NetApp

# DOCUMENTATION

## MAF implementation for BPER Services ScpA

# *Contents:*

# 1. MAF

## MAF is a collection of Ansible roles, custom Ansible modules and filters.

Roles can be defined by function they serve:

- Manipulate with ONTAP resources
- Implement logic that allows decision making on ONTAP resources
- Manupulating with data
- Other applications comminucation
- Any other functionality that Ansible provides Data structure it uses is mimicing ONTAP REST API structure.

ONTAP related roles implement the following operations with objects:

- Create
- Modify parameters
- Delete
- Manupilate (Ex: vol move, vol efficiency start etc.)

Logic operations may include:

- Implement corner cases
- Prepare values based on other values

Custom Ansible modules allow to implement functionality unique to the environment it operates in. Custom modules allow to fine tune the behavior to the task requiremnts. Ex.: custom module for identifying best aggregate candidate for the new volume basing on multiple values and exceptions.

Custom filters allow data manipulation and are useful with naming convention, logging and data manupulation.

# 2. Automation task and purpose

Playbook name: bper_vol_qtree_create.yml

Requested operation:

1. Create a single volume with 1 qtree on primary Metrocluster (MCC) ONTAP system
2. Volume name must be unique across MCC environment
3. Qtree must be exported via NFS to the given network
4. Once primary volume is created - secondary volume must be created as SnapMirror destination for vaulting purposes
5. SnapMirror relationship must be established
6. Playbook must support integration with VMWare Aria solution (via remote ssh exec)

# 3. Requirements

### *Ansible:*

Ansible-core min. version: 2.11 Ansible min version: 8.7.0 Ansible Collection netapp.ontap min version: 22.11.0

### *Python:*

minimal version: 3.7

### *Python modules:*

netapp-ontap >= 9.13

### *Operating System:*

Linux, no distribution dependencies

### *NetApp ONTAP:*

Minimal tested version: 9.13.1 All tasks addressed to ONTAP preresume operations via REST API only. ONTAP ZAPI calls are excluded from all roles.

```
## NOTE: ZAPI is diabled on ONTAP from 9.15.1 release (but can be enabled
manually).
## is to be completely decommissioned starting ONTAP 9.17.1 release (as of
December 2024).
```

### *NetApp ONTAP SVM:*

SnapMirror:

- SVM must exist and be online
- SVM must have network interface
- Source primary SVM and vault secondary SVM must be peered
- ONTAP cluster must have snapshot policy pre-configured

### *Network:*

Ansible control station must have direct connection to port 443 on ONTAP cluster management interface

### *ONTAP user:*

User under which operations are performed on primary and secondary ONTAP clusters must have the following access rights: Applications - http Scope - cluster

User role must have write(all) access to the following REST API endpoints:
/api/protocols/nfs/export-policies all
/api/snapmirror all
/api/storage/qtrees all
/api/storage/volumes all

# 4. Execution environment

To execute the playbook all folders that are the part of MAF must reside in write accessible folder for the linux user that runs ansible-playbook command. There are 2 supported execution environments:

- Linux shell with extra variables passed in command line
- VMWare Aria launching remote shell via SSH with extra variables passed as input

Playbook is not using Ansible inventories due to:

- Ansible to ONTAP communication is performed over HTTPS, not default SSH
- Task execution is not targeting a group or all the clusters
- Variables design and SVM selection logic are not well suited for Ansible Inventory

All operations are performed against any ONTAP cluster are made to cluster management interface with cluster scope. No SVM scope operations are performed.

# 5. Ansible playbook operations

## 5.1 Roles

Roles that are the part of the solution:

- ontap/export_policy
  Purpose:
  - prepare role facts values
  - create export policy on primary ONTAP cluster
  - delete export policy in a rollback scenario
- ontap/volume
  Purpose:
  - prepare role facts values
  - create volume on primary or secondary ONTAP cluster and mount to a junction
  - configure volume extended parameters: efficiency, autosize, compression, etc
  - delete volume in a rollback scenario
- ontap/qtree
  Purpose:
  - prepare role facts values
  - create qtree in a given volume
  - delete qtree in a rollback scenario
- ontap/snapmirror
  Purpose:
  - prepare role facts values
  - create Snapmirror relationship
  - initialize Snapmirror relationship

## 5.2 Inventory

Inventory is designed to be loaded as variables of a spcific design.
Location: ./vars/inventrory_bper_prod.yml

Inventory is SVM based and contains variables for each Primary SVM. Inventory is a list of hosts:

```
hosts:
  - name: _PRI_SVM_NAME_
    cluster_mgmt: _SVM_CLUSTER_IP_
    cluster_name: _CLUSTER_NAME_
    vault_svm: _PRI_SVM_VAULT_PARTNER_SVM_NAME_
```

Example:

```
- name: svm_cse_01
  cluster_mgmt: 192.168.78.22
  cluster_name: fas8200-cse
  vault_svm: svm_cse_new_vault
```

Having vault_svm with the name of peer SVM allows to specify a pair Primary SVM <--> Vault SVM.

Secondary SVM records:

```
hosts:
    - name: svm_cse_new_vault
      cluster_mgmt: 192.168.65.207
      cluster_name: fas2820-moc
```

The selector code in playbook is finding the right primary SVM peer using vault_svm value for selection in Secondary SVM list.

Validation: Inventory structure and consistency is validated before main tasks execition.

## 5.3 Credentials

The solution is designed to use the same creadentials for both Primary and Secondary vault ONTAP clusters. User msut have identical access rights on all clusters in the inventory.

MAF supports both username/password pair and certificate authenticaton. The solution is configured to use username/password pair.

Validation: Credentials are validated before main tasks execition. If they do not satisfy the requirements - play exits.

## 5.4 Input values

The follwoing input values are supported:

**input_env (mandatory):**

- is defined
- is in
    - PR
    - PP
    - CT
    - SE
    - CE
    - SR
    - CR
    - SV
    - CO

**input_size (mandatory):**

- is defined
- is one of the following values (defined in GB):
    - 5
    - 10
    - 25
    - 50

**input_snaplock (mandatory):**

- is defined
- is in ['true', 'false']

**input_clientmatch (mandatory):**

- is defined
- is given in valid network notation (IP, network)

**input_proc (mandatory):**

- is defined
- is 5 characters long

**input_dryrun (optional):**

- is in ['true', 'false', 'yes', 'no'] if defined

Validation: input values are validated before main tasks execition. If they do not satisfy the requirements - play exits.

## 5.5 Data structure

The variables that required for the workflow design and execution have to be stored and represented in a structure that represents ONTAP REST, inculding:

- flat variables
- lists, lists of dictionaries
- dictionaries
- names and structures of those variables

## 5.6 Default values

Default values are devided into 2 categories: ONTAP instancies related and playbook control related.
Default values are loaded in the play as variables in vars section of the play.
Location: ./vars/default.yml

**Playbook control:**

- to reduce console output on large collections
  nolog: true
- set dryrun to true to see what parameters will be used for ONTAP instancies creation. No changes on any ONTAP clusters will be applied. Play will end after printing the variables.
  input_dryrun: false
- set precheck_inventory to false to skip inventory check (SVM check on ONTAP clusters)
  precheck_inventory: true
- playbook dummy folder, actual value for this variable will be taken from Ansible magic variables while in the play
  playbook_dir: '/root'

**ONTAP variables**

ONTAP variables are designed per object instance as a flat variable, dictionary, list or list of dictionaries. This solution includes default parameters specified in global variable vars_defaults.
Other parameters collected from ONTAP clusters and generated vales based on ONTAP facts in are added or modified in the play. See details in section 5.7.
Static major varsiables are:

- source: contains variables to be used when creating instancies on primary ONTAP cluster
- destination: contains variables to be used when creating instancies on secondary ONTAP cluster
- snapmirror: contains Snapmirror only variables to be used when creating Snapmirror relationship on secondary ONTAP cluster

Default values do not include values that are being generated.
Variables provided to the play has the following structure:

```
vars_defaults:
    source:                  # definition of source instancies
        volume:              # parameters for source volume
            type:           *default_value*
            language:       *default_value*
            volume_autosize:
                mode:       *default_value*
        etc...
        qtree:
            security_style: *default_value*
            oplocks:        *default_value*
    destination:             # definition of destination instancies
        volume:              # parameters for source volume
                type:       *default_value*
    snapmirror:              # definition of destination instancies
        policy:             *default_value*
```

## 5.7 Name generation and variable merge

Volume name is required to be generated basing on the following:

- must be unique across MCC clusters
- must follow naming convention (see separate document)
- new volume index digits (*****NN) must be incremented by 1
- qtree, export policy and destination vault volume name are inherited fro the first part of the volume name
- SVM selection for the new volume must consider volume count attached to the SVM

Variables values are being collected and generated in the following tasks:

- bper/facts/prepare_facts.yml
- bper/logic/01_preflight_setup.yml
- bper/logic/02_source_setup.yml
- bper/logic/03_vault_setup.yml

There are 2 global variables involved in the process of values generation:

1. Defualts (var/defaults.yml: vars_defaults)
2. Local vars (var/local.yml: vars_local)

vars_local is the global variable, values for what are being collected throughout roles execution above.
When local variables collection and generation is completed - they are having the same structure as vars_defaults but contain non-default values.
These values based on the play execution and valid only for this execution.

Every execution of any instance create or delete role preceeds variables merge procedure.

Variables collection and generation workflow:

1. Loading vars_local: first version of variables set is generated - based on input extra variables
2. Prepare facts role task: generates name for special case of FUS-n environment (NOTE: is currently disabled as this environment name is being passed to playbook already prepared)
3. Prepare facts role task: adds aggregate inclusion or exclusion for aggregate Snaplock functionality
4. 01_preflight setup role task: collects existing volume names from all ONTAP clusters in the inventory and sets new volume name incremented by 1
5. 01_preflight setup role task: using new volume name the following values are being generated:
    1. qtree name
    2. export policy name
    3. destination volume name
    4. snapmirror source values
    5. snapmirror destination values
6. 01_preflight setup role task: values are being added to vars_local variable
7. 02_source_setup role task: selects source SVM basing on volume count and retrieves avaliable aggregates for this SVM
8. 02_source_setup role task: values are being added to vars_local variable
9. 03_vailt_setup role task: identifying destination vault SVM, sets and adds variables for Snapmirror

Once vars_local variables are generated they are merged with vars_defauls before every create or delete operation.

## 5.8 Logging

Every operation is being logged in a separate file with the follwoing format: {qlogdir}/date_time{qlogname}. qlogdir and qlogname are defined in the playbook and can be modified as necessary.

Logfile contains values passed to the create/delete role for the future review.

## 5.9 Dryrun

Dryrun {true, false} is instructing the playbook to print extra debug information. If enabled the final global variables will be printed and playbook ends without creating any instance. input_dryrun is set to false by default in vars/defaults.yml - it can be overwritten by extra varsiable passed to the playbook on execution.

# 6. Play workflow

Every instance create operation includes rescue section for the case when create operation has failed. Rescue section implements rollback scenario.

## 6.1 Pre-flight checks and values setup

The playbook is built on fail-fast design - it tries to identify issues before any instance is created and exit as soon as possible.

Pre-flight includes:

- input variables validation and exit if they do not comply
- objects details collection from all primary ONTAP clusters

Values setup includes:

- values setup based on decision made on information retrieved from ONTAP clusters
- selection of SVM on primary ONTAP cluster by volume count criteria
- selection of SVM on secondary cluster by predefined mapping
- variables merge into vars_local global variable

## 6.2 Create export policy

1. Merges vars_local with vars_default variables
2. Creates export policy on primary SVM basing on merged variables

## 6.3 Create source volume

1. Merges vars_local with vars_default variables
2. Selects most suitable aggregate to place the volume
3. Creates source volume on primary ONTAP cluster with specified parameters in merged variables

## 6.4 Create qtree

1. Merges vars_local with vars_default variables
2. Creates qtree on source volume with specified parameters in merged variables

## 6.5 Create destination volume

1. Merges vars_local with vars_default variables
2. Selects most suitable aggregate to place the volume
3. Creates vault volume on seconary ONTAP cluster with specified parameters in merged variables

## 6.6 Create Snapmirror

1. Merges vars_local with vars_default variables
2. Creates Snapmirror relationship from primary SVM:volume to secondary SVM:volume with specified parameters in merged variables

## 6.7 Rollback

On every step of creating ONTAP instances create operation may fail for some reason. As per requirements already created instances must be deleted. Every next step includes 1 more instance that needs to be deleted in a spcific order. Each create task includes rescue section to delete all previously created instances.

## 7. Installation

Unzip tarball with code package.

```
>cd bper-ontap-maf-vXXXX
>pip install -r requirements-python.txt
>ansible-galaxy collection install -r requirements-ansible.yml -f
```

Playbook is ready.

## 8. Execution

```
Example:
>ansible-playbook bper_vol_qtree_create.yml -e "input_password=SECRET_PASS
input_username=USERNAME input_env=PR input_size=5 input_proc=ABCDE
input_clientmatch=0.0.0.0/0 input_snaplock=false"
```

The playbook can be triggered by any external automation orchestrator with given input extra variables.

## 9. Authors and contacts