Centre d'automatisation NetApp Console

NetApp Automation

NetApp
November 18, 2025

Sommaire

Centre d’automatisation NetApp Console

Présentation du hub d’automatisation de la NetApp Console

La plateforme d’automatisation NetApp Console est un ensemble de solutions d’automatisation mises à la disposition des clients, partenaires et employés de NetApp . Le hub d’automatisation possède plusieurs fonctionnalités et avantages.

Un seul emplacement pour vos besoins d’automatisation

Vous pouvez accéder au "Centre d’automatisation NetApp Console" via l’interface utilisateur Web de la console. Cela permet de centraliser les scripts, les playbooks et les modules nécessaires à l’amélioration de l’automatisation et du fonctionnement de vos produits et services NetApp .

Les solutions sont créées et testées par NetApp

Toutes les solutions d’automatisation et tous les scripts ont été créés et testés par NetApp. Chaque solution cible une demande ou un cas d’utilisation spécifique. La priorité est donnée à l’intégration avec les services de fichiers et de données NetApp.

Documentation

Chacune des solutions d’automatisation comprend une documentation associée pour vous aider à démarrer. Bien que les solutions soient accessibles via l’interface Web de la console, toute la documentation est disponible sur ce site. La documentation est organisée en fonction des produits NetApp et des services cloud.

Une base solide pour l’avenir

NetApp s’engage à aider ses clients à améliorer et à rationaliser l’automatisation de leurs centres de données et de leurs environnements cloud. Nous prévoyons de continuer à améliorer la plateforme d’automatisation Console afin de répondre aux exigences des clients, aux évolutions technologiques et à l’intégration continue des produits.

Votre avis nous intéresse

L’équipe d’automatisation du service d’expérience client (CXO) de NetApp voudrait vous entendre. Si vous avez des commentaires, des problèmes ou des demandes concernant des fonctionnalités, veuillez envoyer un e-mail à équipe d’automatisation CXO.

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP : passage au cloud

Vous pouvez utiliser cette solution d’automatisation pour provisionner Amazon FSx for NetApp ONTAP avec des volumes et un FlexCache associé.

La gestion Amazon FSx for NetApp ONTAP est également appelée **FSx pour ONTAP**.

À propos de cette solution

À un niveau élevé, le code d’automatisation fourni avec cette solution effectue les actions suivantes :

- Provisionnez un système de fichiers FSX pour ONTAP de destination
- Provisionnement des SVM (Storage Virtual machines) pour le système de fichiers

- Création d'une relation de peering de cluster entre les systèmes source et destination
- Création d'une relation de peering de SVM entre le système source et le système de destination pour FlexCache
- Vous pouvez également créer des volumes FlexVol à l'aide de FSX pour ONTAP
- Créez un volume FlexCache dans FSX pour ONTAP, la source pointant vers un stockage sur site

L'automatisation est basée sur Docker et Docker compose qui doivent être installés sur la machine virtuelle Linux comme décrit ci-dessous.

Avant de commencer

Pour terminer le provisionnement et la configuration, vous devez disposer des éléments suivants :

- Vous devez télécharger le "Amazon FSx for NetApp ONTAP : passage au cloud" solution d'automatisation via l'interface utilisateur Web de la NetApp Console . La solution est conditionnée sous forme de fichier `AWS_FSxN_BTC.zip`.
- Connectivité réseau entre les systèmes source et de destination.
- Une VM Linux présentant les caractéristiques suivantes :
  - Distribution Linux basée sur Debian
  - Déployé sur le même sous-ensemble VPC utilisé pour le provisionnement FSX pour ONTAP
- Compte AWS.

Étape 1 : installer et configurer Docker

Installez et configurez Docker sur une machine virtuelle Linux basée sur Debian.

Étapes

1. Préparez l'environnement.

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl gnupg-
agent software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
```

2. Installez Docker et vérifiez l'installation.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker --version
```

3. Ajoutez le groupe Linux requis avec un utilisateur associé.

   Vérifiez d'abord si le groupe **docker** existe dans votre système Linux. Si ce n'est pas le cas, créez le

groupe et ajoutez l'utilisateur. Par défaut, l'utilisateur de shell actuel est ajouté au groupe.

```
sudo groupadd docker
sudo usermod -aG docker $(whoami)
```

4. Activez les nouvelles définitions de groupe et d'utilisateur

   Si vous avez créé un nouveau groupe avec un utilisateur, vous devez activer les définitions. Pour ce faire, vous pouvez vous déconnecter de Linux puis vous reconnecter. Ou vous pouvez exécuter la commande suivante.

```
newgrp docker
```

Étape 2 : installez Docker compose

Installez Docker compose sur une machine virtuelle Linux basée sur Debian.

Étapes

1. Installez Docker compose.

```
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

2. Vérifiez que l'installation a réussi.

```
docker-compose --version
```

Étape 3 : préparez l'image Docker

Vous devez extraire et charger l'image Docker fournie avec la solution d'automatisation.

Étapes

1. Copiez le fichier de solution `AWS_FSxN_BTC.zip` sur la machine virtuelle sur laquelle le code d'automatisation sera exécuté.

```
scp -i ~/<private-key.pem> -r AWS_FSxN_BTC.zip user@<IP_ADDRESS_OF_VM>
```

   Le paramètre d'entrée `private-key.pem` correspond à votre fichier de clé privée utilisé pour l'authentification des serveurs virtuels AWS (instance EC2).

2. Accédez au dossier approprié avec le fichier de solution et décompressez le fichier.

```
unzip AWS_FSxN_BTC.zip
```

3. Accédez au nouveau dossier `AWS_FSxN_BTC` créé avec l'opération de décompression et répertoriez les fichiers. Vous devriez voir le fichier `aws_fsxn_flexcache_image_latest.tar.gz`.

```
ls -la
```

4. Chargez le fichier image Docker. Le chargement doit normalement se terminer en quelques secondes.

```
docker load -i aws_fsxn_flexcache_image_latest.tar.gz
```

5. Vérifiez que l'image Docker est chargée.

```
docker images
```

Vous devriez voir l'image Docker `aws_fsxn_flexcache_image` avec la balise `latest`.

```
   REPOSITORY              TAG     IMAGE ID      CREATED      SIZE
aws_fsxn_flexcahce_image  latest  ay98y7853769  2 weeks ago  1.19GB
```

Étape 4 : créez un fichier d'environnement pour les identifiants AWS

Vous devez créer un fichier de variables locales pour l'authentification à l'aide de la clé d'accès et de la clé secrète. Ajoutez ensuite le fichier au `.env` fichier.

Étapes

1. Créez le `awsauth.env` fichier à l'emplacement suivant :

   `path/to/env-file/awsauth.env`

2. Ajoutez le contenu suivant au fichier :

```
access_key=<>
secret_key=<>
```

   Le format **doit** doit être exactement comme indiqué ci-dessus sans espaces entre `key` et `value`.

3. Ajoutez le chemin d'accès absolu au fichier à `.env` l'aide de la `AWS_CREDS` variable. Par exemple :

   `AWS_CREDS=path/to/env-file/awsauth.env`

Étape 5 : créer un volume externe

Vous avez besoin d'un volume externe pour vous assurer que les fichiers d'état Terraform et les autres fichiers importants sont persistants. Ces fichiers doivent être disponibles pour Terraform afin d'exécuter le workflow et les déploiements.

Étapes

1. Créez un volume externe en dehors de Docker compose.

   Assurez-vous de mettre à jour le nom du volume (dernier paramètre) à la valeur appropriée avant d'exécuter la commande.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Ajoutez le chemin d'accès au volume externe au fichier d'environnement à `.env` l'aide de la commande :

   `PERSISTENT_VOL=path/to/external/volume:/volume_name`

   N'oubliez pas de conserver le contenu du fichier existant et le formatage des deux points. Par exemple :

   ```
   PERSISTENT_VOL=aws_fsxn_volume:/aws_fsxn_flexcache
   ```

   Vous pouvez à la place ajouter un partage NFS en tant que volume externe à l'aide d'une commande, par exemple :

   `PERSISTENT_VOL=nfs/mnt/document:/aws_fsx_flexcache`

3. Mettre à jour les variables Terraform.
   a. Naviguez jusqu'au dossier `aws_fsxn_variables`.
   b. Vérifiez que les deux fichiers suivants existent : `terraform.tfvars` et `variables.tf`.
   c. Mettez à jour les valeurs dans `terraform.tfvars` selon les besoins de votre environnement.

      Voir "Ressource Terraform : système_fichier_aws_fsx_ONTAP" pour plus d'informations.

Étape 6 : provisionner Amazon FSx for NetApp ONTAP et FlexCache

Vous pouvez provisionner Amazon FSx for NetApp ONTAP et FlexCache.

Étapes

1. Accédez à la racine du dossier (AWS_FSXN_BTC) et exécutez la commande de provisionnement.

   ```
   docker-compose -f docker-compose-provision.yml up
   ```

   Cette commande crée deux conteneurs. Le premier conteneur déploie FSX pour ONTAP et le second conteneur crée le peering de cluster, le peering de SVM, le volume de destination et FlexCache.

2. Surveiller le processus de provisionnement.

```
docker-compose -f docker-compose-provision.yml logs -f
```

Cette commande vous donne la sortie en temps réel, mais a été configurée pour capturer les journaux via le fichier `deployment.log`. Vous pouvez modifier le nom de ces fichiers journaux en modifiant le fichier et en `.env` mettant à jour les variables `DEPLOYMENT_LOGS`.

Étape 7 : Détruire Amazon FSx for NetApp ONTAP et FlexCache

Vous pouvez éventuellement supprimer et retirer Amazon FSx for NetApp ONTAP et FlexCache.

1. Définissez la variable `flexcache_operation` du `terraform.tfvars` fichier sur « détruire ».
2. Naviguez jusqu'au dossier racine (AWS_FSXN_BTC) et exécutez la commande suivante.

```
docker-compose -f docker-compose-destroy.yml up
```

Cette commande crée deux conteneurs. Le premier conteneur supprime FlexCache et le second conteneur supprime FSX pour ONTAP.

3. Surveiller le processus de provisionnement.

```
docker-compose -f docker-compose-destroy.yml logs -f
```

Amazon FSx for NetApp ONTAP - Reprise après sinistre

Vous pouvez utiliser cette solution d'automatisation pour effectuer une sauvegarde de reprise après sinistre d'un système source à l'aide d' Amazon FSx for NetApp ONTAP .

La gestion Amazon FSx for NetApp ONTAP est également appelée **FSx pour ONTAP**.

À propos de cette solution

À un niveau élevé, le code d'automatisation fourni avec cette solution effectue les actions suivantes :

- Provisionnez un système de fichiers FSX pour ONTAP de destination
- Provisionnement des SVM (Storage Virtual machines) pour le système de fichiers
- Création d'une relation de peering de cluster entre les systèmes source et destination
- Création d'une relation de peering de SVM entre le système source et le système de destination pour SnapMirror
- Créer des volumes de destination
- Création d'une relation SnapMirror entre les volumes source et de destination
- Lancez le transfert SnapMirror entre les volumes source et de destination

L'automatisation est basée sur Docker et Docker compose qui doivent être installés sur la machine virtuelle Linux comme décrit ci-dessous.

Avant de commencer

Pour terminer le provisionnement et la configuration, vous devez disposer des éléments suivants :

- Vous devez télécharger le "Amazon FSx for NetApp ONTAP - Reprise après sinistre" solution d'automatisation via l'interface utilisateur Web de la NetApp Console . La solution est conditionnée comme suit : `FSxN_DR.zip`. Ce fichier zip contient `AWS_FSxN_Bck_Prov.zip` fichier que vous utiliserez pour déployer la solution décrite dans ce document.
- Connectivité réseau entre les systèmes source et de destination.
- Une VM Linux présentant les caractéristiques suivantes :
  - Distribution Linux basée sur Debian
  - Déployé sur le même sous-ensemble VPC utilisé pour le provisionnement FSX pour ONTAP
- D'un compte AWS.

Étape 1 : installer et configurer Docker

Installez et configurez Docker sur une machine virtuelle Linux basée sur Debian.

Étapes

1. Préparez l'environnement.

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl gnupg-
agent softwareproperties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
```

2. Installez Docker et vérifiez l'installation.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker --version
```

3. Ajoutez le groupe Linux requis avec un utilisateur associé.

   Vérifiez d'abord si le groupe **docker** existe dans votre système Linux. S'il n'existe pas, créez le groupe et ajoutez l'utilisateur. Par défaut, l'utilisateur de shell actuel est ajouté au groupe.

```
sudo groupadd docker
sudo usermod -aG docker $(whoami)
```

4. Activez les nouvelles définitions de groupe et d'utilisateur

   Si vous avez créé un nouveau groupe avec un utilisateur, vous devez activer les définitions. Pour ce faire, vous pouvez vous déconnecter de Linux puis vous reconnecter. Ou vous pouvez exécuter la commande suivante.

   ```
   newgrp docker
   ```

Étape 2 : installez Docker compose

Installez Docker compose sur une machine virtuelle Linux basée sur Debian.

Étapes

1. Installez Docker compose.

   ```
   sudo curl -L
   "https://github.com/docker/compose/releases/latest/download/docker-
   compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
   sudo chmod +x /usr/local/bin/docker-compose
   ```

2. Vérifiez que l'installation a réussi.

   ```
   docker-compose --version
   ```

Étape 3 : préparez l'image Docker

Vous devez extraire et charger l'image Docker fournie avec la solution d'automatisation.

Étapes

1. Copiez le fichier de solution `AWS_FSxN_Bck_Prov.zip` sur la machine virtuelle sur laquelle le code d'automatisation sera exécuté.

   ```
   scp -i ~/<private-key.pem> -r AWS_FSxN_Bck_Prov.zip
   user@<IP_ADDRESS_OF_VM>
   ```

   Le paramètre d'entrée `private-key.pem` correspond à votre fichier de clé privée utilisé pour l'authentification des serveurs virtuels AWS (instance EC2).

2. Accédez au dossier approprié avec le fichier de solution et décompressez le fichier.

   ```
   unzip AWS_FSxN_Bck_Prov.zip
   ```

3. Accédez au nouveau dossier `AWS_FSxN_Bck_Prov` créé avec l'opération de décompression et

répertoriez les fichiers. Vous devriez voir le fichier `aws_fsxn_bck_image_latest.tar.gz`.

```
ls -la
```

4. Chargez le fichier image Docker. Le chargement doit normalement se terminer en quelques secondes.

```
docker load -i aws_fsxn_bck_image_latest.tar.gz
```

5. Vérifiez que l'image Docker est chargée.

```
docker images
```

Vous devriez voir l'image Docker `aws_fsxn_bck_image` avec la balise `latest`.

```
   REPOSITORY         TAG      IMAGE ID       CREATED       SIZE
aws_fsxn_bck_image   latest   da87d4974306   2 weeks ago   1.19GB
```

Étape 4 : créez un fichier d'environnement pour les identifiants AWS

Vous devez créer un fichier de variables locales pour l'authentification à l'aide de la clé d'accès et de la clé secrète. Ajoutez ensuite le fichier au `.env` fichier.

Étapes

1. Créez le `awsauth.env` fichier à l'emplacement suivant :

   `path/to/env-file/awsauth.env`

2. Ajoutez le contenu suivant au fichier :

```
access_key=<>
secret_key=<>
```

   Le format **doit** doit être exactement comme indiqué ci-dessus sans espaces entre `key` et `value`.

3. Ajoutez le chemin d'accès absolu au fichier à `.env` l'aide de la `AWS_CREDS` variable. Par exemple :

   `AWS_CREDS=path/to/env-file/awsauth.env`

Étape 5 : créer un volume externe

Vous avez besoin d'un volume externe pour vous assurer que les fichiers d'état Terraform et les autres fichiers importants sont persistants. Ces fichiers doivent être disponibles pour Terraform afin d'exécuter le workflow et les déploiements.

Étapes

1. Créez un volume externe en dehors de Docker compose.

   Assurez-vous de mettre à jour le nom du volume (dernier paramètre) à la valeur appropriée avant d'exécuter la commande.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Ajoutez le chemin d'accès au volume externe au fichier d'environnement à `.env` l'aide de la commande :

   `PERSISTENT_VOL=path/to/external/volume:/volume_name`

   N'oubliez pas de conserver le contenu du fichier existant et le formatage des deux points. Par exemple :

   ```
   PERSISTENT_VOL=aws_fsxn_volume:/aws_fsxn_bck
   ```

   Vous pouvez à la place ajouter un partage NFS en tant que volume externe à l'aide d'une commande, par exemple :

   `PERSISTENT_VOL=nfs/mnt/document:/aws_fsx_bck`

3. Mettre à jour les variables Terraform.
   a. Naviguez jusqu'au dossier `aws_fsxn_variables`.
   b. Vérifiez que les deux fichiers suivants existent : `terraform.tfvars` et `variables.tf`.
   c. Mettez à jour les valeurs dans `terraform.tfvars` selon les besoins de votre environnement.

      Voir "Ressource Terraform : système_fichier_aws_fsx_ONTAP" pour plus d'informations.

Étape 6 : déployer la solution de sauvegarde

Vous pouvez déployer et provisionner la solution de sauvegarde de reprise sur incident.

Étapes

1. Naviguez jusqu'au dossier racine (AWS_FSxN_BCK_Prov) et exécutez la commande de provisionnement.

   ```
   docker-compose up -d
   ```

   Cette commande crée trois conteneurs. Le premier conteneur déploie FSX pour ONTAP. Le second conteneur crée le peering de cluster, le peering de SVM et le volume de destination. Le troisième conteneur crée la relation SnapMirror et lance le transfert SnapMirror.

2. Surveiller le processus de provisionnement.

   ```
   docker-compose logs -f
   ```

Cette commande vous donne la sortie en temps réel, mais a été configurée pour capturer les journaux via le fichier `deployment.log`. Vous pouvez modifier le nom de ces fichiers journaux en modifiant le fichier et en `.env` mettant à jour les variables `DEPLOYMENT_LOGS`.

Azure NetApp Files

Installez Oracle à l'aide de Azure NetApp Files

Vous pouvez utiliser cette solution d'automatisation pour provisionner les volumes Azure NetApp Files et installer Oracle sur une machine virtuelle disponible. Oracle utilise ensuite les volumes pour le stockage des données.

À propos de cette solution

À un niveau élevé, le code d'automatisation fourni avec cette solution effectue les actions suivantes :

- Configuration d'un compte NetApp sur Azure
- Configurez un pool de capacité de stockage sur Azure
- Provisionnez les volumes Azure NetApp Files en fonction de la définition
- Créer les points de montage
- Montez les volumes Azure NetApp Files sur les points de montage
- Installez Oracle sur le serveur Linux
- Créez les auditeurs et la base de données
- Créer les bases de données enfichables (PDB)
- Démarrez l'écouteur et l'instance Oracle
- Installez et configurez `azacsnap` l'utilitaire pour prendre un instantané

Avant de commencer

Vous devez disposer des éléments suivants pour terminer l'installation :

- Vous devez télécharger le "Oracle avec Azure NetApp Files" solution d'automatisation via l'interface utilisateur Web de la NetApp Console . La solution est conditionnée sous forme de fichier `na_oracle19c_deploy-master.zip`.
- Une VM Linux présentant les caractéristiques suivantes :
  - RHEL 8 (Standard_D8s_v3-RHEL-8)
  - Déployé sur le même réseau virtuel Azure que celui utilisé pour le provisionnement Azure NetApp Files
- Un compte Azure

La solution d'automatisation est fournie sous forme d'image et exécutée à l'aide de Docker et Docker compose. Vous devez installer les deux sur la machine virtuelle Linux comme décrit ci-dessous.

Vous devez également enregistrer la machine virtuelle avec RedHat à l'aide de la commande `sudo subscription-manager register`. La commande vous invite à saisir les informations d'identification de votre compte. Si nécessaire, vous pouvez créer un compte chez https://developers.redhat.com/.

Étape 1 : installer et configurer Docker

Installez et configurez Docker sur une machine virtuelle RHEL 8 Linux.

Étapes

1. Installez le logiciel Docker à l'aide des commandes suivantes.

```
dnf config-manager --add
-repo=https://download.docker.com/linux/centos/docker-ce.repo
dnf install docker-ce --nobest -y
```

2. Démarrez Docker et affichez la version pour confirmer que l'installation a réussi.

```
systemctl start docker
systemctl enable docker
docker --version
```

3. Ajoutez le groupe Linux requis avec un utilisateur associé.

   Vérifiez d'abord si le groupe **docker** existe dans votre système Linux. Si ce n'est pas le cas, créez le groupe et ajoutez l'utilisateur. Par défaut, l'utilisateur de shell actuel est ajouté au groupe.

```
sudo groupadd docker
sudo usermod -aG docker $USER
```

4. Activez les nouvelles définitions de groupe et d'utilisateur

   Si vous avez créé un nouveau groupe avec un utilisateur, vous devez activer les définitions. Pour ce faire, vous pouvez vous déconnecter de Linux puis vous reconnecter. Ou vous pouvez exécuter la commande suivante.

```
newgrp docker
```

Étape 2 : installez Docker compose et les utilitaires NFS

Installez et configurez Docker compose avec le package des utilitaires NFS.

Étapes

1. Installez Docker compose et affichez la version pour confirmer que l'installation a réussi.

```
dnf install curl -y
curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

2. Installez le package des utilitaires NFS.

```
sudo yum install nfs-utils
```

Étape 3 : téléchargez les fichiers d'installation d'Oracle

Téléchargez les fichiers d'installation et de correctifs Oracle requis, ainsi que `azacsnap` l'utilitaire.

Étapes

1. Connectez-vous à votre compte Oracle si nécessaire.
2. Téléchargez les fichiers suivants.

Fichier	Description
LINUX.X64_193000_db_home.zip	19.3 outil de pose de base
p31281355_190000_Linux-x86-64.zip	Patch 19.8 RU
p6880880_190000_Linux-x86-64.zip	version 12.2.0.1.23 d'opatch
azacsnap_installer_v5.0.run	programme d'installation d'azacsnap

3. Placez tous les fichiers d'installation dans le dossier `/tmp/archive`.
4. Assurez-vous que tous les utilisateurs du serveur de base de données disposent d'un accès complet (lecture, écriture, exécution) au dossier `/tmp/archive`.

Étape 4 : préparez l'image Docker

Vous devez extraire et charger l'image Docker fournie avec la solution d'automatisation.

Étapes

1. Copiez le fichier de solution `na_oracle19c_deploy-master.zip` sur la machine virtuelle sur laquelle le code d'automatisation sera exécuté.

```
scp -i ~/<private-key.pem> -r na_oracle19c_deploy-master.zip
user@<IP_ADDRESS_OF_VM>
```

Le paramètre d'entrée `private-key.pem` correspond à votre fichier de clé privée utilisé pour l'authentification d'une machine virtuelle Azure.

2. Accédez au dossier approprié avec le fichier de solution et décompressez le fichier.

```
unzip na_oracle19c_deploy-master.zip
```

3. Accédez au nouveau dossier `na_oracle19c_deploy-master` créé avec l'opération de décompression et répertoriez les fichiers. Vous devriez voir le fichier `ora_anf_bck_image.tar`.

```
ls -lt
```

4. Chargez le fichier image Docker. Le chargement doit normalement se terminer en quelques secondes.

```
docker load -i ora_anf_bck_image.tar
```

5. Vérifiez que l'image Docker est chargée.

```
docker images
```

Vous devriez voir l'image Docker `ora_anf_bck_image` avec la balise `latest`.

```
   REPOSITORY          TAG       IMAGE ID      CREATED      SIZE
ora_anf_bck_image    latest   ay98y7853769   1 week ago   2.58GB
```

Étape 5 : créer un volume externe

Vous avez besoin d'un volume externe pour vous assurer que les fichiers d'état Terraform et les autres fichiers importants sont persistants. Ces fichiers doivent être disponibles pour Terraform afin d'exécuter le workflow et les déploiements.

Étapes

1. Créez un volume externe en dehors de Docker compose.

   Assurez-vous de mettre à jour le nom du volume avant d'exécuter la commande.

```
docker volume create <VOLUME_NAME>
```

2. Ajoutez le chemin d'accès au volume externe au fichier d'environnement à `.env` l'aide de la commande :

   `PERSISTENT_VOL=path/to/external/volume:/ora_anf_prov`.

   N'oubliez pas de conserver le contenu du fichier existant et le formatage des deux points. Par exemple :

```
PERSISTENT_VOL= ora_anf _volume:/ora_anf_prov
```

3. Mettre à jour les variables Terraform.
   a. Naviguez jusqu'au dossier `ora_anf_variables`.
   b. Vérifiez que les deux fichiers suivants existent : `terraform.tfvars` et `variables.tf`.
   c. Mettez à jour les valeurs dans `terraform.tfvars` selon les besoins de votre environnement.

Étape 6 : installez Oracle

Vous pouvez désormais provisionner et installer Oracle.

Étapes

1. Installez Oracle à l'aide de la séquence de commandes suivante.

```
docker-compose up terraform_ora_anf
bash /ora_anf_variables/setup.sh
docker-compose up linux_config
bash /ora_anf_variables/permissions.sh
docker-compose up oracle_install
```

2. Rechargez vos variables Bash et confirmez en affichant la valeur de `ORACLE_HOME`.
   a. `cd /home/oracle`
   b. `source .bash_profile`
   c. `echo $ORACLE_HOME`
3. Vous devriez pouvoir vous connecter à Oracle.

```
sudo su oracle
```

Étape 7 : validation de l'installation d'Oracle

Vous devez confirmer que l'installation d'Oracle a réussi.

Étapes

1. Connectez-vous au serveur Oracle Linux et affichez la liste des processus Oracle. Cela confirme que l'installation est terminée comme prévu et que la base de données Oracle est en cours d'exécution.

```
ps -ef | grep ora
```

2. Connectez-vous à la base de données pour examiner la configuration de la base de données et confirmer que les PDB ont été créés correctement.

```
sqlplus / as sysdba
```

Vous devez voir les résultats similaires à ce qui suit :

```
SQL*Plus: Release 19.0.0.0.0 - Production on Thu May 6 12:52:51 2021
Version 19.8.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.8.0.0.0
```

3. Exécutez quelques commandes SQL simples pour confirmer que la base de données est disponible.

```
select name, log_mode from v$database;
show pdbs.
```

Étape 8 : installez l'utilitaire azacsnap et effectuez une sauvegarde de snapshot

Vous devez installer et exécuter `azacsnap` l'utilitaire pour effectuer une sauvegarde d'instantané.

Étapes

1. Installez le conteneur.

```
docker-compose up azacsnap_install
```

2. Passez au compte d'utilisateur de snapshot.

```
su - azacsnap
execute /tmp/archive/ora_wallet.sh
```

3. Configurer un fichier de détails de sauvegarde de stockage. Cela va créer le `azacsnap.json` fichier de configuration.

```
cd /home/azacsnap/bin/
azacsnap -c configure --configuration new
```

4. Effectuer une sauvegarde de snapshot.

```
azacsnap -c backup --other data --prefix ora_test --retention=1
```

Étape 9 : migrer éventuellement un PDB sur site vers le cloud

Vous pouvez éventuellement migrer le boîtier de distribution électrique sur site vers le cloud.

Étapes

1. Définissez les variables dans les `tfvars` fichiers en fonction des besoins de votre environnement.
2. Migrer le PDB.

```
docker-compose -f docker-compose-relocate.yml up
```

Cloud Volumes ONTAP pour AWS

Cloud Volumes ONTAP pour AWS : en rafale vers le cloud

Cet article traite de la solution d'automatisation NetApp Cloud Volumes ONTAP pour AWS, disponible pour les clients NetApp depuis le hub d'automatisation de la NetApp Console .

La solution d'automatisation Cloud Volumes ONTAP pour AWS automatise le déploiement conteneurisé d'Cloud Volumes ONTAP pour AWS à l'aide de Terraform, ce qui vous permet de déployer rapidement Cloud Volumes ONTAP pour AWS sans aucune intervention manuelle.

Avant de commencer

- Vous devez télécharger le "Cloud Volumes ONTAP AWS : en rafale vers le cloud" Solution d'automatisation via l'interface utilisateur Web de la console. La solution est conditionnée comme suit : `cvo_aws_flexcache.zip`.
- Vous devez installer une machine virtuelle Linux sur le même réseau que Cloud Volumes ONTAP.
- Après avoir installé la machine virtuelle Linux, vous devez suivre les étapes de cette solution pour installer les dépendances requises.

Étape 1 : installez Docker et Docker compose

Installez Docker

Les étapes suivantes utilisent le logiciel de distribution Ubuntu 20.04 Debian Linux comme exemple. Les commandes que vous exécutez dépendent du logiciel de distribution Linux que vous utilisez. Reportez-vous à la documentation spécifique du logiciel de distribution Linux pour votre configuration.

Étapes

1. Installez Docker en exécutant les commandes suivantes `sudo` :

```
sudo apt-get update
sudo apt-get install apt-transport-https cacertificates curl gnupg-agent
software-properties-common curl -fsSL
https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -
sudo add-apt-repository “deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable”
sudo apt-get update
sudo apt-get install dockerce docker-ce-cli containerd.io
```

2. Vérifiez l’installation :

```
docker –version
```

3. Vérifiez qu’un groupe nommé « docker » a été créé sur votre système Linux. Si nécessaire, créez le groupe :

```
sudo groupadd docker
```

4. Ajoutez l’utilisateur qui doit accéder à Docker au groupe :

```
sudo usermod -aG docker $(whoami)
```

5. Vos modifications sont appliquées une fois que vous vous êtes déconnecter et que vous vous êtes de nouveau connecté au terminal. Vous pouvez également appliquer les modifications immédiatement :

```
newgrp docker
```

Installez Docker compose

Étapes

1. Installez Docker compose en exécutant les commandes suivantes `sudo` :

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose
```

2. Vérifiez l’installation :

```
docker-compose –version
```

Étape 2 : préparez l'image Docker

Étapes

1. Copiez le `cvo_aws_flexcache.zip` dossier sur la machine virtuelle Linux que vous souhaitez utiliser pour déployer Cloud Volumes ONTAP :

```
scp -i ~/<private-key>.pem -r cvo_aws_flexcache.zip
<awsuser>@<IP_ADDRESS_OF_VM>:<LOCATION_TO_BE_COPIED>
```

   - `private-key.pem` est votre fichier de clé privée pour la connexion sans mot de passe.
   - `awsuser` Est le nom d'utilisateur de la machine virtuelle.
   - `IP_ADDRESS_OF_VM` Est l'adresse IP de la machine virtuelle.
   - `LOCATION_TO_BE_COPIED` est l'emplacement où le dossier sera copié.

2. Extraire le `cvo_aws_flexcache.zip` dossier. Vous pouvez extraire le dossier dans le répertoire actuel ou dans un emplacement personnalisé.

   Pour extraire le dossier dans le répertoire actuel, exécutez :

```
unzip cvo_aws_flexcache.zip
```

   Pour extraire le dossier dans un emplacement personnalisé, exécutez :

```
unzip cvo_aws_flexcache.zip -d ~/<your_folder_name>
```

3. Une fois le contenu extrait, accédez au `CVO_Aws_Deployment` dossier et exécutez la commande suivante pour afficher les fichiers :

```
 ls -la
```

   Vous devriez voir une liste de fichiers, similaire à l'exemple suivant :

```
total 32
    drwxr-xr-x   8 user1  staff   256 Mar 23 12:26 .
    drwxr-xr-x   6 user1  staff   192 Mar 22 08:04 ..
    -rw-r--r--   1 user1  staff   324 Apr 12 21:37 .env
    -rw-r--r--   1 user1  staff  1449 Mar 23 13:19 Dockerfile
    drwxr-xr-x  15 user1  staff   480 Mar 23 13:19 cvo_Aws_source_code
    drwxr-xr-x   4 user1  staff   128 Apr 27 13:43 cvo_Aws_variables
    -rw-r--r--   1 user1  staff   996 Mar 24 04:06 docker-compose-
deploy.yml
    -rw-r--r--   1 user1  staff  1041 Mar 24 04:06 docker-compose-
destroy.yml
```

4. Localisez le `cvo_aws_flexcache_ubuntu_image.tar` fichier. Ce document contient l'image Docker requise pour déployer Cloud Volumes ONTAP pour AWS.
5. Décompressez le fichier :

```
docker load -i cvo_aws_flexcache_ubuntu_image.tar
```

6. Attendez quelques minutes que l'image Docker se charge, puis vérifiez que l'image Docker a bien été chargée :

```
docker images
```

Vous devez voir une image Docker nommée `cvo_aws_flexcache_ubuntu_image` avec la `latest` balise, comme dans l'exemple suivant :

```
REPOSITORY                                TAG        IMAGE ID       CREATED
SIZE
 cvo_aws_flexcache_ubuntu_image     latest    18db15a4d59c   2 weeks ago
1.14GB
```

Vous pouvez modifier le nom de l'image Docker si nécessaire. Si vous modifiez le nom de l'image Docker, veillez à mettre à jour le nom de l'image Docker dans les `docker-compose-deploy` fichiers et `docker-compose-destroy`.

Étape 3 : création de fichiers de variables d'environnement

À ce stade, vous devez créer deux fichiers de variables d'environnement. Un fichier est destiné à l'authentification des API AWS Resource Manager à l'aide des clés d'accès et secrètes AWS. Le deuxième fichier permet de définir des variables d'environnement pour permettre aux modules de la console Terraform de localiser et d'authentifier les API AWS.

Étapes

1. Créez le `awsauth.env` fichier à l'emplacement suivant :

   `path/to/env-file/awsauth.env`

   a. Ajoutez le contenu suivant au `awsauth.env` fichier :

      access_key=<> secret_key=<>

   Le format **doit** doit être exactement comme indiqué ci-dessus.

2. Ajoutez le chemin d'accès absolu au `.env` fichier.

   Entrez le chemin absolu du `awsauth.env` fichier d'environnement correspondant à la variable d'environnement `AWS_CREDS`.

   `AWS_CREDS=path/to/env-file/awsauth.env`

3. Accédez au `cvo_aws_variable` dossier et mettez à jour la clé d'accès et la clé secrète dans le fichier d'informations d'identification.

   Ajoutez le contenu suivant au fichier :

   aws_access_key_id=<> aws_secret_access_key=<>

   Le format **doit** doit être exactement comme indiqué ci-dessus.

Étape 4 : Inscrivez-vous aux services intelligents NetApp

Inscrivez-vous aux services intelligents NetApp via votre fournisseur de cloud pour payer à l'heure (PAYGO) ou via un contrat annuel. Les services intelligents NetApp incluent NetApp Backup and Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience et NetApp Disaster Recovery. La classification des données NetApp est incluse dans votre abonnement sans frais supplémentaires.

Étapes

1. Depuis le portail Amazon Web Services (AWS), accédez à **SaaS** et sélectionnez *S'abonner aux services intelligents NetApp *.

   Vous pouvez utiliser le même groupe de ressources que Cloud Volumes ONTAP ou un autre groupe de ressources.

2. Configurez le portail de la console NetApp pour importer l'abonnement SaaS dans la console.

   Vous pouvez le configurer directement à partir du portail AWS.

   Vous êtes redirigé vers le portail de la console pour confirmer la configuration.

3. Confirmez la configuration dans le portail de la console en sélectionnant **Enregistrer**.

Étape 5 : créer un volume externe

Vous devez créer un volume externe pour conserver les fichiers d'état Terraform et d'autres fichiers importants persistants. Vous devez vous assurer que les fichiers sont disponibles pour Terraform pour exécuter le workflow et les déploiements.

Étapes

1. Créer un volume externe en dehors de Docker compose :

```
docker volume create <volume_name>
```

Exemple :

```
docker volume create cvo_aws_volume_dst
```

2. Utilisez l'une des options suivantes :

   a. Ajoutez un chemin de volume externe au `.env` fichier d'environnement.

      Vous devez suivre le format exact indiqué ci-dessous.

      Format :

      `PERSISTENT_VOL=path/to/external/volume:/cvo_aws`

      Exemple :
      `PERSISTENT_VOL=cvo_aws_volume_dst:/cvo_aws`

   b. Ajoutez des partages NFS comme volume externe.

      Assurez-vous que le conteneur Docker peut communiquer avec les partages NFS et que les autorisations appropriées, telles que lecture/écriture, sont configurées.

      i. Ajoutez le chemin des partages NFS comme chemin d'accès au volume externe dans le fichier Docker compose, comme illustré ci-dessous : format :

      `PERSISTENT_VOL=path/to/nfs/volume:/cvo_aws`

Exemple :
`PERSISTENT_VOL=nfs/mnt/document:/cvo_aws`

3. Accédez au `cvo_aws_variables` dossier.

   Le fichier de variable suivant doit apparaître dans le dossier :

   - `terraform.tfvars`
   - `variables.tf`

4. Modifiez les valeurs à l'intérieur du `terraform.tfvars` fichier en fonction de vos besoins.

   Vous devez lire la documentation spécifique lors de la modification de l'une des valeurs de variable du `terraform.tfvars` fichier. Ces valeurs peuvent varier en fonction de la région, des zones de disponibilité et d'autres facteurs pris en charge par Cloud Volumes ONTAP pour AWS. Notamment les licences, la taille des disques et la taille des machines virtuelles pour les nœuds uniques et les paires haute disponibilité.

Toutes les variables de support pour l'agent de console et les modules Cloud Volumes ONTAP Terraform sont déjà définies dans le `variables.tf` déposer. Vous devez faire référence aux noms de variables dans le `variables.tf` fichier avant de l'ajouter au `terraform.tfvars` déposer.

5. Selon vos besoins, vous pouvez activer ou désactiver FlexCache et FlexClone en définissant les options suivantes sur `true` ou `false`.

   Les exemples suivants activent FlexCache et FlexClone :

   - `is_flexcache_required = true`
   - `is_flexclone_required = true`

Étape 6 : déploiement de Cloud Volumes ONTAP pour AWS

Procédez comme suit pour déployer Cloud Volumes ONTAP pour AWS.

Étapes

1. Depuis le dossier racine, exécutez la commande suivante pour déclencher le déploiement :

   ```
   docker-compose -f docker-compose-deploy.yml up -d
   ```

   Deux conteneurs sont déclenchés, le premier conteneur déploie Cloud Volumes ONTAP et le second envoie des données de télémétrie à AutoSupport.

   Le deuxième conteneur attend jusqu'à ce que le premier conteneur termine toutes les étapes avec succès.

2. Surveiller la progression du processus de déploiement à l'aide des fichiers journaux :

   ```
   docker-compose -f docker-compose-deploy.yml logs -f
   ```

   Cette commande fournit des résultats en temps réel et capture les données dans les fichiers journaux suivants :
   `deployment.log`

   `telemetry_asup.log`

   Vous pouvez modifier le nom de ces fichiers journaux en modifiant le `.env` fichier à l'aide des variables d'environnement suivantes :

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   Les exemples suivants montrent comment modifier les noms des fichiers journaux :

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Une fois que vous avez terminé

Vous pouvez utiliser les étapes suivantes pour supprimer l'environnement temporaire et nettoyer les éléments créés pendant le processus de déploiement.

Étapes

1. Si vous avez déployé FlexCache, définissez l'option suivante dans le `terraform.tfvars` fichier de variables, cela nettoie les volumes FlexCache et supprime l'environnement temporaire créé précédemment.

   ```
   flexcache_operation = "destroy"
   ```

   Les options possibles sont `deploy` et `destroy`.

2. Si vous avez déployé FlexClone, définissez l'option suivante dans le `terraform.tfvars` fichier de variables, cela nettoie les volumes FlexClone et supprime l'environnement temporaire créé précédemment.

   ```
   flexclone_operation = "destroy"
   ```

   Les options possibles sont `deploy` et `destroy`.

Cloud Volumes ONTAP pour Azure

Cloud Volumes ONTAP pour Azure : en rafale vers le cloud

Cet article traite de la solution d'automatisation NetApp Cloud Volumes ONTAP pour Azure, disponible pour les clients NetApp depuis le hub d'automatisation de la NetApp Console .

La solution d'automatisation Cloud Volumes ONTAP pour Azure automatise le déploiement conteneurisé d'Cloud Volumes ONTAP pour Azure à l'aide de Terraform, ce qui vous permet de déployer rapidement Cloud Volumes ONTAP pour Azure sans aucune intervention manuelle.

Avant de commencer

- Vous devez télécharger le "Cloud Volumes ONTAP Azure : en rafale vers le cloud" Solution d'automatisation via l'interface utilisateur Web de la console. La solution est conditionnée comme suit : `CVO-Azure-Burst-To-Cloud.zip`.
- Vous devez installer une machine virtuelle Linux sur le même réseau que Cloud Volumes ONTAP.
- Après avoir installé la machine virtuelle Linux, vous devez suivre les étapes de cette solution pour installer les dépendances requises.

Étape 1 : installez Docker et Docker compose

Installez Docker

Les étapes suivantes utilisent le logiciel de distribution Ubuntu 20.04 Debian Linux comme exemple. Les commandes que vous exécutez dépendent du logiciel de distribution Linux que vous utilisez. Reportez-vous à la documentation spécifique du logiciel de distribution Linux pour votre configuration.

Étapes

1. Installez Docker en exécutant les commandes suivantes `sudo` :

```
sudo apt-get update
sudo apt-get install apt-transport-https cacertificates curl gnupg-agent
software-properties-common curl -fsSL
https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -
sudo add-apt-repository “deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable”
sudo apt-get update
sudo apt-get install dockerce docker-ce-cli containerd.io
```

2. Vérifiez l’installation :

```
docker –version
```

3. Vérifiez qu’un groupe nommé « docker » a été créé sur votre système Linux. Si nécessaire, créez le groupe :

```
sudo groupadd docker
```

4. Ajoutez l’utilisateur qui doit accéder à Docker au groupe :

```
sudo usermod -aG docker $(whoami)
```

5. Vos modifications sont appliquées une fois que vous vous êtes déconnecter et que vous vous êtes de nouveau connecté au terminal. Vous pouvez également appliquer les modifications immédiatement :

```
newgrp docker
```

Installez Docker compose

Étapes

1. Installez Docker compose en exécutant les commandes suivantes `sudo` :

```
sudo curl -L
“https://github.com/docker/compose/releases/download/1.29.2/dockercompos
e-(□□□□□ - □)-(uname -m)” -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

2. Vérifiez l’installation :

```
docker-compose –version
```

Étape 2 : préparez l'image Docker

Étapes

1. Copiez le `CVO-Azure-Burst-To-Cloud.zip` dossier sur la machine virtuelle Linux que vous souhaitez utiliser pour déployer Cloud Volumes ONTAP :

```
scp -i ~/<private-key>.pem -r CVO-Azure-Burst-To-Cloud.zip
<azureuser>@<IP_ADDRESS_OF_VM>:<LOCATION_TO_BE_COPIED>
```

   - `private-key.pem` est votre fichier de clé privée pour la connexion sans mot de passe.
   - `azureuser` Est le nom d'utilisateur de la machine virtuelle.
   - `IP_ADDRESS_OF_VM` Est l'adresse IP de la machine virtuelle.
   - `LOCATION_TO_BE_COPIED` est l'emplacement où le dossier sera copié.

2. Extraire le `CVO-Azure-Burst-To-Cloud.zip` dossier. Vous pouvez extraire le dossier dans le répertoire actuel ou dans un emplacement personnalisé.

   Pour extraire le dossier dans le répertoire actuel, exécutez :

```
unzip CVO-Azure-Burst-To-Cloud.zip
```

   Pour extraire le dossier dans un emplacement personnalisé, exécutez :

```
unzip CVO-Azure-Burst-To-Cloud.zip -d ~/<your_folder_name>
```

3. Une fois le contenu extrait, accédez au `CVO_Azure_Deployment` dossier et exécutez la commande suivante pour afficher les fichiers :

```
 ls -la
```

   Vous devriez voir une liste de fichiers, similaire à l'exemple suivant :

```
drwxr-xr-x@ 11 user1 staff 352 May 5 13:56 .
drwxr-xr-x@ 5 user1 staff 160 May 5 14:24 ..
-rw-r--r--@ 1 user1 staff 324 May 5 13:18 .env
-rw-r--r--@ 1 user1 staff 1449 May 5 13:18 Dockerfile
-rw-r--r--@ 1 user1 staff 35149 May 5 13:18 LICENSE
-rw-r--r--@ 1 user1 staff 13356 May 5 14:26 README.md
-rw-r--r-- 1  user1 staff 354318151 May 5 13:51
cvo_azure_flexcache_ubuntu_image_latest
drwxr-xr-x@ 4 user1 staff 128 May 5 13:18 cvo_azure_variables
-rw-r--r--@ 1 user1 staff 996 May 5 13:18 docker-compose-deploy.yml
-rw-r--r--@ 1 user1 staff 1041 May 5 13:18 docker-compose-destroy.yml
-rw-r--r--@ 1 user1 staff 4771 May 5 13:18 sp_role.json
```

4. Localisez le `cvo_azure_flexcache_ubuntu_image_latest.tar.gz` fichier. Ce document contient l'image Docker requise pour déployer Cloud Volumes ONTAP pour Azure.
5. Décompressez le fichier :

```
docker load -i cvo_azure_flexcache_ubuntu_image_latest.tar.gz
```

6. Attendez quelques minutes que l'image Docker se charge, puis vérifiez que l'image Docker a bien été chargée :

```
docker images
```

Vous devez voir une image Docker nommée `cvo_azure_flexcache_ubuntu_image_latest` avec la `latest` balise, comme dans l'exemple suivant :

```
REPOSITORY TAG IMAGE ID CREATED SIZE
cvo_azure_flexcache_ubuntu_image latest 18db15a4d59c 2 weeks ago 1.14GB
```

Étape 3 : création de fichiers de variables d'environnement

À ce stade, vous devez créer deux fichiers de variables d'environnement. Un fichier est destiné à l'authentification des API Azure Resource Manager à l'aide des informations d'identification du principal de service. Le deuxième fichier permet de définir des variables d'environnement pour permettre aux modules de la console Terraform de localiser et d'authentifier les API Azure.

Étapes

1. Créez une entité de service.

Avant de pouvoir créer les fichiers de variables d'environnement, vous devez créer une entité de service en suivant les étapes de la section "Créez une application Azure Active Directory et une entité de service pouvant accéder aux ressources".

2. Attribuez le rôle **Contributor** à l'entité de service nouvellement créée.
3. Créez un rôle personnalisé.
   a. Recherchez le `sp_role.json` fichier et vérifiez les autorisations requises sous les actions répertoriées.
   b. Insérez ces autorisations et associez le rôle personnalisé au principal de service nouvellement créé.
4. Naviguez jusqu'à **certificats et secrets** et sélectionnez **Nouveau secret client** pour créer le secret client.

   Lorsque vous créez le secret client, vous devez enregistrer les détails de la colonne **valeur** car vous ne pourrez plus voir cette valeur. Vous devez également enregistrer les informations suivantes :

   - ID client
   - ID d'abonnement
   - ID locataire

     Vous aurez besoin de ces informations pour créer les variables d'environnement. Vous trouverez des informations sur l'ID client et l'ID locataire dans la section **Présentation** de l'interface utilisateur principale du service.
5. Créez les fichiers d'environnement.
   a. Créez le `azureauth.env` fichier à l'emplacement suivant :

      `path/to/env-file/azureauth.env`

      i. Ajoutez le contenu suivant au fichier :

         ClientID=<> clientSecret=<> subscriptionId=<> tenantId=<>

      Le format **doit** doit être exactement comme indiqué ci-dessus, sans espace entre la clé et la valeur.

   b. Créez le `credentials.env` fichier à l'emplacement suivant :

      `path/to/env-file/credentials.env`

      i. Ajoutez le contenu suivant au fichier :

         AZURE_TENANT_ID=<> AZURE_CLIENT_SECRET=<> AZURE_CLIENT_ID=<> AZURE_SUBSCRIPTION_ID=<>

   Le format **doit** doit être exactement comme indiqué ci-dessus, sans espace entre la clé et la valeur.
6. Ajoutez les chemins de fichier absolus au `.env` fichier.

   Entrez le chemin absolu du `azureauth.env` fichier d'environnement dans le `.env` fichier correspondant à la `AZURE_RM_CREDS` variable d'environnement.

   `AZURE_RM_CREDS=path/to/env-file/azureauth.env`

   Entrez le chemin absolu du `credentials.env` fichier d'environnement dans le `.env` fichier correspondant à la `BLUEXP_TF_AZURE_CREDS` variable d'environnement.

```
BLUEXP_TF_AZURE_CREDS=path/to/env-file/credentials.env
```

Étape 4 : Inscrivez-vous aux services intelligents NetApp

Inscrivez-vous aux services intelligents NetApp via votre fournisseur de cloud pour payer à l'heure (PAYGO) ou via un contrat annuel. Les services intelligents NetApp incluent NetApp Backup and Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience et NetApp Disaster Recovery. La classification des données NetApp est incluse dans votre abonnement sans frais supplémentaires

Étapes

1. Depuis le portail Azure, accédez à **SaaS** et sélectionnez *S'abonner aux services intelligents NetApp *.
2. Sélectionnez le plan **Cloud Manager (par Cap PYGO par heure, WORM et services de données)**.

   Vous pouvez utiliser le même groupe de ressources que Cloud Volumes ONTAP ou un autre groupe de ressources.

3. Configurez le portail de la console pour importer l'abonnement SaaS dans la console.

   Vous pouvez le configurer directement à partir du portail Azure en accédant à **Détails du produit et du plan** et en sélectionnant l'option **configurer le compte maintenant**.

   Vous serez ensuite redirigé vers le portail de la console pour confirmer la configuration.

4. Confirmez la configuration dans le portail de la console en sélectionnant **Enregistrer**.

Étape 5 : créer un volume externe

Vous devez créer un volume externe pour conserver les fichiers d'état Terraform et d'autres fichiers importants persistants. Vous devez vous assurer que les fichiers sont disponibles pour Terraform pour exécuter le workflow et les déploiements.

Étapes

1. Créer un volume externe en dehors de Docker compose :

   ```
   docker volume create « volume_name »
   ```

   Exemple :

   ```
   docker volume create cvo_azure_volume_dst
   ```

2. Utilisez l'une des options suivantes :

   a. Ajoutez un chemin de volume externe au `.env` fichier d'environnement.

      Vous devez suivre le format exact indiqué ci-dessous.

      Format :

      ```
      PERSISTENT_VOL=path/to/external/volume:/cvo_azure
      ```

Exemple :
`PERSISTENT_VOL=cvo_azure_volume_dst:/cvo_azure`

b. Ajoutez des partages NFS comme volume externe.

Assurez-vous que le conteneur Docker peut communiquer avec les partages NFS et que les autorisations appropriées, telles que lecture/écriture, sont configurées.

i. Ajoutez le chemin des partages NFS comme chemin d'accès au volume externe dans le fichier Docker compose, comme illustré ci-dessous : format :

`PERSISTENT_VOL=path/to/nfs/volume:/cvo_azure`

Exemple :
`PERSISTENT_VOL=nfs/mnt/document:/cvo_azure`

3. Accédez au `cvo_azure_variables` dossier.

Vous devriez voir les fichiers de variables suivants dans le dossier :

`terraform.tfvars`

`variables.tf`

4. Modifiez les valeurs à l'intérieur du `terraform.tfvars` fichier en fonction de vos besoins.

Vous devez lire la documentation spécifique lors de la modification de l'une des valeurs de variable du `terraform.tfvars` fichier. Ces valeurs peuvent varier en fonction de la région, des zones de disponibilité et d'autres facteurs pris en charge par Cloud Volumes ONTAP pour Azure. Notamment les licences, la taille des disques et la taille des machines virtuelles pour les nœuds uniques et les paires haute disponibilité.

Toutes les variables de support pour l'agent de console et les modules Cloud Volumes ONTAP Terraform sont déjà définies dans le `variables.tf` déposer. Vous devez faire référence aux noms de variables dans le `variables.tf` fichier avant de l'ajouter au `terraform.tfvars` déposer.

5. Selon vos besoins, vous pouvez activer ou désactiver FlexCache et FlexClone en définissant les options suivantes sur `true` ou `false`.

Les exemples suivants activent FlexCache et FlexClone :

- `is_flexcache_required = true`
- `is_flexclone_required = true`

6. Si nécessaire, vous pouvez récupérer la valeur de la variable Terraform `az_service_principal_object_id` à partir du service Azure Active Directory :

a. Accédez à **applications d'entreprise –> toutes les applications** et sélectionnez le nom du principal de service que vous avez créé précédemment.

b. Copiez l'ID d'objet et insérez la valeur de la variable Terraform :

`az_service_principal_object_id`

Étape 6 : déploiement de Cloud Volumes ONTAP pour Azure

Procédez comme suit pour déployer Cloud Volumes ONTAP pour Azure.

Étapes

1. Depuis le dossier racine, exécutez la commande suivante pour déclencher le déploiement :

   ```
   docker-compose up -d
   ```

   Deux conteneurs sont déclenchés, le premier conteneur déploie Cloud Volumes ONTAP et le second envoie des données de télémétrie à AutoSupport.

   Le deuxième conteneur attend jusqu'à ce que le premier conteneur termine toutes les étapes avec succès.

2. Surveiller la progression du processus de déploiement à l'aide des fichiers journaux :

   ```
   docker-compose logs -f
   ```

   Cette commande fournit des résultats en temps réel et capture les données dans les fichiers journaux suivants :

   `deployment.log`

   `telemetry_asup.log`

   Vous pouvez modifier le nom de ces fichiers journaux en modifiant le `.env` fichier à l'aide des variables d'environnement suivantes :

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   Les exemples suivants montrent comment modifier les noms des fichiers journaux :

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Une fois que vous avez terminé

Vous pouvez utiliser les étapes suivantes pour supprimer l'environnement temporaire et nettoyer les éléments créés pendant le processus de déploiement.

Étapes

1. Si vous avez déployé FlexCache, définissez l'option suivante dans le `terraform.tfvars` fichier, cela nettoie les volumes FlexCache et supprime l'environnement temporaire créé précédemment.

   `flexcache_operation = "destroy"`

   Les options possibles sont `deploy` et `destroy`.

2. Si vous avez déployé FlexClone, définissez l'option suivante dans le `terraform.tfvars` fichier, cela nettoie les volumes FlexClone et supprime l'environnement temporaire créé précédemment.

```
flexclone_operation = "destroy"
```

Les options possibles sont `deploy` et `destroy`.

Cloud Volumes ONTAP pour Google Cloud

Cloud Volumes ONTAP pour Google Cloud - en rafale vers le cloud

Cet article traite de la solution d'automatisation NetApp Cloud Volumes ONTAP pour Google Cloud, disponible pour les clients NetApp depuis le hub d'automatisation de la NetApp Console .

La solution d'automatisation Cloud Volumes ONTAP pour Google Cloud automatise le déploiement conteneurisé d'Cloud Volumes ONTAP pour Google Cloud, ce qui vous permet de déployer rapidement Cloud Volumes ONTAP pour Google Cloud sans aucune intervention manuelle.

Avant de commencer

- Vous devez télécharger le "Cloud Volumes ONTAP pour Google Cloud - en rafale vers le cloud" Solution d'automatisation via l'interface utilisateur Web de la console. La solution est conditionnée comme suit : `cvo_gcp_flexcache.zip`.
- Vous devez installer une machine virtuelle Linux sur le même réseau que Cloud Volumes ONTAP.
- Après avoir installé la machine virtuelle Linux, vous devez suivre les étapes de cette solution pour installer les dépendances requises.

Étape 1 : installez Docker et Docker compose

Installez Docker

Les étapes suivantes utilisent le logiciel de distribution Ubuntu 20.04 Debian Linux comme exemple. Les commandes que vous exécutez dépendent du logiciel de distribution Linux que vous utilisez. Reportez-vous à la documentation spécifique du logiciel de distribution Linux pour votre configuration.

Étapes

1. Installez Docker en exécutant les commandes suivantes :

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl gnupg-
agent software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

2. Vérifiez l'installation :

```
docker –version
```

3. Vérifiez qu'un groupe nommé « docker » a été créé sur votre système Linux. Si nécessaire, créez le groupe :

```
sudo groupadd docker
```

4. Ajoutez l'utilisateur qui doit accéder à Docker au groupe :

```
sudo usermod -aG docker $(whoami)
```

5. Vos modifications sont appliquées une fois que vous vous êtes déconnecter et que vous vous êtes de nouveau connecté au terminal. Vous pouvez également appliquer les modifications immédiatement :

```
newgrp docker
```

Installez Docker compose

Étapes

1. Installez Docker compose en exécutant les commandes suivantes `sudo` :

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose
```

2. Vérifiez l'installation :

```
docker-compose –version
```

Étape 2 : préparez l'image Docker

Étapes

1. Copiez le `cvo_gcp_flexcache.zip` dossier sur la machine virtuelle Linux que vous souhaitez utiliser pour déployer Cloud Volumes ONTAP :

```
scp -i ~/private-key.pem -r cvo_gcp_flexcache.zip
gcpuser@IP_ADDRESS_OF_VM:LOCATION_TO_BE_COPIED
```

- `private-key.pem` est votre fichier de clé privée pour la connexion sans mot de passe.
- `gcpuser` Est le nom d'utilisateur de la machine virtuelle.
- `IP_ADDRESS_OF_VM` Est l'adresse IP de la machine virtuelle.
- `LOCATION_TO_BE_COPIED` est l'emplacement où le dossier sera copié.

2. Extraire le `cvo_gcp_flexcache.zip` dossier. Vous pouvez extraire le dossier dans le répertoire actuel ou dans un emplacement personnalisé.

   Pour extraire le dossier dans le répertoire actuel, exécutez :

   ```
   unzip cvo_gcp_flexcache.zip
   ```

   Pour extraire le dossier dans un emplacement personnalisé, exécutez :

   ```
   unzip cvo_gcp_flexcache.zip -d ~/<your_folder_name>
   ```

3. Une fois le contenu extrait, exécutez la commande suivante pour afficher les fichiers :

   ```
    ls -la
   ```

   Vous devriez voir une liste de fichiers, similaire à l'exemple suivant :

   ```
       total 32
       drwxr-xr-x   8 user  staff   256 Mar 23 12:26 .
       drwxr-xr-x   6 user  staff   192 Mar 22 08:04 ..
       -rw-r--r--   1 user  staff   324 Apr 12 21:37 .env
       -rw-r--r--   1 user  staff  1449 Mar 23 13:19 Dockerfile
       drwxr-xr-x  15 user  staff   480 Mar 23 13:19 cvo_gcp_source_code
       drwxr-xr-x   4 user  staff   128 Apr 27 13:43 cvo_gcp_variables
       -rw-r--r--   1 user  staff   996 Mar 24 04:06 docker-compose-
   deploy.yml
       -rw-r--r--   1 user  staff  1041 Mar 24 04:06 docker-compose-
   destroy.yml
   ```

4. Localisez le `cvo_gcp_flexcache_ubuntu_image.tar` fichier. Ce document contient l'image Docker requise pour déployer Cloud Volumes ONTAP pour Google Cloud.
5. Décompressez le fichier :

```
docker load -i cvo_gcp_flexcache_ubuntu_image.tar
```

6. Attendez quelques minutes que l'image Docker se charge, puis vérifiez que l'image Docker a bien été chargée :

```
docker images
```

Vous devez voir une image Docker nommée `cvo_gcp_flexcache_ubuntu_image` avec la `latest` balise, comme dans l'exemple suivant :

```
REPOSITORY                                TAG         IMAGE ID       CREATED
SIZE
    cvo_gcp_flexcache_ubuntu_image     latest    18db15a4d59c   2 weeks
ago   1.14GB
```

Vous pouvez modifier le nom de l'image Docker si nécessaire. Si vous modifiez le nom de l'image Docker, veillez à mettre à jour le nom de l'image Docker dans les `docker-compose-deploy` fichiers et `docker-compose-destroy`.

Étape 3 : mettez à jour le fichier JSON

À ce stade, vous devez mettre à jour `cxo-automation-gcp.json` le fichier avec une clé de compte de service pour authentifier le fournisseur Google Cloud.

1. Créez un compte de service avec des autorisations pour déployer Cloud Volumes ONTAP et un agent de console"En savoir plus sur la création de comptes de service."
2. Téléchargez le fichier de clé pour le compte et mettez à jour le `cxo-automation-gcp.json` fichier avec les informations du fichier de clé. Le `cxo-automation-gcp.json` fichier se trouve dans le `cvo_gcp_variables` dossier.

Exemple

```
{
  "type": "service_account",
  "project_id": "",
  "private_key_id": "",
  "private_key": "",
  "client_email": "",
  "client_id": "",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "",
  "universe_domain": "googleapis.com"
}
```

Le format de fichier doit être exactement comme indiqué ci-dessus.

Étape 4 : Inscrivez-vous aux services intelligents NetApp

Inscrivez-vous aux services intelligents NetApp via votre fournisseur de cloud pour payer à l'heure (PAYGO) ou via un contrat annuel. Les services intelligents NetApp incluent NetApp Backup and Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience et NetApp Disaster Recovery. La classification des données NetApp est incluse dans votre abonnement sans frais supplémentaires.

Étapes

1. Accédez au"Console Google Cloud" et sélectionnez *S'abonner aux services intelligents NetApp *.
2. Configurez le portail de la console NetApp pour importer l'abonnement SaaS dans la console.

   Vous pouvez configurer cela directement depuis Google Cloud Platform. Vous serez redirigé vers le portail de la console pour confirmer la configuration.

3. Confirmez la configuration dans le portail de la console en sélectionnant **Enregistrer**.

Pour plus d'informations, consultez la section "Gérer les informations d'identification et les abonnements Google Cloud pour la console NetApp" .

Étape 5 : activation des API Google Cloud requises

Vous devez activer les API Google Cloud suivantes dans votre projet pour déployer Cloud Volumes ONTAP et l'agent de la console.

- API Cloud Deployment Manager V2
- API de journalisation cloud
- API Cloud Resource Manager
- API du moteur de calcul
- API de gestion des identités et des accès

"En savoir plus sur l'activation des API"

Étape 6 : créer un volume externe

Vous devez créer un volume externe pour conserver les fichiers d'état Terraform et d'autres fichiers importants persistants. Vous devez vous assurer que les fichiers sont disponibles pour Terraform pour exécuter le workflow et les déploiements.

Étapes

1. Créer un volume externe en dehors de Docker compose :

   ```
   docker volume create <volume_name>
   ```

   Exemple :

   ```
   docker volume create cvo_gcp_volume_dst
   ```

2. Utilisez l'une des options suivantes :

   a. Ajoutez un chemin de volume externe au `.env` fichier d'environnement.

      Vous devez suivre le format exact indiqué ci-dessous.

      Format :

      ```
      PERSISTENT_VOL=path/to/external/volume:/cvo_gcp
      ```

      Exemple :
      `PERSISTENT_VOL=cvo_gcp_volume_dst:/cvo_gcp`

   b. Ajoutez des partages NFS comme volume externe.

      Assurez-vous que le conteneur Docker peut communiquer avec les partages NFS et que les autorisations appropriées, telles que lecture/écriture, sont configurées.

      i. Ajoutez le chemin des partages NFS comme chemin d'accès au volume externe dans le fichier Docker compose, comme illustré ci-dessous : format :

         ```
         PERSISTENT_VOL=path/to/nfs/volume:/cvo_gcp
         ```

   Exemple :
   `PERSISTENT_VOL=nfs/mnt/document:/cvo_gcp`

3. Accédez au `cvo_gcp_variables` dossier.

   Le dossier doit contenir les fichiers suivants :

   - `terraform.tfvars`
   - `variables.tf`

4. Modifiez les valeurs à l'intérieur du `terraform.tfvars` fichier en fonction de vos besoins.

   Vous devez lire la documentation spécifique lors de la modification de l'une des valeurs de variable du `terraform.tfvars` fichier. Ces valeurs peuvent varier en fonction de la région, des zones de disponibilité et d'autres facteurs pris en charge par Cloud Volumes ONTAP pour Google Cloud. Notamment les licences, la taille des disques et la taille des machines virtuelles pour les nœuds uniques et les paires haute disponibilité.

   Toutes les variables de support pour l'agent de console et les modules Cloud Volumes ONTAP Terraform sont déjà définies dans le `variables.tf` déposer. Vous devez faire référence aux noms de variables dans le `variables.tf` fichier avant de l'ajouter au `terraform.tfvars` déposer.

5. Selon vos besoins, vous pouvez activer ou désactiver FlexCache et FlexClone en définissant les options suivantes sur `true` ou `false`.

   Les exemples suivants activent FlexCache et FlexClone :

   - `is_flexcache_required = true`
   - `is_flexclone_required = true`

Étape 7 : déploiement de Cloud Volumes ONTAP pour Google Cloud

Procédez comme suit pour déployer Cloud Volumes ONTAP pour Google Cloud.

Étapes

1. Depuis le dossier racine, exécutez la commande suivante pour déclencher le déploiement :

   ```
   docker-compose -f docker-compose-deploy.yml up -d
   ```

   Deux conteneurs sont déclenchés, le premier conteneur déploie Cloud Volumes ONTAP et le second envoie des données de télémétrie à AutoSupport.

   Le deuxième conteneur attend jusqu'à ce que le premier conteneur termine toutes les étapes avec succès.

2. Surveiller la progression du processus de déploiement à l'aide des fichiers journaux :

   ```
   docker-compose -f docker-compose-deploy.yml logs -f
   ```

   Cette commande fournit des résultats en temps réel et capture les données dans les fichiers journaux suivants :
   `deployment.log`

   `telemetry_asup.log`

   Vous pouvez modifier le nom de ces fichiers journaux en modifiant le `.env` fichier à l'aide des variables d'environnement suivantes :

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

Les exemples suivants montrent comment modifier les noms des fichiers journaux :

```
DEPLOYMENT_LOGS=<your_deployment_log_filename>.log

TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log
```

Une fois que vous avez terminé

Vous pouvez utiliser les étapes suivantes pour supprimer l'environnement temporaire et nettoyer les éléments créés pendant le processus de déploiement.

Étapes

1. Si vous avez déployé FlexCache, définissez l'option suivante dans le `terraform.tfvars` fichier, cela nettoie les volumes FlexCache et supprime l'environnement temporaire créé précédemment.

   ```
   flexcache_operation = "destroy"
   ```

   Les options possibles sont `deploy` et `destroy`.

2. Si vous avez déployé FlexClone, définissez l'option suivante dans le `terraform.tfvars` fichier, cela nettoie les volumes FlexClone et supprime l'environnement temporaire créé précédemment.

   ```
   flexclone_operation = "destroy"
   ```

   Les options possibles sont `deploy` et `destroy`.

ONTAP

Jour 0/1

Présentation de la solution ONTAP Day 0/1

Vous pouvez utiliser la solution d'automatisation ONTAP jour 0/1 pour déployer et configurer un cluster ONTAP à l'aide d'Ansible. La solution est disponible auprès de "Centre d'automatisation NetApp Console".

Des options de déploiement ONTAP flexibles

Selon les besoins, vous pouvez utiliser du matériel sur site ou simuler un ONTAP pour déployer et configurer un cluster ONTAP à l'aide d'Ansible.

Matériel sur site

Vous pouvez déployer cette solution à l'aide d'un matériel sur site exécutant ONTAP, tel qu'un système FAS ou AFF. Vous devez utiliser une VM Linux pour déployer et configurer le cluster ONTAP à l'aide d'Ansible.

Simuler ONTAP

Pour déployer cette solution à l'aide d'un simulateur ONTAP, vous devez télécharger la dernière version de Simulate ONTAP depuis le site de support NetApp. Simulate ONTAP est un simulateur virtuel pour le logiciel ONTAP. Simulez l'exécution de ONTAP dans un hyperviseur VMware sur un système Windows, Linux ou Mac. Pour les hôtes Windows et Linux, vous devez utiliser l'hyperviseur VMware Workstation pour exécuter cette

solution. Si vous disposez d'un Mac OS, utilisez l'hyperviseur VMware Fusion.

Effet superposé

Le framework Ansible simplifie le développement et la réutilisation de l'exécution de l'automatisation et des tâches logiques. Le cadre fait la distinction entre les tâches de prise de décision (couche logique) et les étapes d'exécution (couche d'exécution) dans l'automatisation. La compréhension du fonctionnement de ces couches vous permet de personnaliser la configuration.

Un « PlayBook » Ansible exécute une série de tâches du début à la fin. Ce `site.yml` PlayBook contient le `logic.yml` PlayBook et le `execution.yml` PlayBook.

Lors de l'exécution d'une demande, le `site.yml` PlayBook `logic.yml` appelle d'abord le PlayBook pour exécuter la `execution.yml` demande de service.

Vous n'êtes pas tenu d'utiliser la couche logique de l'infrastructure. La couche logique offre des options permettant d'étendre la capacité du cadre au-delà des valeurs codées en dur pour l'exécution. Vous pouvez ainsi personnaliser les fonctionnalités de la structure, le cas échéant.

Couche logique

La couche logique comprend les éléments suivants :

- `logic.yml`Le manuel de vente
- Fichiers de tâches logiques dans le `logic-tasks` répertoire

Cette couche logique facilite la prise de décision complexe sans nécessiter une intégration personnalisée significative (par exemple, connexion à ServiceNow). La couche logique est configurable et fournit des entrées aux microservices.

La possibilité de contourner la couche logique est également fournie. Si vous souhaitez contourner la couche logique, ne définissez pas la `logic_operation` variable. L'invocation directe du `logic.yml` PlayBook permet d'effectuer un certain niveau de débogage sans exécution. Vous pouvez utiliser une instruction "debug" pour vérifier que la valeur de `raw_service_request` est correcte.

Remarques importantes :

- Le `logic.yml` PlayBook recherche la `logic_operation` variable. Si la variable est définie dans la requête, elle charge un fichier de tâche à partir du `logic-tasks` répertoire. Le fichier de tâche doit être un fichier .yml. S'il n'y a pas de fichier de tâche correspondant et que la `logic_operation` variable est définie, la couche logique échoue.
- La valeur par défaut de la `logic_operation` variable est `no-op`. Si la variable n'est pas explicitement définie, elle prend par défaut la valeur `no-op`, qui n'exécute aucune opération.
- Si la `raw_service_request` variable est déjà définie, l'exécution passe à la couche d'exécution. Si la variable n'est pas définie, la couche logique échoue.

Couche d'exécution

La couche d'exécution comprend les éléments suivants :

- `execution.yml`Le manuel de vente

La couche d'exécution effectue des appels d'API pour configurer un cluster ONTAP. Le `execution.yml` PlayBook requiert que la `raw_service_request` variable soit définie lors de son exécution.

Prise en charge de la personnalisation

Vous pouvez personnaliser cette solution de différentes manières en fonction de vos besoins.

Les options de personnalisation sont les suivantes :

- Modification des playbooks Ansible
- Ajout de rôles

Personnalisez les fichiers Ansible

Le tableau suivant décrit les fichiers Ansible personnalisables contenus dans cette solution.

Emplacement	Description
`playbooks/inventory/hosts`	Contient un seul fichier avec une liste d'hôtes et de groupes.
`playbooks/group_vars/all/*`	Ansible constitue un moyen pratique d'appliquer des variables à plusieurs hôtes en même temps. Vous pouvez modifier tout ou partie des fichiers de ce dossier, y compris `cfg.yml`, `clusters.yml`, `defaults.yml`, `services.yml`, `standards.yml` et `vault.yml`.
`playbooks/logic-tasks`	Soutient les tâches de prise de décision au sein d'Ansible et maintient la séparation de la logique et de l'exécution. Vous pouvez ajouter à ce dossier des fichiers correspondant au service concerné.
`playbooks/vars/*`	Valeurs dynamiques utilisées dans les playbooks et les rôles Ansible pour assurer la personnalisation, la flexibilité et la réutilisation des configurations. Si nécessaire, vous pouvez modifier tout ou partie des fichiers de ce dossier.

Personnaliser les rôles

Vous pouvez également personnaliser la solution en ajoutant ou en modifiant les rôles Ansible, également appelés microservices. Pour plus de détails, voir "Personnaliser".

Préparez-vous à utiliser la solution ONTAP Day 0/1

Avant de déployer la solution d'automatisation, vous devez préparer l'environnement ONTAP et installer et configurer Ansible.

Considérations de planification initiale

Avant d'utiliser cette solution pour déployer un cluster ONTAP, passez en revue les exigences et considérations suivantes.

Critères de base

Pour utiliser cette solution, vous devez répondre aux exigences de base suivantes :

- Vous devez avoir accès au logiciel ONTAP, sur site ou via un simulateur ONTAP.
- Vous devez savoir comment utiliser le logiciel ONTAP.
- Vous devez savoir utiliser les outils logiciels d'automatisation Ansible.

Planification

Avant de déployer cette solution d'automatisation, vous devez décider :

- Emplacement où vous allez exécuter le nœud de contrôle Ansible.
- Le système ONTAP, du matériel sur site ou un simulateur ONTAP.
- Que vous ayez besoin ou non de personnalisation.

Préparez le système ONTAP

Que vous utilisiez un système ONTAP sur site ou que vous simulez ONTAP, vous devez préparer l'environnement avant de pouvoir déployer la solution d'automatisation.

Si vous le souhaitez, installez et configurez Simulate ONTAP

Si vous souhaitez déployer cette solution via un simulateur ONTAP, vous devez télécharger et exécuter Simulate ONTAP.

Avant de commencer

- Vous devez télécharger et installer l'hyperviseur VMware que vous allez utiliser pour exécuter Simulate ONTAP.
  - Si vous disposez d'un système d'exploitation Windows ou Linux, utilisez VMware Workstation.
  - Si vous disposez d'un système d'exploitation Mac, utilisez VMware Fusion.

> ⓘ Si vous utilisez un Mac OS, vous devez disposer d'un processeur Intel.

Étapes

Utilisez la procédure suivante pour installer deux simulateurs ONTAP dans votre environnement local :

1. Télécharger Simulate ONTAP à partir du "Site de support NetApp".

> ⓘ Bien que vous installiez deux simulateurs ONTAP, il vous suffit de télécharger une seule copie du logiciel.

2. Si ce n'est pas déjà fait, démarrez votre application VMware.
3. Recherchez le fichier de simulateur téléchargé et cliquez avec le bouton droit de la souris pour l'ouvrir avec l'application VMware.
4. Définissez le nom de la première instance ONTAP.
5. Attendez le démarrage du simulateur et suivez les instructions pour créer un cluster à un seul nœud.

   Répétez les étapes pour la deuxième instance ONTAP.

6. Si vous le souhaitez, ajoutez un complément de disque complet.

   Depuis chaque cluster, exécutez les commandes suivantes :

```
security unlock -username <user_01>
security login password -username <user_01>
set -priv advanced
systemshell local
disk assign -all -node <Cluster-01>-01
```

État du système ONTAP

Vous devez vérifier l'état initial du système ONTAP, qu'il soit sur site ou exécuté via un simulateur ONTAP.

Vérifiez que la configuration système ONTAP requise est respectée :

- ONTAP est installé et fonctionne, aucun cluster n'est encore défini.
- La ONTAP démarre et affiche l'adresse IP pour accéder au cluster.
- Le réseau est accessible.
- Vous avez des identifiants d'administrateur.
- La bannière message du jour (MOTD) s'affiche avec l'adresse de gestion.

Installez le logiciel d'automatisation requis

Cette section explique comment installer Ansible et préparer la solution d'automatisation au déploiement.

Installez Ansible

Ansible peut être installé sur les systèmes Linux ou Windows.

La méthode de communication par défaut utilisée par Ansible pour communiquer avec un cluster ONTAP est SSH.

Reportez-vous "Mise en route avec NetApp et Ansible : installation d'Ansible"à pour installer Ansible.

Ansible doit être installé sur le nœud de contrôle du système.

Téléchargez et préparez la solution d'automatisation

Vous pouvez suivre les étapes suivantes pour télécharger et préparer la solution d'automatisation pour le déploiement.

1. Téléchargez le "ONTAP - jour 0/1 et amp ; vérifications de l'état" Solution d'automatisation via l'interface utilisateur Web de la console. La solution est conditionnée comme suit : `ONTAP_DAY0_DAY1.zip`.
2. Extrayez le dossier zip et copiez les fichiers à l'emplacement souhaité sur le nœud de contrôle dans votre environnement Ansible.

Configuration initiale de la structure Ansible

Effectuez la configuration initiale du framework Ansible :

1. Accédez à `playbooks/inventory/group_vars/all`.
2. Déchiffrement du `vault.yml` fichier :

   ```
   ansible-vault decrypt playbooks/inventory/group_vars/all/vault.yml
   ```

   Lorsque vous êtes invité à entrer le mot de passe du coffre-fort, entrez le mot de passe temporaire suivant :

   ```
   NetApp123!
   ```

« NetApp123! » Est un mot de passe temporaire permettant de décrypter `vault.yml` le fichier et le mot de passe du coffre-fort correspondant. Après la première utilisation, vous **devez** crypter le fichier à l'aide de votre propre mot de passe.

3. Modifiez les fichiers Ansible suivants :

   - `clusters.yml` - Modifiez les valeurs de ce fichier en fonction de votre environnement.
   - `vault.yml` - Après avoir décrypté le fichier, modifiez les valeurs du cluster ONTAP, du nom d'utilisateur et du mot de passe en fonction de votre environnement.
   - `cfg.yml` - Définissez le chemin d'accès au fichier pour `log2file` et `show_request` sous `cfg` à `True` pour afficher le `raw_service_request`.

     La `raw_service_request` variable s'affiche dans les fichiers journaux et pendant l'exécution.

   Chaque fichier répertorié contient des commentaires avec des instructions sur la façon de le modifier en fonction de vos besoins.

4. Re-crypter le `vault.yml` fichier :

   ```
   ansible-vault encrypt playbooks/inventory/group_vars/all/vault.yml
   ```

   Vous êtes invité à choisir un nouveau mot de passe pour le coffre-fort lors du cryptage.

5. Accédez à `playbooks/inventory/hosts` un interpréteur Python valide et définissez-le.
6. Déployer le `framework_test` service :

   La commande suivante exécute le `na_ontap_info` module avec une `gather_subset` valeur de `cluster_identity_info`. Cette opération valide que la configuration de base est correcte et vérifie que vous pouvez communiquer avec le cluster.

   ```
   ansible-playbook -i inventory/hosts site.yml -e
   cluster_name=<CLUSTER_NAME>
   -e logic_operation=framework-test
   ```

   Exécutez la commande pour chaque cluster.

   Si le résultat est réussi, vous devriez voir un résultat similaire à l'exemple suivant :

   ```
   PLAY RECAP
   ********************************************************************************
   *********
   localhost : ok=12 changed=1 unreachable=0 failed=0 skipped=6
   The key is 'rescued=0' and 'failed=0'..
   ```

Déployez le cluster ONTAP à l'aide de la solution

Une fois la préparation et la planification terminée, vous êtes prêt à utiliser la solution ONTAP Day 0/1 pour configurer rapidement un cluster ONTAP à l'aide d'Ansible.

À tout moment au cours des étapes de cette section, vous pouvez choisir de tester une demande au lieu de l'exécuter. Pour tester une demande, remplacez le `site.yml` PlayBook sur la ligne de commande par `logic.yml`.

> L'`docs/tutorial-requests.txt`emplacement contient la version finale de toutes les demandes de service utilisées tout au long de cette procédure. Si vous avez des difficultés à exécuter une demande de service, vous pouvez copier la demande pertinente du `tutorial-requests.txt` fichier vers l' `playbooks/inventory/group_vars/all/tutorial-requests.yml`emplacement et modifier les valeurs codées en dur comme requis (adresse IP, noms d'agrégats, etc.). Vous devriez alors être en mesure d'exécuter la demande avec succès.

Avant de commencer

- Ansible doit être installé sur votre système.
- Vous devez avoir téléchargé la solution ONTAP Day 0/1 et extrait le dossier à l'emplacement souhaité sur le nœud de contrôle Ansible.
- L'état du système ONTAP doit répondre aux exigences et vous devez disposer des informations d'identification nécessaires.
- Vous devez avoir effectué toutes les tâches requises décrites dans la "Préparation" section.

> Dans les exemples de cette solution, on utilise les noms « Cluster_01 » et « Cluster_02 » pour les deux clusters. Vous devez remplacer ces valeurs par les noms des clusters de votre environnement.

Étape 1 : configuration initiale du cluster

À ce stade, vous devez effectuer certaines étapes initiales de configuration du cluster.

Étapes

1. Naviguez jusqu'à `playbooks/inventory/group_vars/all/tutorial-requests.yml` l'emplacement et examinez la `cluster_initial` demande dans le fichier. Apportez les modifications nécessaires à votre environnement.
2. Créez un fichier dans le `logic-tasks` dossier de la demande de service. Par exemple, créez un fichier appelé `cluster_initial.yml`.

   Copiez les lignes suivantes dans le nouveau fichier :

```
- name: Validate required inputs
  ansible.builtin.assert:
    that:
    - service is defined

- name: Include data files
  ansible.builtin.include_vars:
    file:   "{{ data_file_name }}.yml"
  loop:
  - common-site-stds
  - user-inputs
  - cluster-platform-stds
  - vserver-common-stds
  loop_control:
    loop_var:    data_file_name

- name: Initial cluster configuration
  set_fact:
    raw_service_request:
```

3. Définissez la `raw_service_request` variable.

   Vous pouvez utiliser l'une des options suivantes pour définir la `raw_service_request` variable dans `cluster_initial.yml` le fichier que vous avez créé dans le `logic-tasks` dossier :

   - **Option 1** : définissez manuellement la `raw_service_request` variable.

     Ouvrez le `tutorial-requests.yml` fichier à l'aide d'un éditeur et copiez le contenu de la ligne 11 à la ligne 165. Collez le contenu sous la `raw service request` variable dans le nouveau `cluster_initial.yml` fichier, comme indiqué dans les exemples suivants :

```
3   # This file contains the final version of the various service
4   # requests used throughout the tutorial in TUTORIAL.md.
5   #-----------------------------------------------------------------
6   #-----------------------------------------------------------------
7   # cluster_initial:
8   #
9   #-----------------------------------------------------------------
11    service:          cluster_initial
12    operation:        create
13    std_name:         none
14    req_details:
15
16      ontap_aggr:
17      - hostname:                 "{{ cluster_name }}"
18        disk_count:               24
19        name:                     n01_aggr1
20        nodes:                    "{{ cluster_name }}-01"
```

Montrer l'exemple

Exemple de `cluster_initial.yml` fichier :

```
- name: Validate required inputs
  ansible.builtin.assert:
    that:
    - service is defined

- name: Include data files
  ansible.builtin.include_vars:
    file:   "{{ data_file_name }}.yml"
  loop:
  - common-site-stds
  - user-inputs
  - cluster-platform-stds
  - vserver-common-stds
  loop_control:
    loop_var:    data_file_name

- name: Initial cluster configuration
  set_fact:
    raw_service_request:
     service:          cluster_initial
     operation:         create
     std_name:           none
     req_details:

      ontap_aggr:
      - hostname:                   "{{ cluster_name }}"
        disk_count:                 24
        name:                       n01_aggr1
        nodes:                      "{{ cluster_name }}-01"
        raid_type:                  raid4

      - hostname:                   "{{ peer_cluster_name }}"
        disk_count:                 24
        name:                       n01_aggr1
        nodes:                      "{{ peer_cluster_name }}-01"
        raid_type:                  raid4

      ontap_license:
      - hostname:                   "{{ cluster_name }}"
        license_codes:
        - XXXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXXAAAAAAAAAAAAAA
```

```
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA

    - hostname:                    "{{ peer_cluster_name }}"
      license_codes:
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
        - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
```

```
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA
      - XXXXXXXXXXXXXXAAAAAAAAAAAAAA

  ontap_motd:
  - hostname:                    "{{ cluster_name }}"
    vserver:                     "{{ cluster_name }}"
    message:                     "New MOTD"

  - hostname:                    "{{ peer_cluster_name }}"
    vserver:                     "{{ peer_cluster_name }}"
    message:                     "New MOTD"

  ontap_interface:
  - hostname:                    "{{ cluster_name }}"
    vserver:                     "{{ cluster_name }}"
    interface_name:              ic01
    role:                        intercluster
    address:                     10.0.0.101
    netmask:                     255.255.255.0
    home_node:                   "{{ cluster_name }}-01"
    home_port:                   e0c
    ipspace:                     Default
    use_rest:                    never

  - hostname:                    "{{ cluster_name }}"
    vserver:                     "{{ cluster_name }}"
    interface_name:              ic02
    role:                        intercluster
    address:                     10.0.0.101
    netmask:                     255.255.255.0
    home_node:                   "{{ cluster_name }}-01"
    home_port:                   e0c
```

```
    ipspace:                    Default
    use_rest:                   never

  - hostname:                   "{{ peer_cluster_name }}"
    vserver:                    "{{ peer_cluster_name }}"
    interface_name:             ic01
    role:                       intercluster
    address:                    10.0.0.101
    netmask:                    255.255.255.0
    home_node:                  "{{ peer_cluster_name }}-01"
    home_port:                  e0c
    ipspace:                    Default
    use_rest:                   never

  - hostname:                   "{{ peer_cluster_name }}"
    vserver:                    "{{ peer_cluster_name }}"
    interface_name:             ic02
    role:                       intercluster
    address:                    10.0.0.101
    netmask:                    255.255.255.0
    home_node:                  "{{ peer_cluster_name }}-01"
    home_port:                  e0c
    ipspace:                    Default
    use_rest:                   never

  ontap_cluster_peer:
  - hostname:                   "{{ cluster_name }}"
    dest_cluster_name:          "{{ peer_cluster_name }}"
    dest_intercluster_lifs:     "{{ peer_lifs }}"
    source_cluster_name:        "{{ cluster_name }}"
    source_intercluster_lifs:   "{{ cluster_lifs }}"
    peer_options:
      hostname:                 "{{ peer_cluster_name }}"
```

- **Option 2** : utilisez un modèle Jinja pour définir la demande :

  Vous pouvez également utiliser le format de modèle Jinja suivant pour obtenir la `raw_service_request` valeur.

```
raw_service_request: "{{ cluster_initial }}"
```

4. Effectuez la configuration initiale du cluster pour le premier cluster :

```
ansible-playbook -i inventory/hosts site.yml -e
cluster_name=<Cluster_01>
```

Vérifiez qu'il n'y a pas d'erreur avant de continuer.

5. Répéter la commande pour le second cluster :

```
ansible-playbook -i inventory/hosts site.yml -e
cluster_name=<Cluster_02>
```

Vérifiez qu'il n'y a pas d'erreur pour le second cluster.

Lorsque vous faites défiler vers le haut vers le début de la sortie Ansible, vous devez voir la demande qui a été envoyée à la structure, comme illustré ci-dessous :

Montrer l'exemple

```
TASK [Show the raw_service_request]
********************************************************************
****************************************
ok: [localhost] => {
    "raw_service_request": {
        "operation": "create",
        "req_details": {
            "ontap_aggr": [
                {
                    "disk_count": 24,
                    "hostname": "Cluster_01",
                    "name": "n01_aggr1",
                    "nodes": "Cluster_01-01",
                    "raid_type": "raid4"
                }
            ],
            "ontap_license": [
                {
                    "hostname": "Cluster_01",
                    "license_codes": [
                        "XXXXXXXXXXXXXXXAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
```

```
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA",
                        "XXXXXXXXXXXXXXAAAAAAAAAAAAA"
                    ]
                }
            ],
            "ontap_motd": [
                {
                    "hostname": "Cluster_01",
                    "message": "New MOTD",
                    "vserver": "Cluster_01"
                }
            ]
        },
        "service": "cluster_initial",
        "std_name": "none"
    }
}
```

6. Connectez-vous à chaque instance ONTAP et vérifiez que la demande a abouti.

Étape 2 : configurer les LIFs intercluster

Vous pouvez maintenant configurer les LIFs intercluster en ajoutant les définitions du LIF à la requête et en `cluster_initial` définissant le `ontap_interface` microservice.

La définition de service et la demande fonctionnent ensemble pour déterminer l'action :

- Si vous fournissez une demande de service pour un microservice qui ne figure pas dans les définitions de service, la demande n'est pas exécutée.
- Si vous fournissez une demande de service avec un ou plusieurs microservices définis dans les définitions de service, mais que vous n'avez pas inclus dans la requête, la requête n'est pas exécutée.

Ce `execution.yml` PlayBook évalue la définition de service en analysant la liste des microservices dans l'ordre indiqué :

- S'il existe une entrée dans la demande avec une clé de dictionnaire correspondant à l' `args`entrée contenue dans les définitions de microservice, la demande est exécutée.
- S'il n'y a pas d'entrée correspondante dans la demande de service, la demande est ignorée sans erreur.

Étapes

1. Accédez au `cluster_initial.yml` fichier que vous avez créé précédemment et modifiez la demande en ajoutant les lignes suivantes aux définitions de la demande :

```
ontap_interface:
- hostname:                    "{{ cluster_name }}"
  vserver:                     "{{ cluster_name }}"
  interface_name:              ic01
  role:                        intercluster
  address:                     <ip_address>
  netmask:                     <netmask_address>
  home_node:                   "{{ cluster_name }}-01"
  home_port:                   e0c
  ipspace:                     Default
  use_rest:                    never

- hostname:                    "{{ cluster_name }}"
  vserver:                     "{{ cluster_name }}"
  interface_name:              ic02
  role:                        intercluster
  address:                     <ip_address>
  netmask:                     <netmask_address>
  home_node:                   "{{ cluster_name }}-01"
  home_port:                   e0c
  ipspace:                     Default
  use_rest:                    never

- hostname:                    "{{ peer_cluster_name }}"
  vserver:                     "{{ peer_cluster_name }}"
  interface_name:              ic01
  role:                        intercluster
  address:                     <ip_address>
  netmask:                     <netmask_address>
  home_node:                   "{{ peer_cluster_name }}-01"
  home_port:                   e0c
  ipspace:                     Default
  use_rest:                    never

- hostname:                    "{{ peer_cluster_name }}"
  vserver:                     "{{ peer_cluster_name }}"
  interface_name:              ic02
  role:                        intercluster
  address:                     <ip_address>
  netmask:                     <netmask_address>
  home_node:                   "{{ peer_cluster_name }}-01"
  home_port:                   e0c
  ipspace:                     Default
  use_rest:                    never
```

2. Lancer la commande :

```
ansible-playbook -i inventory/hosts  site.yml -e
cluster_name=<Cluster_01> -e peer_cluster_name=<Cluster_02>
```

3. Connectez-vous à chaque instance pour vérifier si les LIFs ont été ajoutées au cluster :

Montrer l'exemple

```
Cluster_01::> net int show
  (network interface show)
           Logical    Status     Network            Current
Current Is
Vserver     Interface  Admin/Oper Address/Mask       Node
Port    Home
----------- ---------- ---------- ------------------ -------------
------- ----
Cluster_01
            Cluster_01-01_mgmt up/up 10.0.0.101/24   Cluster_01-01
e0c     true
            Cluster_01-01_mgmt_auto up/up 10.101.101.101/24
Cluster_01-01 e0c true
            cluster_mgmt up/up    10.0.0.110/24      Cluster_01-01
e0c     true
5 entries were displayed.
```

Le résultat indique que les LIFs ont été **non** ajoutées. En effet, le `ontap_interface` microservice doit toujours être défini dans `services.yml` le fichier.

4. Vérifier que les LIFs ont été ajoutées à la `raw_service_request` variable.

Montrer l'exemple

L'exemple suivant montre que les LIFs ont été ajoutées à la requête :


```
            "ontap_interface": [
                 {
                     "address": "10.0.0.101",
                     "home_node": "Cluster_01-01",
                     "home_port": "e0c",
                     "hostname": "Cluster_01",
                     "interface_name": "ic01",
                     "ipspace": "Default",
                     "netmask": "255.255.255.0",
                     "role": "intercluster",
                     "use_rest": "never",
                     "vserver": "Cluster_01"
                 },
                 {
                     "address": "10.0.0.101",
                     "home_node": "Cluster_01-01",
                     "home_port": "e0c",
                     "hostname": "Cluster_01",
                     "interface_name": "ic02",
                     "ipspace": "Default",
                     "netmask": "255.255.255.0",
                     "role": "intercluster",
                     "use_rest": "never",
                     "vserver": "Cluster_01"
                 },
                 {
                     "address": "10.0.0.101",
                     "home_node": "Cluster_02-01",
                     "home_port": "e0c",
                     "hostname": "Cluster_02",
                     "interface_name": "ic01",
                     "ipspace": "Default",
                     "netmask": "255.255.255.0",
                     "role": "intercluster",
                     "use_rest": "never",
                     "vserver": "Cluster_02"
                 },
                 {
                     "address": "10.0.0.126",
                     "home_node": "Cluster_02-01",
                     "home_port": "e0c",
                     "hostname": "Cluster_02",
```

```
                    "interface_name": "ic02",
                    "ipspace": "Default",
                    "netmask": "255.255.255.0",
                    "role": "intercluster",
                    "use_rest": "never",
                    "vserver": "Cluster_02"
                }
            ],
```

5. Définissez le `ontap_interface` microservice sous `cluster_initial` dans `services.yml` le fichier.

   Copiez les lignes suivantes dans le fichier pour définir le microservice :

```
        - name: ontap_interface
          args: ontap_interface
          role: na/ontap_interface
```

6. Maintenant que le `ontap_interface` microservice a été défini dans la demande et le `services.yml` fichier, réexécutez la demande :

```
ansible-playbook -i inventory/hosts  site.yml -e
cluster_name=<Cluster_01> -e peer_cluster_name=<Cluster_02>
```

7. Connectez-vous à chaque instance ONTAP et vérifiez que les LIF ont été ajoutées.

Étape 3 : si vous le souhaitez, configurez plusieurs clusters

Si nécessaire, vous pouvez configurer plusieurs clusters dans la même demande. Vous devez fournir des noms de variable pour chaque cluster lors de la définition de la demande.

Étapes

1. Ajoutez une entrée pour le second cluster dans le `cluster_initial.yml` fichier pour configurer les deux clusters dans la même demande.

   L'exemple suivant affiche le `ontap_aggr` champ après l'ajout de la deuxième entrée.

```
ontap_aggr:
 - hostname:                    "{{ cluster_name }}"
   disk_count:                  24
   name:                        n01_aggr1
   nodes:                       "{{ cluster_name }}-01"
   raid_type:                   raid4

 - hostname:                    "{{ peer_cluster_name }}"
   disk_count:                  24
   name:                        n01_aggr1
   nodes:                       "{{ peer_cluster_name }}-01"
   raid_type:                   raid4
```

2. Appliquez les modifications pour tous les autres éléments sous `cluster_initial`.
3. Ajouter le cluster peering à la demande en copiant les lignes suivantes dans le fichier :

```
ontap_cluster_peer:
- hostname:                     "{{ cluster_name }}"
  dest_cluster_name:            "{{ cluster_peer }}"
  dest_intercluster_lifs:       "{{ peer_lifs }}"
  source_cluster_name:          "{{ cluster_name }}"
  source_intercluster_lifs:     "{{ cluster_lifs }}"
  peer_options:
    hostname:                   "{{ cluster_peer }}"
```

4. Exécutez la requête Ansible :

```
ansible-playbook -i inventory/hosts -e cluster_name=<Cluster_01>
site.yml -e peer_cluster_name=<Cluster_02> -e
cluster_lifs=<cluster_lif_1_IP_address,cluster_lif_2_IP_address>
-e peer_lifs=<peer_lif_1_IP_address,peer_lif_2_IP_address>
```

Étape 4 : configuration initiale du SVM

À ce stade de la procédure, on configure les SVM au sein du cluster.

Étapes

1. Mettre à jour la `svm_initial` requête dans `tutorial-requests.yml` le fichier pour configurer une relation SVM et SVM peer.

   Vous devez configurer les éléments suivants :

   - SVM

- Relation entre SVM
- L'interface SVM pour chaque SVM

2. Mettez à jour les définitions de variables dans les `svm_initial` définitions de la demande. Vous devez modifier les définitions de variables suivantes :
   - `cluster_name`
   - `vserver_name`
   - `peer_cluster_name`
   - `peer_vserver`

   Pour mettre à jour les définitions, supprimez le **'{}'** après `req_details` pour la `svm_initial` définition et ajoutez la définition correcte.

3. Créez un fichier dans le `logic-tasks` dossier de la demande de service. Par exemple, créez un fichier appelé `svm_initial.yml`.

   Copiez les lignes suivantes dans le fichier :

```
- name: Validate required inputs
  ansible.builtin.assert:
    that:
    - service is defined

- name: Include data files
  ansible.builtin.include_vars:
    file:   "{{ data_file_name }}.yml"
  loop:
  - common-site-stds
  - user-inputs
  - cluster-platform-stds
  - vserver-common-stds
  loop_control:
    loop_var:    data_file_name

- name: Initial SVM configuration
  set_fact:
    raw_service_request:
```

4. Définissez la `raw_service_request` variable.

   Vous pouvez utiliser l'une des options suivantes pour définir la `raw_service_request` variable pour `svm_initial` dans le `logic-tasks` dossier :

   - **Option 1** : définissez manuellement la `raw_service_request` variable.

     Ouvrez le `tutorial-requests.yml` fichier à l'aide d'un éditeur et copiez le contenu de la ligne 179

à la ligne 222. Collez le contenu sous la `raw service request` variable dans le nouveau `svm_initial.yml` fichier, comme indiqué dans les exemples suivants :

Montrer l'exemple

Exemple de `svm_initial.yml` fichier :

```
- name: Validate required inputs
  ansible.builtin.assert:
    that:
    - service is defined

- name: Include data files
  ansible.builtin.include_vars:
    file:   "{{ data_file_name }}.yml"
  loop:
  - common-site-stds
  - user-inputs
  - cluster-platform-stds
  - vserver-common-stds
  loop_control:
    loop_var:    data_file_name

- name: Initial SVM configuration
  set_fact:
    raw_service_request:
     service:          svm_initial
     operation:        create
     std_name:         none
     req_details:

      ontap_vserver:
      - hostname:                   "{{ cluster_name }}"
        name:                       "{{ vserver_name }}"
        root_volume_aggregate:      n01_aggr1

      - hostname:                   "{{ peer_cluster_name }}"
       name:                       "{{ peer_vserver }}"
       root_volume_aggregate:      n01_aggr1

      ontap_vserver_peer:
      - hostname:                   "{{ cluster_name }}"
        vserver:                    "{{ vserver_name }}"
        peer_vserver:               "{{ peer_vserver }}"
        applications:               snapmirror
        peer_options:
          hostname:                 "{{ peer_cluster_name }}"

      ontap_interface:
```

```
      - hostname:                    "{{ cluster_name }}"
        vserver:                     "{{ vserver_name }}"
        interface_name:              data01
        role:                        data
        address:                     10.0.0.200
        netmask:                     255.255.255.0
        home_node:                   "{{ cluster_name }}-01"
        home_port:                   e0c
        ipspace:                     Default
        use_rest:                    never

      - hostname:                    "{{ peer_cluster_name }}"
        vserver:                     "{{ peer_vserver }}"
        interface_name:              data01
        role:                        data
        address:                     10.0.0.201
        netmask:                     255.255.255.0
        home_node:                   "{{ peer_cluster_name }}-01"
        home_port:                   e0c
        ipspace:                     Default
        use_rest:                    never
```

- **Option 2** : utilisez un modèle Jinja pour définir la demande :

  Vous pouvez également utiliser le format de modèle Jinja suivant pour obtenir la `raw_service_request` valeur.

```
raw_service_request: "{{ svm_initial }}"
```

5. Exécutez la demande :

```
ansible-playbook -i inventory/hosts -e cluster_name=<Cluster_01> -e
peer_cluster_name=<Cluster_02> -e peer_vserver=<SVM_02>  -e
vserver_name=<SVM_01> site.yml
```

6. Connectez-vous à chaque instance ONTAP et validez la configuration.
7. Ajouter les interfaces SVM

   Définissez le `ontap_interface` service sous `svm_initial` dans `services.yml` le fichier et exécutez à nouveau la demande :

```
ansible-playbook -i inventory/hosts -e cluster_name=<Cluster_01> -e
peer_cluster_name=<Cluster_02> -e peer_vserver=<SVM_02>  -e
vserver_name=<SVM_01> site.yml
```

8. Connectez-vous à chaque instance ONTAP et vérifiez que les interfaces du SVM ont été configurées.

Étape 5 : vous pouvez éventuellement définir une demande de service de façon dynamique

Dans les étapes précédentes, la `raw_service_request` variable est codée en dur. Ceci est utile pour l'apprentissage, le développement et les tests. Vous pouvez également générer une demande de service de manière dynamique.

La section suivante fournit une option pour produire dynamiquement les requis si vous ne voulez pas les `raw_service_request` intégrer à des systèmes de niveau supérieur.

- Si la `logic_operation` variable n'est pas définie dans la commande, le `logic.yml` fichier n'importe aucun fichier du `logic-tasks` dossier. Cela signifie que le `raw_service_request` doit être défini en dehors d'Ansible et fourni au cadre d'exécution.
- Un nom de fichier de tâche dans le `logic-tasks` dossier doit correspondre à la valeur de la `logic_operation` variable sans l'extension .yml.
- Les fichiers de tâches dans le `logic-tasks` dossier définissent dynamiquement un `raw_service_request`. la seule exigence est qu'un valide `raw_service_request` soit défini comme la dernière tâche dans le fichier approprié.

Définition dynamique d'une demande de service

Il existe plusieurs façons d'appliquer une tâche logique pour définir dynamiquement une demande de service. Certaines de ces options sont répertoriées ci-dessous :

- À l'aide d'un fichier de tâches Ansible du `logic-tasks` dossier
- Appel d'un rôle personnalisé qui renvoie des données adaptées à la conversion en `raw_service_request` variable.
- Appel d'un autre outil hors de l'environnement Ansible pour fournir les données requises. Par exemple, un appel d'API REST vers Active IQ Unified Manager.

Les exemples de commandes suivants définissent de façon dynamique une demande de service pour chaque cluster à l'aide du `tutorial-requests.yml` fichier :

```
ansible-playbook -i inventory/hosts -e cluster2provision=Cluster_01
-e logic_operation=tutorial-requests site.yml
```

```
ansible-playbook -i inventory/hosts -e cluster2provision=Cluster_02
-e logic_operation=tutorial-requests site.yml
```

Étape 6 : déployer la solution ONTAP dès le début de l'année 0/1

À ce stade, vous devez déjà avoir terminé les tâches suivantes :

- Révision et modification de tous les fichiers dans en `playbooks/inventory/group_vars/all` fonction de vos besoins. Chaque fichier contient des commentaires détaillés qui vous aideront à effectuer les modifications.
- Ajout de tous les fichiers de tâches requis dans le `logic-tasks` répertoire.
- Ajout de tous les fichiers de données requis dans le `playbook/vars` répertoire.

Utilisez les commandes suivantes pour déployer la solution ONTAP Day 0/1 et vérifier l'état de santé de votre déploiement :

> ⓘ À ce stade, vous devez déjà avoir décrypté et modifié le `vault.yml` fichier et il doit être crypté avec votre nouveau mot de passe.

- Exécutez le service ONTAP jour 0 :

```
ansible-playbook -i playbooks/inventory/hosts playbooks/site.yml -e
logic_operation=cluster_day_0 -e service=cluster_day_0 -vvvv --ask-vault
-pass <your_vault_password>
```

- Exécutez le service ONTAP Day 1 :

```
ansible-playbook -i playbooks/inventory/hosts playbooks/site.yml -e
logic_operation=cluster_day_1 -e service=cluster_day_0 -vvvv --ask-vault
-pass <your_vault_password>
```

- Appliquer les paramètres à l'échelle du cluster :

```
ansible-playbook -i playbooks/inventory/hosts playbooks/site.yml -e
logic_operation=cluster_wide_settings -e service=cluster_wide_settings
-vvvv --ask-vault-pass <your_vault_password>
```

- Exécuter des vérifications de l'état :

```
ansible-playbook -i playbooks/inventory/hosts playbooks/site.yml -e
logic_operation=health_checks -e service=health_checks -e
enable_health_reports=true -vvvv --ask-vault-pass <your_vault_password>
```

Personnalisez la solution ONTAP Day 0/1

Pour personnaliser la solution ONTAP Day 0/1 en fonction de vos exigences, vous pouvez ajouter ou modifier des rôles Ansible.

Les rôles représentent les microservices dans le framework Ansible. Chaque microservice effectue une opération. Par exemple, ONTAP Day 0 est un service qui contient plusieurs microservices.

Ajoutez des rôles Ansible

Vous pouvez ajouter des rôles Ansible afin de personnaliser la solution en fonction de votre environnement. Les rôles requis sont définis par des définitions de service dans le framework Ansible.

Un rôle doit répondre aux exigences suivantes pour être utilisé en tant que microservice :

- Acceptez une liste d'arguments dans la `args` variable.
- Utilisez la structure Ansible « bloc, sauvetage, toujours » en fonction des exigences de chaque bloc.
- Utilisez un seul module Ansible et définissez une seule tâche dans le bloc.
- Mettre en œuvre tous les paramètres de module disponibles conformément aux exigences détaillées dans cette section.

Structure de microservice requise

Chaque rôle doit prendre en charge les variables suivantes :

- `mode`: Si le mode est défini sur `test` le rôle tente d'importer le `test.yml` qui indique ce que le rôle fait sans l'exécuter réellement.

  > Il n'est pas toujours possible de le mettre en œuvre en raison de certaines interdépendances.

- `status`: L'état global de l'exécution du PlayBook. Si la valeur n'est pas définie sur `success` le rôle n'est pas exécuté.
- `args` : Une liste de dictionnaires spécifiques aux rôles avec des clés correspondant aux noms des paramètres de rôle.
- `global_log_messages`: Collecte des messages de journal pendant l'exécution du PlayBook. Une entrée est générée chaque fois que le rôle est exécuté.
- `log_name`: Le nom utilisé pour faire référence au rôle dans les `global_log_messages` entrées.
- `task_descr`: Une brève description de ce que fait le rôle.
- `service_start_time`: Horodatage utilisé pour suivre l'heure d'exécution de chaque rôle.
- `playbook_status`: Le statut du PlayBook Ansible.
- `role_result`: La variable qui contient la sortie de rôle et est incluse dans chaque message dans les `global_log_messages` entrées.

Exemple de structure de rôle

L'exemple suivant fournit la structure de base d'un rôle qui implémente un microservice. Dans cet exemple, vous devez modifier les variables de votre configuration.

Montrer l'exemple

Structure des rôles de base :

```
- name:  Set some role attributes
  set_fact:
    log_name:     "<LOG_NAME>"
    task_descr:   "<TASK_DESCRIPTION>"

-  name: "{{ log_name }}"
   block:
      -  set_fact:
            service_start_time: "{{ lookup('pipe', 'date
+%Y%m%d%H%M%S') }}"

      -  name: "Provision the new user"
         <MODULE_NAME>:

#-----------------------------------------------------------------
            # COMMON ATTRIBUTES

#-----------------------------------------------------------------
            hostname:            "{{
clusters[loop_arg['hostname']]['mgmt_ip'] }}"
            username:            "{{
clusters[loop_arg['hostname']]['username'] }}"
            password:            "{{
clusters[loop_arg['hostname']]['password'] }}"

            cert_filepath:       "{{ loop_arg['cert_filepath']
| default(omit) }}"
            feature_flags:       "{{ loop_arg['feature_flags']
| default(omit) }}"
            http_port:           "{{ loop_arg['http_port']
| default(omit) }}"
            https:               "{{ loop_arg['https']
| default('true') }}"
            ontapi:              "{{ loop_arg['ontapi']
| default(omit) }}"
            key_filepath:        "{{ loop_arg['key_filepath']
| default(omit) }}"
            use_rest:            "{{ loop_arg['use_rest']
| default(omit) }}"
            validate_certs:      "{{ loop_arg['validate_certs']
| default('false') }}"
```

```
            <MODULE_SPECIFIC_PARAMETERS>

#-------------------------------------------------------------
            # REQUIRED ATTRIBUTES

#-------------------------------------------------------------
            required_parameter:     "{{ loop_arg['required_parameter']
}}"

#-------------------------------------------------------------
            # ATTRIBUTES w/ DEFAULTS

#-------------------------------------------------------------
            defaulted_parameter:    "{{ loop_arg['defaulted_parameter']
| default('default_value') }}"

#-------------------------------------------------------------
            # OPTIONAL ATTRIBUTES

#-------------------------------------------------------------
            optional_parameter:     "{{ loop_arg['optional_parameter']
| default(omit) }}"
         loop:    "{{ args }}"
         loop_control:
            loop_var:   loop_arg
         register:   role_result

   rescue:
      -  name: Set role status to FAIL
         set_fact:
            playbook_status:   "failed"

   always:
      -  name: add log msg
         vars:
            role_log:
               role: "{{ log_name }}"
               timestamp:
                  start_time: "{{service_start_time}}"
                  end_time: "{{ lookup('pipe', 'date +%Y-%m-
%d@%H:%M:%S') }}"
               service_status: "{{ playbook_status }}"
               result: "{{role_result}}"
         set_fact:
            global_log_msgs:   "{{ global_log_msgs + [ role_log ] }}"
```

Variables utilisées dans l'exemple de rôle :

- `<NAME>`: Une valeur remplaçable qui doit être fournie pour chaque microservice.
- `<LOG_NAME>`: Le nom de forme court du rôle utilisé à des fins de journalisation. Par exemple `ONTAP_VOLUME`, .
- `<TASK_DESCRIPTION>`: Une brève description de ce que fait le microservice.
- `<MODULE_NAME>`: Nom du module Ansible pour la tâche.

> Le PlayBook de niveau supérieur `execute.yml` spécifie la `netapp.ontap` collection. Si le module fait partie de la `netapp.ontap` collection, il n'est pas nécessaire de spécifier entièrement le nom du module.

- `<MODULE_SPECIFIC_PARAMETERS>`: Paramètres du module Ansible spécifiques au module utilisé pour implémenter le microservice. La liste suivante décrit les types de paramètres et leur mode de regroupement.
  - Paramètres requis : tous les paramètres requis sont spécifiés sans valeur par défaut.
  - Paramètres ayant une valeur par défaut spécifique au microservice (différente d'une valeur par défaut spécifiée par la documentation du module).
  - Tous les autres paramètres utilisent `default(omit)` comme valeur par défaut.

Utilisation de dictionnaires multi-niveaux comme paramètres de module

Certains modules Ansible fournis par NetApp utilisent des dictionnaires multiniveaux pour les paramètres de module (par exemple, des groupes de règles de QoS fixes et adaptatifs).

L'utilisation `default(omit)` seule ne fonctionne pas lorsque ces dictionnaires sont utilisés, surtout lorsqu'il y en a plus d'un et qu'ils s'excluent mutuellement.

Si vous devez utiliser des dictionnaires multiniveaux comme paramètres de module, vous devez diviser la fonctionnalité en plusieurs microservices (rôles) de sorte que chacun d'entre eux puisse fournir au moins une valeur de dictionnaire de second niveau pour le dictionnaire approprié.

Les exemples suivants présentent les groupes de règles de QoS fixes et adaptatifs répartis sur deux microservices.

Le premier microservice contient des valeurs de groupe de règles QoS fixes :

```
fixed_qos_options:
  capacity_shared:           "{{
loop_arg['fixed_qos_options']['capacity_shared']         | default(omit)
}}"
  max_throughput_iops:       "{{
loop_arg['fixed_qos_options']['max_throughput_iops']     | default(omit)
}}"
  min_throughput_iops:       "{{
loop_arg['fixed_qos_options']['min_throughput_iops']     | default(omit)
}}"
  max_throughput_mbps:       "{{
loop_arg['fixed_qos_options']['max_throughput_mbps']     | default(omit)
}}"
  min_throughput_mbps:       "{{
loop_arg['fixed_qos_options']['min_throughput_mbps']     | default(omit)
}}"
```

Le second microservice contient les valeurs des groupes de règles de QoS adaptative :

```
adaptive_qos_options:
  absolute_min_iops:         "{{
loop_arg['adaptive_qos_options']['absolute_min_iops'] | default(omit) }}"
  expected_iops:             "{{
loop_arg['adaptive_qos_options']['expected_iops']     | default(omit) }}"
  peak_iops:                 "{{
loop_arg['adaptive_qos_options']['peak_iops']         | default(omit) }}"
```