask: 255.255.255.0
    ports:
      - channel: 1
        address: 10.10.10.110

# Storage provisioning definitions
eseries_storage_pool_configuration:
  - name: beegfs_storage_vg
    raid_level: raid6
    criteria_drive_count: 12
    common_volume_configuration:
      workload_name: beegfs_storage
      metadata:
        type: storage
    volumes:
      - name: beegfs-storage-0[1-2]_[1-5]
        host: beegfs_storage
        size: 1000  # gb is the default size unit
```

Roles are also a great way for building complete policy collections, applying best practices and enforcing consistency in your inventory files.

For more information, visit https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

For more information about NetApp roles, visit https://github.com/NetApp/Ansible/roles/

2.5 Tags

The Ansible tags feature allows you to select which policies in playbooks and roles to check. Tags are included in the task definitions as shown in the example playbook found in Figure 5. These tags provide administrators the ability to selectively choose which policies to update or check. For example, to execute only the policies that pertain to E-Series storage array storage pools and volumes just `add --tags “storage_pools,volumes”` to your ansible-playbook command. Specific groups of tags can be skipped using the `--skip-tags` flag.

For more information, visit https://docs.ansible.com/ansible/latest/user_guide/playbooks_tags.html

3 Getting Started

Now that you have a foundation for Ansible with E-Series here is a quick start guide to using Ansible with our E-Series host role.

1. Install Ansible. For more information on install Ansible your specific Linux distribution, see https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html.
2. Retrieve the `netapp-eseries.host` role by running the `ansible-galaxy` command. This will download the role and place it within the ansible structure so that it can be used by your playbook.

```
$ ansible-galaxy install netapp-eseries.host
```

3. Create and move into a new Ansible project in home directory.
4. Create your project inventory files for your project. Below are examples for you to copy, paste, and update to reflect your inventory.

```
# hosts.yml
all:
  children:
    eseries_arrays:
      hosts:
        storage1:
        storage2:
    database_servers:
      hosts:
        storage_server:
        metadata_server:
```

```
# host_vars/storage1.yml
eseries_ssid: 1
eseries_api_url: https://192.168.1.100:8443/devmgr/v2
eseries_api_username: admin
eseries_api_password: adminpass
eseries_validate_certs: no

eseries_initiator_protocol: iscsi
eseries_controller_port_config_method: static
eseries_controller_port_subnet_mask: 255.255.255.0
eseries_controller_port:
  controller_a:
    ports:
      - channel: 1
        address: 10.10.11.110
  controller_b:
    ports:
      - channel: 1
        address: 10.10.10.110

eseries_storage_pool_configuration:
  - name: storage1_volume_group
    raid_level: raid6
    criteria_drive_count: 12
    volumes:
      - name: database_volume[1-3]
        host: storage_server
        size: 100  # Default unit size is gb
      - name: metadata_volume
        host: metadata_server
        size: 100
```

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

```
- hosts: eseries_arrays                    # Specifies the inventory target(s)
  connection: local                        # Forces commands to run on the local
                                           #   control node instead of over SSH
  gather_facts: false                      # Tells Ansible not to gather facts
                                           #   about inventory target(s)
  tasks:                                   # Start of tasks block.
    - name: Apply E-Series host role
      import_role:
        name: netapp-eseries.netapp_eseries_host
      tags:
        - storage_pools
        - volumes
        - initiators
        - lun_mappings
```

6. Execute your Ansible playbook.

4 Support

While NetApp E-Series produces quality Ansible modules and roles that each undergo specific testing to ensure they meet expectations, you can contact our dedicated Ansible engineers for help or to provide suggestions.

4.1 Modules

Any issues or feature requests that pertain to NetApp E-Series modules should be posted to Ansible's official GitHub repository, https://github.com/ansible/ansible/issues. We welcome you to partner with us by posting feature enhancement requests and any issues that you may encounter. The E-Series Ansible team receives notifications for any feedback that pertains to our modules, so you can count on our team to respond in a timely fashion.

4.2 Roles

You can go the NetApp GitHub Ansible repository, https://github.com/netapp/ansible/issues, to submit any issues, feature enhancements, or role ideas that you would like to see resolved or implemented.

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

4.3 NetApp Pub

Feel free to join the discussion on NetApp's official Slack #configurationmgmt channel for your Ansible questions. Go the site https://netapp.io/slack/ to interact directly with the NetApp community.

5 E-Series Ansible Modules

The following chart provides a comprehensive listing of the NetApp E-Series Ansible modules with a short description. For more complete module information please visit the Ansible storage modules site; see listing under NetApp (https://docs.ansible.com/ansible/latest/modules/list_of_storage_modules.html).

Table 1) NetApp E-Series modules list.

Module Name	Description
netapp_e_alerts	NetApp E-Series manage email notification settings
netapp_e_amg	NetApp E-Series create, remove, and update asynchronous mirror groups
netapp_e_amg_role	NetApp E-Series update storage array's role for Asynchronous Mirror Group (AMG).
netapp_e_amg_sync	NetApp E-Series conduct synchronization actions on asynchronous mirror groups.
netapp_e_asup	NetApp E-Series manage auto-support settings
netapp_e_auditlog	NetApp E-Series manage audit-log configuration
netapp_e_auth	NetApp E-Series set or update the password for a storage array.
netapp_e_facts	NetApp E-Series retrieve facts about NetApp E-Series storage arrays
netapp_e_flashcache	NetApp E-Series manage SSD caches

Module Name	Description
netapp_e_global	NetApp E-Series manage global settings configuration
netapp_e_host	NetApp E-Series manage E-Series hosts/host groups
netapp_e_hostgroup	NetApp E-Series manage array host groups
netapp_e_iscsi_interface	NetApp E-Series manage iSCSI interface configuration
netapp_e_iscsi_target	NetApp E-Series manage iSCSI target configuration
netapp_e_ldap	NetApp E-Series manage LDAP integration to use for authentication
netapp_e_lun_mapping	NetApp E-Series create, delete, or modify LUN mappings
netapp_e_mgmt_interface	NetApp E-Series management interface configuration
netapp_e_snapshot_group	NetApp E-Series manage snapshot groups
netapp_e_snapshot_images	NetApp E-Series create and delete snapshot images
netapp_e_snapshot_volume	NetApp E-Series manage snapshot volumes.
netapp_e_storage_system	NetApp E-Series Web Services Proxy manage storage arrays
netapp_e_storagepool	NetApp E-Series manage disk groups and disk pools
netapp_e_syslog	NetApp E-Series manage syslog settings
netapp_e_volume	NetApp E-Series manage storage volumes (standard and thin)
netapp_e_volume_copy	NetApp E-Series create volume copy pairs

Where to Find Additional Information

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

- Ansible Installation Guide – Linux distribution specific installation guide
  https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html
- NetApp Modules – Comprehensive list for all NetApp modules
  https://docs.ansible.com/ansible/latest/modules/list_of_storage_modules.html#netapp
- Ansible Module Index – Categorized lists of all currently available Ansible modules for your playbook
  https://docs.ansible.com/ansible/latest/modules/modules_by_category.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 practical best practices guide
  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 on 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 1.0	May 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.

TR-4574-0519