 # DOCUMENTATION ## MAF implementation for BPER Services ScpA

Contents:

++

1. MAF

MAF is a collection of Ansible roles, modules, 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

Playbook name: bper_vol_qtree_create.yml

Task:

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

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

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

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

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:

- is defined
- is in

    - PR
    - PP
    - CT
    - SE
    - CE
    - SR
    - CR
    - SV
    - CO

input_size:

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

input_snaplock:

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

input_clientmatch:

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

input_proc:

- is defined
- is 5 characters long

input_dryrun:

- 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