NetApp Console Automatisierungszentrale

NetApp Automation

NetApp
November 18, 2025

Inhalt

NetApp Console Automatisierungszentrale

Überblick über den NetApp Console Automatisierungshub

Die NetApp Console ist ein Automatisierungs-Hub, der eine Sammlung von Automatisierungslösungen für NetApp Kunden, -Partner und -Mitarbeiter bereitstellt. Der Automatisierungs-Hub bietet zahlreiche Funktionen und Vorteile.

Ein Ort für Ihre Automatisierungsanforderungen

Sie können auf die "NetApp Console Automatisierungszentrale" über die webbasierte Benutzeroberfläche der Konsole. Hier finden Sie alle Skripte, Playbooks und Module, die Sie benötigen, um die Automatisierung und den Betrieb Ihrer NetApp -Produkte und -Dienste zu optimieren.

Lösungen werden von NetApp entwickelt und getestet

Alle Automatisierungslösungen und Skripte wurden von NetApp erstellt und getestet. Jede Lösung richtet sich an einen bestimmten Kunden-Anwendungsfall oder eine bestimmte Anfrage. Der Schwerpunkt liegt hauptsächlich auf der Integration in NetApp File- und Datenservices.

Dokumentation

Zu jeder Automatisierungslösung gehört eine zugehörige Dokumentation, die Ihnen den Einstieg erleichtert. Während auf die Lösungen über die Weboberfläche der Konsole zugegriffen werden kann, ist die gesamte Dokumentation auf dieser Site verfügbar. Die Dokumentation ist nach den NetApp -Produkten und Cloud-Diensten organisiert.

Solide Grundlage für die Zukunft

NetApp hat sich zum Ziel gesetzt, seine Kunden bei der Verbesserung und Optimierung der Automatisierung ihrer Rechenzentren und Cloud-Umgebungen zu unterstützen. Wir gehen davon aus, dass wir die Automatisierungsplattform Console kontinuierlich verbessern werden, um Kundenanforderungen, technologischen Veränderungen und der fortlaufenden Produktintegration gerecht zu werden.

Wir möchten von Ihnen hören

Das NetApp Customer Experience Office (CXO) Automation Team würde gerne von Ihnen hören. Wenn Sie Feedback, Probleme oder Funktionsanforderungen haben, senden Sie bitte eine E-Mail an:ng-cxo-Automation-Admins@NetApp.com[CXO-Automatisierungsteam].

Amazon FSx for NetApp ONTAP -Verwaltung

Amazon FSx for NetApp ONTAP Management – Burst to Cloud

Mit dieser Automatisierungslösung können Sie Amazon FSx for NetApp ONTAP -Verwaltung mit Volumes und einem zugehörigen FlexCache bereitstellen.

Amazon FSx for NetApp ONTAP Management wird auch als **FSx für ONTAP** bezeichnet.

Über diese Lösung sprechen

Der mit dieser Lösung bereitgestellte Automatisierungscode führt im allgemeinen die folgenden Aktionen durch:

- Bereitstellen eines Ziel-FSX für ONTAP-Dateisystem

- Storage Virtual Machines (SVMs) für das Filesystem bereitstellen
- Cluster-Peering-Beziehung zwischen den Quell- und Zielsystemen erstellen
- SVM-Peering-Beziehung zwischen dem Quellsystem und dem Zielsystem für FlexCache erstellen
- Optional können Sie FlexVol Volumes mithilfe von FSX für ONTAP erstellen
- Erstellen Sie ein FlexCache-Volume in FSX für ONTAP mit der Quelle, die auf On-Premises-Storage verweist

Die Automatisierung basiert auf Docker und Docker Compose, die wie unten beschrieben auf der virtuellen Linux-Maschine installiert werden müssen.

Bevor Sie beginnen

Sie müssen über Folgendes verfügen, um die Bereitstellung und Konfiguration abzuschließen:

- Sie müssen die "Amazon FSx for NetApp ONTAP Management – Burst to Cloud" Automatisierungslösung über die NetApp Console Web-Benutzeroberfläche. Die Lösung ist als Datei verpackt. `AWS_FSxN_BTC.zip`Die
- Netzwerk-Konnektivität zwischen Quell- und Zielsystemen
- Eine Linux-VM mit den folgenden Eigenschaften:
  - Debian-basierte Linux-Distribution
  - Implementierung mit derselben VPC-Untermenge, die für FSX für die ONTAP-Bereitstellung verwendet wurde
- Konto bei AWS.

Schritt: Installieren und konfigurieren Sie Docker

Installieren und konfigurieren Sie Docker auf einer Debian-basierten virtuellen Linux-Maschine.

Schritte

1. Bereiten Sie die Umgebung vor.

```
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. Installieren Sie Docker und überprüfen Sie die Installation.

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

3. Fügen Sie die erforderliche Linux-Gruppe einem zugeordneten Benutzer hinzu.

Prüfen Sie zunächst, ob die Gruppe **Docker** in Ihrem Linux-System existiert. Wenn dies nicht der Fall ist, erstellen Sie die Gruppe und fügen Sie den Benutzer hinzu. Standardmäßig wird der aktuelle Shell-Benutzer der Gruppe hinzugefügt.

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

4. Aktivieren Sie die neuen Gruppen- und Benutzerdefinitionen

   Wenn Sie eine neue Gruppe mit einem Benutzer erstellt haben, müssen Sie die Definitionen aktivieren. Dazu können Sie sich von Linux abmelden und dann wieder in. Oder Sie können den folgenden Befehl ausführen.

```
newgrp docker
```

Schritt 2: Installieren Sie Docker Compose

Installieren Sie Docker Compose auf einer Debian-basierten virtuellen Linux-Maschine.

Schritte

1. Installieren Sie 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. Überprüfen Sie, ob die Installation erfolgreich war.

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

Schritt 3: Vorbereiten des Docker Images

Sie müssen das mit der Automatisierungslösung bereitgestellte Docker-Image extrahieren und laden.

Schritte

1. Kopieren Sie die Lösungsdatei `AWS_FSxN_BTC.zip` auf die virtuelle Maschine, auf der der Automatisierungscode ausgeführt wird.

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

   Der Eingabeparameter `private-key.pem` ist Ihre private Schlüsseldatei, die für die Authentifizierung der AWS Virtual Machine (EC2-Instanz) verwendet wird.

2. Navigieren Sie zum richtigen Ordner mit der Lösungsdatei, und entpacken Sie die Datei.

```
unzip AWS_FSxN_BTC.zip
```

3. Navigieren Sie zu dem neuen Ordner `AWS_FSxN_BTC`, der mit dem Entpacken erstellt wurde, und führen Sie die Dateien auf. Sie sollten die Datei sehen `aws_fsxn_flexcache_image_latest.tar.gz`.

```
ls -la
```

4. Laden Sie die Docker-Image-Datei. Der Ladevorgang sollte in der Regel in wenigen Sekunden abgeschlossen sein.

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

5. Bestätigen Sie, dass das Docker-Image geladen ist.

```
docker images
```

Sie sollten das Docker Image mit dem Tag `latest` sehen `aws_fsxn_flexcache_image`.

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

Schritt 4: Umgebungsdatei für AWS Zugangsdaten erstellen

Sie müssen eine lokale Variablendatei für die Authentifizierung mit dem Zugriff und dem geheimen Schlüssel erstellen. Fügen Sie dann die Datei der `.env` Datei hinzu.

Schritte

1. Erstellen Sie die `awsauth.env` Datei an folgendem Speicherort:

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

2. Fügen Sie der Datei folgenden Inhalt hinzu:

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

   Das Format **muss** genau wie oben dargestellt sein, ohne Leerzeichen zwischen `key` und `value`.

3. Fügen Sie den absoluten Dateipfad mithilfe der Variablen zur Datei `AWS_CREDS` hinzu `.env`. Beispiel:

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

Schritt 5: Erstellen Sie ein externes Volume

Sie benötigen ein externes Volume, um sicherzustellen, dass die Terraform-Statusdateien und andere wichtige Dateien persistent sind. Diese Dateien müssen für Terraform verfügbar sein, um den Workflow und die Implementierungen auszuführen.

Schritte

1. Erstellen Sie ein externes Volume außerhalb von Docker Compose.

   Stellen Sie sicher, dass Sie den Volume-Namen (letzten Parameter) auf den entsprechenden Wert aktualisieren, bevor Sie den Befehl ausführen.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Fügen Sie den Pfad zum externen Volume zur Umgebungsdatei mit dem folgenden Befehl hinzu `.env`:

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

   Denken Sie daran, den vorhandenen Dateiinhalt und die Doppelpunkt-Formatierung beizubehalten. Beispiel:

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

   Stattdessen können Sie eine NFS-Freigabe mit einem Befehl wie dem folgenden als externes Volume hinzufügen:

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

3. Aktualisieren Sie die Terraform-Variablen.
   a. Navigieren Sie zum Ordner `aws_fsxn_variables`.
   b. Bestätigen Sie, dass die folgenden beiden Dateien vorhanden sind: `terraform.tfvars` Und `variables.tf`.
   c. Aktualisieren Sie die Werte in `terraform.tfvars`, wie für Ihre Umgebung erforderlich.

      Weitere Informationen finden Sie unter "Terraform-Ressource: aws_fsx_ONTAP_File_System" .

Schritt 6: Bereitstellen von Amazon FSx for NetApp ONTAP Management und FlexCache

Sie können Amazon FSx for NetApp ONTAP Management und FlexCache bereitstellen.

Schritte

1. Navigieren Sie zum Ordner root (AWS_FSXN_BTC), und geben Sie den Provisionierungsbefehl aus.

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

Mit diesem Befehl werden zwei Container erstellt. Der erste Container implementiert FSX for ONTAP, der zweite Container erstellt Cluster-Peering, SVM-Peering, Ziel-Volume und FlexCache.

2. Monitoring des Bereitstellungsprozesses

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

Dieser Befehl gibt Ihnen die Ausgabe in Echtzeit, wurde aber so konfiguriert, dass die Protokolle durch die Datei erfasst `deployment.log` werden. Sie können den Namen dieser Protokolldateien ändern, indem Sie die Datei bearbeiten `.env` und die Variablen aktualisieren `DEPLOYMENT_LOGS`.

Schritt 7: Zerstören Sie Amazon FSx for NetApp ONTAP Management und FlexCache

Sie können Amazon FSx for NetApp ONTAP Management und FlexCache optional löschen und entfernen.

1. Setzen Sie die Variable `flexcache_operation` in der `terraform.tfvars` Datei auf "Destroy".
2. Navigieren Sie zum Ordner root (AWS_FSXN_BTC), und geben Sie den folgenden Befehl ein.

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

Mit diesem Befehl werden zwei Container erstellt. Der erste Container löscht FlexCache und der zweite Container löscht FSX für ONTAP.

3. Monitoring des Bereitstellungsprozesses

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

Amazon FSx for NetApp ONTAP Management – Notfallwiederherstellung

Mit dieser Automatisierungslösung können Sie mithilfe von Amazon FSx for NetApp ONTAP Verwaltung eine Notfallwiederherstellungssicherung eines Quellsystems erstellen.

Amazon FSx for NetApp ONTAP Management wird auch als **FSx für ONTAP** bezeichnet.

Über diese Lösung sprechen

Der mit dieser Lösung bereitgestellte Automatisierungscode führt im allgemeinen die folgenden Aktionen durch:

- Bereitstellen eines Ziel-FSX für ONTAP-Dateisystem
- Storage Virtual Machines (SVMs) für das Filesystem bereitstellen
- Cluster-Peering-Beziehung zwischen den Quell- und Zielsystemen erstellen
- SVM-Peering-Beziehung zwischen dem Quellsystem und dem Zielsystem für SnapMirror erstellen
- Erstellung von Ziel-Volumes

- Erstellen Sie eine SnapMirror-Beziehung zwischen den Quell- und Ziel-Volumes
- Starten Sie den SnapMirror-Transfer zwischen den Quell- und Ziel-Volumes

Die Automatisierung basiert auf Docker und Docker Compose, die wie unten beschrieben auf der virtuellen Linux-Maschine installiert werden müssen.

Bevor Sie beginnen

Sie müssen über Folgendes verfügen, um die Bereitstellung und Konfiguration abzuschließen:

- Sie müssen die "Amazon FSx for NetApp ONTAP Management – Notfallwiederherstellung" Automatisierungslösung über die NetApp Console Web-Benutzeroberfläche. Die Lösung ist verpackt als `FSxN_DR.zip`Die Diese ZIP-Datei enthält die `AWS_FSxN_Bck_Prov.zip` Datei, die Sie zum Bereitstellen der in diesem Dokument beschriebenen Lösung verwenden werden.
- Netzwerk-Konnektivität zwischen Quell- und Zielsystemen
- Eine Linux-VM mit den folgenden Eigenschaften:
  - Debian-basierte Linux-Distribution
  - Implementierung mit derselben VPC-Untermenge, die für FSX für die ONTAP-Bereitstellung verwendet wurde
- Ein AWS-Konto.

Schritt: Installieren und konfigurieren Sie Docker

Installieren und konfigurieren Sie Docker auf einer Debian-basierten virtuellen Linux-Maschine.

Schritte

1. Bereiten Sie die Umgebung vor.

```
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. Installieren Sie Docker und überprüfen Sie die Installation.

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

3. Fügen Sie die erforderliche Linux-Gruppe einem zugeordneten Benutzer hinzu.

   Prüfen Sie zunächst, ob die Gruppe **Docker** in Ihrem Linux-System existiert. Wenn sie nicht vorhanden ist, erstellen Sie die Gruppe und fügen Sie den Benutzer hinzu. Standardmäßig wird der aktuelle Shell-Benutzer der Gruppe hinzugefügt.

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

4. Aktivieren Sie die neuen Gruppen- und Benutzerdefinitionen

   Wenn Sie eine neue Gruppe mit einem Benutzer erstellt haben, müssen Sie die Definitionen aktivieren. Dazu können Sie sich von Linux abmelden und dann wieder in. Oder Sie können den folgenden Befehl ausführen.

```
newgrp docker
```

Schritt 2: Installieren Sie Docker Compose

Installieren Sie Docker Compose auf einer Debian-basierten virtuellen Linux-Maschine.

Schritte

1. Installieren Sie 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. Überprüfen Sie, ob die Installation erfolgreich war.

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

Schritt 3: Vorbereiten des Docker Images

Sie müssen das mit der Automatisierungslösung bereitgestellte Docker-Image extrahieren und laden.

Schritte

1. Kopieren Sie die Lösungsdatei `AWS_FSxN_Bck_Prov.zip` auf die virtuelle Maschine, auf der der Automatisierungscode ausgeführt wird.

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

   Der Eingabeparameter `private-key.pem` ist Ihre private Schlüsseldatei, die für die Authentifizierung der AWS Virtual Machine (EC2-Instanz) verwendet wird.

2. Navigieren Sie zum richtigen Ordner mit der Lösungsdatei, und entpacken Sie die Datei.

```
unzip AWS_FSxN_Bck_Prov.zip
```

3. Navigieren Sie zu dem neuen Ordner `AWS_FSxN_Bck_Prov`, der mit dem Entpacken erstellt wurde, und führen Sie die Dateien auf. Sie sollten die Datei sehen `aws_fsxn_bck_image_latest.tar.gz`.

```
ls -la
```

4. Laden Sie die Docker-Image-Datei. Der Ladevorgang sollte in der Regel in wenigen Sekunden abgeschlossen sein.

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

5. Bestätigen Sie, dass das Docker-Image geladen ist.

```
docker images
```

Sie sollten das Docker Image mit dem Tag `latest` sehen `aws_fsxn_bck_image`.

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

Schritt 4: Umgebungsdatei für AWS Zugangsdaten erstellen

Sie müssen eine lokale Variablendatei für die Authentifizierung mit dem Zugriff und dem geheimen Schlüssel erstellen. Fügen Sie dann die Datei der `.env` Datei hinzu.

Schritte

1. Erstellen Sie die `awsauth.env` Datei an folgendem Speicherort:

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

2. Fügen Sie der Datei folgenden Inhalt hinzu:

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

   Das Format **muss** genau wie oben dargestellt sein, ohne Leerzeichen zwischen `key` und `value`.

3. Fügen Sie den absoluten Dateipfad mithilfe der Variablen zur Datei `AWS_CREDS` hinzu `.env`. Beispiel:

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

Schritt 5: Erstellen Sie ein externes Volume

Sie benötigen ein externes Volume, um sicherzustellen, dass die Terraform-Statusdateien und andere wichtige Dateien persistent sind. Diese Dateien müssen für Terraform verfügbar sein, um den Workflow und die Implementierungen auszuführen.

Schritte

1. Erstellen Sie ein externes Volume außerhalb von Docker Compose.

   Stellen Sie sicher, dass Sie den Volume-Namen (letzten Parameter) auf den entsprechenden Wert aktualisieren, bevor Sie den Befehl ausführen.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Fügen Sie den Pfad zum externen Volume zur Umgebungsdatei mit dem folgenden Befehl hinzu `.env`:

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

   Denken Sie daran, den vorhandenen Dateiinhalt und die Doppelpunkt-Formatierung beizubehalten. Beispiel:

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

   Stattdessen können Sie eine NFS-Freigabe mit einem Befehl wie dem folgenden als externes Volume hinzufügen:

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

3. Aktualisieren Sie die Terraform-Variablen.
   a. Navigieren Sie zum Ordner `aws_fsxn_variables`.
   b. Bestätigen Sie, dass die folgenden beiden Dateien vorhanden sind: `terraform.tfvars` Und `variables.tf`.
   c. Aktualisieren Sie die Werte in `terraform.tfvars`, wie für Ihre Umgebung erforderlich.

      Weitere Informationen finden Sie unter "Terraform-Ressource: aws_fsx_ONTAP_File_System" .

Schritt 6: Bereitstellung der Backup-Lösung

Sie können die Disaster Recovery Backup-Lösung implementieren und bereitstellen.

Schritte

1. Navigieren Sie zum Ordner root (AWS_FSxN_BCK_Prov), und geben Sie den Befehl Provisioning aus.

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

   Mit diesem Befehl werden drei Container erstellt. Der erste Container implementiert FSX für ONTAP. Der zweite Container erstellt Cluster-Peering, SVM-Peering und Ziel-Volume. Der dritte Container erstellt die

SnapMirror-Beziehung und initiiert den SnapMirror-Transfer.

2. Monitoring des Bereitstellungsprozesses

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

Dieser Befehl gibt Ihnen die Ausgabe in Echtzeit, wurde aber so konfiguriert, dass die Protokolle durch die Datei erfasst `deployment.log` werden. Sie können den Namen dieser Protokolldateien ändern, indem Sie die Datei bearbeiten `.env` und die Variablen aktualisieren `DEPLOYMENT_LOGS`.

Azure NetApp Dateien

Installieren Sie Oracle mit Azure NetApp Files

Mit dieser Automatisierungslösung können Sie Azure NetApp Files Volumes bereitstellen und Oracle auf einer verfügbaren Virtual Machine installieren. Anschließend verwendet Oracle die Volumes für die Datenspeicherung.

Über diese Lösung sprechen

Der mit dieser Lösung bereitgestellte Automatisierungscode führt im allgemeinen die folgenden Aktionen durch:

- Richten Sie ein NetApp-Konto auf Azure ein
- Richten Sie auf Azure einen Storage-Kapazitäts-Pool ein
- Provisionierung der Azure NetApp Files Volumes basierend auf der Definition
- Erstellen Sie die Mount-Punkte
- Mounten Sie die Azure NetApp Files Volumes an den Bereitstellungspunkten
- Installieren Sie Oracle auf dem Linux-Server
- Erstellen Sie die Listeners und die Datenbank
- Erstellen der steckbaren Datenbanken (PDBs)
- Starten Sie den Listener und die Oracle-Instanz
- Installieren und konfigurieren Sie das `azacsnap` Dienstprogramm, um einen Snapshot zu erstellen

Bevor Sie beginnen

Sie müssen über Folgendes verfügen, um die Installation abzuschließen:

- Sie müssen die "Oracle mit Azure NetApp Files" Automatisierungslösung über die NetApp Console Web-Benutzeroberfläche. Die Lösung ist als Datei verpackt. `na_oracle19c_deploy-master.zip`Die
- Eine Linux-VM mit den folgenden Eigenschaften:
  - RHEL 8 (Standard_D8S_v3-RHEL-8)
  - Wird auf demselben virtuellen Azure Netzwerk bereitgestellt, das auch für die Azure NetApp Files-Bereitstellung verwendet wird
- Ein Azure-Konto

Die Automatisierungslösung wird als Image bereitgestellt und mit Docker und Docker Compose ausgeführt. Sie müssen beide auf der virtuellen Linux-Maschine installieren, wie unten beschrieben.

Sie sollten die VM auch mit dem Befehl bei RedHat registrieren `sudo subscription-manager register`. Der Befehl fordert Sie zur Eingabe Ihrer Kontoanmeldeinformationen auf. Bei Bedarf können Sie ein Konto bei https://developers.redhat.com/. erstellen

Schritt: Installieren und konfigurieren Sie Docker

Installation und Konfiguration von Docker auf einer virtuellen RHEL 8 Linux-Maschine

Schritte

1. Installieren Sie die Docker-Software mithilfe der folgenden Befehle.

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

2. Starten Sie Docker und zeigen Sie die Version an, um zu bestätigen, dass die Installation erfolgreich war.

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

3. Fügen Sie die erforderliche Linux-Gruppe einem zugeordneten Benutzer hinzu.

   Prüfen Sie zunächst, ob die Gruppe **Docker** in Ihrem Linux-System existiert. Wenn dies nicht der Fall ist, erstellen Sie die Gruppe und fügen Sie den Benutzer hinzu. Standardmäßig wird der aktuelle Shell-Benutzer der Gruppe hinzugefügt.

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

4. Aktivieren Sie die neuen Gruppen- und Benutzerdefinitionen

   Wenn Sie eine neue Gruppe mit einem Benutzer erstellt haben, müssen Sie die Definitionen aktivieren. Dazu können Sie sich von Linux abmelden und dann wieder in. Oder Sie können den folgenden Befehl ausführen.

```
newgrp docker
```

Schritt 2: Installieren Sie Docker Compose und die NFS-Dienstprogramme

Installieren und konfigurieren Sie Docker Compose zusammen mit dem NFS-Dienstprogramme-Paket.

Schritte

1. Installieren Sie Docker Compose, und zeigen Sie die Version an, um zu bestätigen, dass die Installation erfolgreich war.

```
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. Installieren Sie das NFS Utilities-Paket.

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

Schritt 3: Laden Sie die Oracle Installationsdateien herunter

Laden Sie die erforderlichen Oracle-Installations- und Patch-Dateien sowie das Dienstprogramm herunter `azacsnap`.

Schritte

1. Melden Sie sich bei Bedarf bei Ihrem Oracle Konto an.
2. Laden Sie die folgenden Dateien herunter.

Datei	Beschreibung
LINUX.X64_193000_db_home.zip	19.3 Basisinstallateur
p31281355_190000_Linux-x86-64.zip	19.8-HE-Patch
p6880880_190000_Linux-x86-64.zip	opatch-Version 12.2.0.1.23
azacsnap_installer_v5.0.run	Azacsnap-Installationsprogramm

3. Legen Sie alle Installationsdateien in den Ordner `/tmp/archive`.
4. Stellen Sie sicher, dass alle Benutzer auf dem Datenbankserver vollen Zugriff (Lesen, Schreiben, Ausführen) auf den Ordner haben `/tmp/archive`.

Schritt 4: Vorbereiten des Docker Images

Sie müssen das mit der Automatisierungslösung bereitgestellte Docker-Image extrahieren und laden.

Schritte

1. Kopieren Sie die Lösungsdatei `na_oracle19c_deploy-master.zip` auf die virtuelle Maschine, auf der der Automatisierungscode ausgeführt wird.

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

Der Eingabeparameter `private-key.pem` ist Ihre private Schlüsseldatei, die für die Authentifizierung der virtuellen Azure-Maschinen verwendet wird.

2. Navigieren Sie zum richtigen Ordner mit der Lösungsdatei, und entpacken Sie die Datei.

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

3. Navigieren Sie zu dem neuen Ordner `na_oracle19c_deploy-master`, der mit dem Entpacken erstellt wurde, und führen Sie die Dateien auf. Sie sollten die Datei sehen `ora_anf_bck_image.tar`.

```
ls -lt
```

4. Laden Sie die Docker-Image-Datei. Der Ladevorgang sollte in der Regel in wenigen Sekunden abgeschlossen sein.

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

5. Bestätigen Sie, dass das Docker-Image geladen ist.

```
docker images
```

Sie sollten das Docker Image mit dem Tag `latest` sehen `ora_anf_bck_image`.

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

Schritt 5: Erstellen Sie ein externes Volume

Sie benötigen ein externes Volume, um sicherzustellen, dass die Terraform-Statusdateien und andere wichtige Dateien persistent sind. Diese Dateien müssen für Terraform verfügbar sein, um den Workflow und die Implementierungen auszuführen.

Schritte

1. Erstellen Sie ein externes Volume außerhalb von Docker Compose.

   Stellen Sie sicher, dass Sie den Volume-Namen aktualisieren, bevor Sie den Befehl ausführen.

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

2. Fügen Sie den Pfad zum externen Volume zur Umgebungsdatei mit dem folgenden Befehl hinzu `.env`:

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

Denken Sie daran, den vorhandenen Dateiinhalt und die Doppelpunkt-Formatierung beizubehalten. Beispiel:

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

3. Aktualisieren Sie die Terraform-Variablen.
   a. Navigieren Sie zum Ordner `ora_anf_variables`.
   b. Bestätigen Sie, dass die folgenden beiden Dateien vorhanden sind: `terraform.tfvars` Und `variables.tf`.
   c. Aktualisieren Sie die Werte in `terraform.tfvars`, wie für Ihre Umgebung erforderlich.

Schritt 6: Installieren Sie Oracle

Sie können jetzt Oracle bereitstellen und installieren.

Schritte

1. Installieren Sie Oracle mithilfe der folgenden Befehlssequenz.

```
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. Laden Sie Ihre Bash-Variablen neu und bestätigen Sie, indem Sie den Wert für anzeigen `ORACLE_HOME`.
   a. `cd /home/oracle`
   b. `source .bash_profile`
   c. `echo $ORACLE_HOME`
3. Sie sollten sich bei Oracle anmelden können.

```
sudo su oracle
```

Schritt 7: Validierung der Oracle-Installation

Sie sollten bestätigen, dass die Oracle-Installation erfolgreich war.

Schritte

1. Melden Sie sich beim Linux Oracle-Server an, und zeigen Sie eine Liste der Oracle-Prozesse an. Damit wird bestätigt, dass die Installation wie erwartet abgeschlossen wurde und die Oracle-Datenbank ausgeführt wird.

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

2. Melden Sie sich bei der Datenbank an, um die Datenbankkonfiguration zu überprüfen und zu bestätigen, dass die PDBs ordnungsgemäß erstellt wurden.

```
sqlplus / as sysdba
```

Sie sollten eine Ausgabe wie die folgende sehen:

```
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. Führen Sie ein paar einfache SQL-Befehle aus, um zu bestätigen, dass die Datenbank verfügbar ist.

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

Schritt 8: Installieren Sie das Dienstprogramm azacsnap und führen Sie ein Snapshot-Backup durch

Sie müssen das Dienstprogramm installieren und ausführen `azacsnap`, um ein Snapshot-Backup durchzuführen.

Schritte

1. Den Behälter einbauen.

```
docker-compose up azacsnap_install
```

2. Wechseln Sie zum Snapshot-Benutzerkonto.

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

3. Konfigurieren einer Speicherdetaildatei. Dadurch wird die Konfigurationsdatei erstellt `azacsnap.json`.

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

4. Führen Sie ein Snapshot-Backup durch.

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

Schritt 9: Optional Migration einer lokalen PDB in die Cloud

Optional können Sie die lokale PDB in die Cloud migrieren.

Schritte

1. Legen Sie die Variablen in den Dateien nach Bedarf für Ihre Umgebung fest `tfvars`.
2. Migrieren Sie die PDB.

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

Cloud Volumes ONTAP für AWS

Cloud Volumes ONTAP für AWS – Burst in die Cloud

Dieser Artikel unterstützt die NetApp Cloud Volumes ONTAP for AWS Automation Solution, die NetApp -Kunden über den NetApp Console Automation Hub zur Verfügung steht.

Die Automatisierungslösung Cloud Volumes ONTAP für AWS automatisiert die Container-Implementierung von Cloud Volumes ONTAP für AWS mithilfe von Terraform, sodass Sie Cloud Volumes ONTAP für AWS schnell und ohne manuelles Eingreifen implementieren können.

Bevor Sie beginnen

- Sie müssen die "Cloud Volumes ONTAP AWS – Burst in die Cloud" Automatisierungslösung über die Web-Benutzeroberfläche der Konsole. Die Lösung ist verpackt als `cvo_aws_flexcache.zip`Die
- Sie müssen eine Linux-VM im gleichen Netzwerk wie Cloud Volumes ONTAP installieren.
- Nach der Installation der Linux-VM müssen Sie die Schritte in dieser Lösung befolgen, um die erforderlichen Abhängigkeiten zu installieren.

Schritt: Installieren Sie Docker und Docker Compose

Installation Von Docker

Die folgenden Schritte verwenden Ubuntu 20.04 Debian Linux-Distributionssoftware als Beispiel. Die Befehle, die Sie ausführen, hängen von der Linux-Distributionssoftware ab, die Sie verwenden. Informationen zur Konfiguration finden Sie in der Dokumentation der jeweiligen Linux-Distributionssoftware.

Schritte

1. Installieren Sie Docker, indem Sie die folgenden Befehle ausführen `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. Überprüfen Sie die Installation:

```
docker –version
```

3. Vergewissern Sie sich, dass auf Ihrem Linux-System eine Gruppe namens „Docker“ erstellt wurde. Erstellen Sie bei Bedarf die Gruppe:

```
sudo groupadd docker
```

4. Fügen Sie den Benutzer hinzu, der der Gruppe Zugriff auf Docker benötigt:

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

5. Ihre Änderungen werden übernommen, nachdem Sie sich beim Terminal abmelden und wieder anmelden. Alternativ können Sie die Änderungen sofort anwenden:

```
newgrp docker
```

Installieren Sie Docker Compose

Schritte

1. Installieren Sie Docker Compose, indem Sie die folgenden Befehle ausführen `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. Überprüfen Sie die Installation:

```
docker-compose –version
```

Schritt 2: Vorbereiten des Docker Images

Schritte

1. Kopieren Sie den `cvo_aws_flexcache.zip` Ordner auf die Linux-VM, die Sie zum Bereitstellen von Cloud Volumes ONTAP verwenden möchten:

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

   - `private-key.pem` Ist Ihre private Schlüsseldatei für die Anmeldung ohne Kennwort.
   - `awsuser` Ist der VM-Benutzername.
   - `IP_ADDRESS_OF_VM` Ist die VM IP-Adresse.
   - `LOCATION_TO_BE_COPIED` Ist der Speicherort, an den der Ordner kopiert werden soll.

2. Extrahieren Sie den `cvo_aws_flexcache.zip` Ordner. Sie können den Ordner im aktuellen Verzeichnis oder an einem benutzerdefinierten Speicherort extrahieren.

   Um den Ordner im aktuellen Verzeichnis zu extrahieren, führen Sie Folgendes aus:

```
unzip cvo_aws_flexcache.zip
```

   Um den Ordner an einem benutzerdefinierten Speicherort zu extrahieren, führen Sie Folgendes aus:

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

3. Navigieren Sie nach dem Extrahieren des Inhalts zum Ordner, `CVO_Aws_Deployment` und führen Sie den folgenden Befehl aus, um die Dateien anzuzeigen:

```
 ls -la
```

   Sie sollten eine Liste von Dateien sehen, ähnlich wie das folgende Beispiel:

```
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. Suchen Sie die `cvo_aws_flexcache_ubuntu_image.tar` Datei. Dieses enthält das Docker Image, das für die Implementierung von Cloud Volumes ONTAP für AWS erforderlich ist.
5. Enttaren Sie die Datei:

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

6. Warten Sie einige Minuten, bis das Docker-Image geladen ist, und überprüfen Sie dann, ob das Docker-Image erfolgreich geladen wurde:

```
docker images
```

Sie sollten ein Docker-Image mit dem `latest` Tag sehen `cvo_aws_flexcache_ubuntu_image`, wie im folgenden Beispiel gezeigt:

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

Bei Bedarf können Sie den Docker-Image-Namen ändern. Wenn Sie den Docker-Image-Namen ändern, müssen Sie den Docker-Image-Namen in den Dateien und `docker-compose-destroy` aktualisieren `docker-compose-deploy`.

Schritt 3: Erstellen Sie variable Umgebungsdateien

In dieser Phase müssen Sie zwei Umgebungsvariablendateien erstellen. Eine Datei dient der Authentifizierung von AWS Resource Manager-APIs mithilfe des AWS-Zugriffs und der geheimen Schlüssel. Die zweite Datei dient zum Festlegen von Umgebungsvariablen, damit die Terraform-Konsolenmodule AWS-APIs lokalisieren und authentifizieren können.

Schritte

1. Erstellen Sie die `awsauth.env` Datei an folgendem Speicherort:

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

   a. Fügen Sie der Datei folgenden Inhalt hinzu `awsauth.env`:

      Access_Key=<> secret_key=<>

   Das Format **muss** genau wie oben dargestellt sein.

2. Fügen Sie der Datei den absoluten Dateipfad hinzu `.env`.

   Geben Sie den absoluten Pfad für die Umgebungsdatei ein `awsauth.env`, die der Umgebungsvariable entspricht `AWS_CREDS`.

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

3. Navigieren Sie zu dem `cvo_aws_variable` Ordner, und aktualisieren Sie den Zugriffs- und Geheimschlüssel in der Datei mit den Anmeldeinformationen.

   Fügen Sie der Datei folgenden Inhalt hinzu:

   aws_Access_Key_id=<> aws_Secret_Access_Key=<>

   Das Format **muss** genau wie oben dargestellt sein.

Schritt 4: Registrieren Sie sich für NetApp Intelligent Services

Melden Sie sich über Ihren Cloud-Anbieter für NetApp Intelligent Services an und zahlen Sie stundenweise (PAYGO) oder über einen Jahresvertrag. Zu den intelligenten Diensten von NetApp gehören NetApp Backup und Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience und NetApp Disaster Recovery. Die NetApp Datenklassifizierung ist ohne zusätzliche Kosten in Ihrem Abonnement enthalten.

Schritte

1. Navigieren Sie im Amazon Web Services (AWS)-Portal zu **SaaS** und wählen Sie * NetApp Intelligent Services abonnieren*.

   Sie können entweder dieselbe Ressourcengruppe wie Cloud Volumes ONTAP oder eine andere Ressourcengruppe verwenden.

2. Konfigurieren Sie das NetApp Konsolenportal, um das SaaS-Abonnement in die Konsole zu importieren.

   Sie können dies direkt über das AWS-Portal konfigurieren.

   Sie werden zum Konsolenportal weitergeleitet, um die Konfiguration zu bestätigen.

3. Bestätigen Sie die Konfiguration im Konsolenportal, indem Sie **Speichern** auswählen.

Schritt 5: Erstellen Sie ein externes Volume

Sie sollten ein externes Volume erstellen, damit die Terraform-Statusdateien und andere wichtige Dateien erhalten bleiben. Sie müssen sicherstellen, dass die Dateien für Terraform verfügbar sind, um den Workflow und die Implementierungen auszuführen.

Schritte

1. Externes Volume außerhalb von Docker Compose erstellen:

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

Beispiel:

```
docker volume create cvo_aws_volume_dst
```

2. Verwenden Sie eine der folgenden Optionen:

   a. Fügen Sie einen externen Volume-Pfad zur Umgebungsdatei hinzu `.env`.

      Sie müssen das genaue unten dargestellte Format einhalten.

      Format:

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

      Beispiel:
      `PERSISTENT_VOL=cvo_aws_volume_dst:/cvo_aws`

   b. Fügen Sie NFS-Freigaben als externes Volume hinzu.

      Stellen Sie sicher, dass der Docker Container mit den NFS-Freigaben kommunizieren kann und dass die korrekten Berechtigungen wie Lese-/Schreibvorgänge konfiguriert sind.

      i. Fügen Sie den Pfad der NFS-Freigaben als Pfad zum externen Volume in der Docker Compose-Datei hinzu, wie unten gezeigt: Format:

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

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

3. Navigieren Sie zum `cvo_aws_variables` Ordner.

   Im Ordner sollte die folgende Variablendatei angezeigt werden:

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

4. Ändern Sie die Werte innerhalb der `terraform.tfvars` Datei entsprechend Ihren Anforderungen.

   Sie müssen die spezifische Begleitdokumentation lesen, wenn Sie einen der Variablenwerte in der Datei ändern `terraform.tfvars`. Die Werte können je nach Region, Verfügbarkeitszonen und anderen von Cloud Volumes ONTAP für AWS unterstützten Faktoren variieren. Dies umfasst Lizenzen, Festplattengröße und VM-Größe für einzelne Nodes sowie Hochverfügbarkeitspaare (HA).

   Alle unterstützenden Variablen für den Konsolenagenten und die Cloud Volumes ONTAP Terraform-Module

sind bereits in der `variables.tf` Datei. Sie müssen auf die Variablennamen in der `variables.tf` Datei vor dem Hinzufügen zur `terraform.tfvars` Datei.

5. Je nach Ihren Anforderungen können Sie FlexCache und FlexClone aktivieren oder deaktivieren, indem Sie die folgenden Optionen auf oder `false` einstellen `true`.

   Die folgenden Beispiele aktivieren FlexCache und FlexClone:

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

Schritt 6: Implementierung von Cloud Volumes ONTAP für AWS

Gehen Sie wie folgt vor, um Cloud Volumes ONTAP für AWS zu implementieren.

Schritte

1. Führen Sie im Stammordner den folgenden Befehl aus, um die Bereitstellung auszulösen:

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

   Zwei Container werden ausgelöst, der erste Container implementiert Cloud Volumes ONTAP und der zweite Container sendet Telemetriedaten an AutoSupport.

   Der zweite Container wartet, bis der erste Container alle Schritte erfolgreich abgeschlossen hat.

2. Überwachen Sie den Fortschritt des Bereitstellungsprozesses mithilfe der Protokolldateien:

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

   Dieser Befehl liefert die Ausgabe in Echtzeit und erfasst die Daten in den folgenden Protokolldateien: `deployment.log`

   `telemetry_asup.log`

   Sie können den Namen dieser Protokolldateien ändern, indem Sie die Datei mithilfe der folgenden Umgebungsvariablen bearbeiten `.env`:

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   Die folgenden Beispiele zeigen, wie Sie die Protokolldateinamen ändern:

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Nachdem Sie fertig sind

Mit den folgenden Schritten können Sie die temporäre Umgebung entfernen und Elemente bereinigen, die

während des Bereitstellungsprozesses erstellt wurden.

Schritte

1. Wenn Sie FlexCache bereitgestellt haben, legen Sie die folgende Option in der `terraform.tfvars` Variablendatei fest. Dadurch werden FlexCache-Volumes bereinigt und die zuvor erstellte temporäre Umgebung wird entfernt.

   ```
   flexcache_operation = "destroy"
   ```

   Die möglichen Optionen sind `deploy` und `destroy`.

2. Wenn Sie FlexClone bereitgestellt haben, legen Sie die folgende Option in der `terraform.tfvars` Variablendatei fest. Dadurch werden FlexClone-Volumes bereinigt und die zuvor erstellte temporäre Umgebung wird entfernt.

   ```
   flexclone_operation = "destroy"
   ```

   Die möglichen Optionen sind `deploy` und `destroy`.

Cloud Volumes ONTAP für Azure

Cloud Volumes ONTAP für Azure – Burst in die Cloud

Dieser Artikel unterstützt die NetApp Cloud Volumes ONTAP for Azure Automation Solution, die NetApp -Kunden über den NetApp Console Automation Hub zur Verfügung steht.

Die Automatisierungslösung Cloud Volumes ONTAP für Azure automatisiert die Container-Implementierung von Cloud Volumes ONTAP für Azure mithilfe von Terraform, sodass Sie Cloud Volumes ONTAP für Azure schnell und ohne manuelles Eingreifen implementieren können.

Bevor Sie beginnen

- Sie müssen die "Cloud Volumes ONTAP Azure – Burst in die Cloud" Automatisierungslösung über die Web-Benutzeroberfläche der Konsole. Die Lösung ist verpackt als `CVO-Azure-Burst-To-Cloud.zip`Die
- Sie müssen eine Linux-VM im gleichen Netzwerk wie Cloud Volumes ONTAP installieren.
- Nach der Installation der Linux-VM müssen Sie die Schritte in dieser Lösung befolgen, um die erforderlichen Abhängigkeiten zu installieren.

Schritt: Installieren Sie Docker und Docker Compose

Installation Von Docker

Die folgenden Schritte verwenden Ubuntu 20.04 Debian Linux-Distributionssoftware als Beispiel. Die Befehle, die Sie ausführen, hängen von der Linux-Distributionssoftware ab, die Sie verwenden. Informationen zur Konfiguration finden Sie in der Dokumentation der jeweiligen Linux-Distributionssoftware.

Schritte

1. Installieren Sie Docker, indem Sie die folgenden Befehle ausführen `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. Überprüfen Sie die Installation:

```
docker –version
```

3. Vergewissern Sie sich, dass auf Ihrem Linux-System eine Gruppe namens „Docker“ erstellt wurde. Erstellen Sie bei Bedarf die Gruppe:

```
sudo groupadd docker
```

4. Fügen Sie den Benutzer hinzu, der der Gruppe Zugriff auf Docker benötigt:

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

5. Ihre Änderungen werden übernommen, nachdem Sie sich beim Terminal abmelden und wieder anmelden. Alternativ können Sie die Änderungen sofort anwenden:

```
newgrp docker
```

Installieren Sie Docker Compose

Schritte

1. Installieren Sie Docker Compose, indem Sie die folgenden Befehle ausführen `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. Überprüfen Sie die Installation:

```
docker-compose –version
```

Schritt 2: Vorbereiten des Docker Images

Schritte

1. Kopieren Sie den `CVO-Azure-Burst-To-Cloud.zip` Ordner auf die Linux-VM, die Sie zum Bereitstellen von Cloud Volumes ONTAP verwenden möchten:

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

   - `private-key.pem` Ist Ihre private Schlüsseldatei für die Anmeldung ohne Kennwort.
   - `azureuser` Ist der VM-Benutzername.
   - `IP_ADDRESS_OF_VM` Ist die VM IP-Adresse.
   - `LOCATION_TO_BE_COPIED` Ist der Speicherort, an den der Ordner kopiert werden soll.

2. Extrahieren Sie den `CVO-Azure-Burst-To-Cloud.zip` Ordner. Sie können den Ordner im aktuellen Verzeichnis oder an einem benutzerdefinierten Speicherort extrahieren.

   Um den Ordner im aktuellen Verzeichnis zu extrahieren, führen Sie Folgendes aus:

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

   Um den Ordner an einem benutzerdefinierten Speicherort zu extrahieren, führen Sie Folgendes aus:

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

3. Navigieren Sie nach dem Extrahieren des Inhalts zum Ordner, `CVO_Azure_Deployment` und führen Sie den folgenden Befehl aus, um die Dateien anzuzeigen:

```
 ls -la
```

   Sie sollten eine Liste von Dateien sehen, ähnlich wie das folgende Beispiel:

```
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. Suchen Sie die `cvo_azure_flexcache_ubuntu_image_latest.tar.gz` Datei. Dieses enthält das Docker Image, das für die Implementierung von Cloud Volumes ONTAP für Azure erforderlich ist.
5. Enttaren Sie die Datei:

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

6. Warten Sie einige Minuten, bis das Docker-Image geladen ist, und überprüfen Sie dann, ob das Docker-Image erfolgreich geladen wurde:

```
docker images
```

Sie sollten ein Docker-Image mit dem `latest` Tag sehen `cvo_azure_flexcache_ubuntu_image_latest`, wie im folgenden Beispiel gezeigt:

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

Schritt 3: Erstellen Sie variable Umgebungsdateien

In dieser Phase müssen Sie zwei Umgebungsvariablendateien erstellen. Eine Datei dient der Authentifizierung von Azure Resource Manager-APIs mithilfe der Anmeldeinformationen des Dienstprinzipals. Die zweite Datei dient zum Festlegen von Umgebungsvariablen, damit die Terraform-Konsolenmodule Azure-APIs finden und authentifizieren können.

Schritte

1. Erstellen Sie einen Dienstprinzipal.

   Bevor Sie die Umgebungsvariablen-Dateien erstellen können, müssen Sie einen Dienstprinzipal erstellen, indem Sie die Schritte in befolgen"Erstellen Sie eine Azure Active Directory-Applikation und einen Dienstprinzipal, die auf Ressourcen zugreifen können".

2. Weisen Sie die Rolle **Contributor** dem neu erstellten Service-Prinzipal zu.
3. Erstellen Sie eine benutzerdefinierte Rolle.
   a. Suchen Sie die `sp_role.json` Datei, und prüfen Sie unter den aufgeführten Aktionen, ob die erforderlichen Berechtigungen vorhanden sind.
   b. Fügen Sie diese Berechtigungen ein und hängen Sie die benutzerdefinierte Rolle an den neu erstellten Dienstprinzipal an.
4. Navigieren Sie zu **Certificates & Secrets** und wählen Sie **New Client secret**, um das Client-Secret zu erstellen.

   Wenn Sie das Client-Secret erstellen, müssen Sie die Details aus der Spalte **Wert** aufzeichnen, da Sie diesen Wert nicht mehr sehen können. Außerdem müssen Sie folgende Informationen erfassen:

   - Client-ID
   - Abonnement-ID
   - Mandanten-ID

   Sie benötigen diese Informationen, um die Umgebungsvariablen zu erstellen. Die Client-ID und die Mandanten-ID finden Sie im Abschnitt **Übersicht** der Service Principal UI.
5. Erstellen Sie die Umgebungsdateien.
   a. Erstellen Sie die `azureauth.env` Datei an folgendem Speicherort:

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

      i. Fügen Sie der Datei folgenden Inhalt hinzu:

         ClientID=<> ClientSecret=<> SubscriptionID=<> tenantId=<>

      Das Format **muss** genau wie oben dargestellt sein, ohne Leerzeichen zwischen Schlüssel und Wert.

   b. Erstellen Sie die `credentials.env` Datei an folgendem Speicherort:

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

      i. Fügen Sie der Datei folgenden Inhalt hinzu:

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

   Das Format **muss** genau wie oben dargestellt sein, ohne Leerzeichen zwischen Schlüssel und Wert.
6. Fügen Sie der Datei die absoluten Dateipfade hinzu `.env`.

   Geben Sie den absoluten Pfad für die Umgebungsdatei in die `.env` Datei ein `azureauth.env`, die der Umgebungsvariable entspricht `AZURE_RM_CREDS`.

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

   Geben Sie den absoluten Pfad für die Umgebungsdatei in die `.env` Datei ein `credentials.env`, die der Umgebungsvariable entspricht `BLUEXP_TF_AZURE_CREDS`.

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

Schritt 4: Registrieren Sie sich für NetApp Intelligent Services

Melden Sie sich über Ihren Cloud-Anbieter für NetApp Intelligent Services an und zahlen Sie stundenweise (PAYGO) oder über einen Jahresvertrag. Zu den intelligenten Diensten von NetApp gehören NetApp Backup und Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience und NetApp Disaster Recovery. NetApp Data Classification ist ohne zusätzliche Kosten in Ihrem Abonnement enthalten

Schritte

1. Navigieren Sie im Azure-Portal zu **SaaS** und wählen Sie * NetApp Intelligent Services abonnieren* aus.
2. Wählen Sie den Plan **Cloud Manager (nach Cap PYGO nach Stunde, WORM und Datendiensten)** aus.

   Sie können entweder dieselbe Ressourcengruppe wie Cloud Volumes ONTAP oder eine andere Ressourcengruppe verwenden.

3. Konfigurieren Sie das Konsolenportal, um das SaaS-Abonnement in die Konsole zu importieren.

   Sie können dies direkt über das Azure-Portal konfigurieren, indem Sie zu **Produkt- und Plandetails** navigieren und die Option **Jetzt Konto konfigurieren** auswählen.

   Sie werden dann zum Konsolenportal weitergeleitet, um die Konfiguration zu bestätigen.

4. Bestätigen Sie die Konfiguration im Konsolenportal, indem Sie **Speichern** auswählen.

Schritt 5: Erstellen Sie ein externes Volume

Sie sollten ein externes Volume erstellen, damit die Terraform-Statusdateien und andere wichtige Dateien erhalten bleiben. Sie müssen sicherstellen, dass die Dateien für Terraform verfügbar sind, um den Workflow und die Implementierungen auszuführen.

Schritte

1. Externes Volume außerhalb von Docker Compose erstellen:

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

   Beispiel:

   ```
   docker volume create cvo_azure_volume_dst
   ```

2. Verwenden Sie eine der folgenden Optionen:

   a. Fügen Sie einen externen Volume-Pfad zur Umgebungsdatei hinzu `.env`.

      Sie müssen das genaue unten dargestellte Format einhalten.

      Format:

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

Beispiel:
`PERSISTENT_VOL=cvo_azure_volume_dst:/cvo_azure`

b. Fügen Sie NFS-Freigaben als externes Volume hinzu.

Stellen Sie sicher, dass der Docker Container mit den NFS-Freigaben kommunizieren kann und dass die korrekten Berechtigungen wie Lese-/Schreibvorgänge konfiguriert sind.

i. Fügen Sie den Pfad der NFS-Freigaben als Pfad zum externen Volume in der Docker Compose-Datei hinzu, wie unten gezeigt: Format:

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

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

3. Navigieren Sie zum `cvo_azure_variables` Ordner.

Im Ordner sollten die folgenden Variablendateien angezeigt werden:

`terraform.tfvars`

`variables.tf`

4. Ändern Sie die Werte innerhalb der `terraform.tfvars` Datei entsprechend Ihren Anforderungen.

Sie müssen die spezifische Begleitdokumentation lesen, wenn Sie einen der Variablenwerte in der Datei ändern `terraform.tfvars`. Die Werte können je nach Region, Verfügbarkeitszonen und anderen von Cloud Volumes ONTAP für Azure unterstützten Faktoren variieren. Dies umfasst Lizenzen, Festplattengröße und VM-Größe für einzelne Nodes sowie Hochverfügbarkeitspaare (HA).

Alle unterstützenden Variablen für den Konsolenagenten und die Cloud Volumes ONTAP Terraform-Module sind bereits in der `variables.tf` Datei. Sie müssen auf die Variablennamen in der `variables.tf` Datei vor dem Hinzufügen zur `terraform.tfvars` Datei.

5. Je nach Ihren Anforderungen können Sie FlexCache und FlexClone aktivieren oder deaktivieren, indem Sie die folgenden Optionen auf oder `false` einstellen `true`.

Die folgenden Beispiele aktivieren FlexCache und FlexClone:

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

6. Bei Bedarf können Sie den Wert für die Terraform-Variable aus dem Azure Active Directory-Dienst abrufen `az_service_principal_object_id`:

a. Navigieren Sie zu **Enterprise Applications –> All Applications** und wählen Sie den Namen des zuvor erstellten Service Principal aus.

b. Kopieren Sie die Objekt-ID, und fügen Sie den Wert für die Terraform-Variable ein:

`az_service_principal_object_id`

Schritt 6: Implementierung von Cloud Volumes ONTAP für Azure

Gehen Sie wie folgt vor, um Cloud Volumes ONTAP für Azure zu implementieren.

Schritte

1. Führen Sie im Stammordner den folgenden Befehl aus, um die Bereitstellung auszulösen:

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

   Zwei Container werden ausgelöst, der erste Container implementiert Cloud Volumes ONTAP und der zweite Container sendet Telemetriedaten an AutoSupport.

   Der zweite Container wartet, bis der erste Container alle Schritte erfolgreich abgeschlossen hat.

2. Überwachen Sie den Fortschritt des Bereitstellungsprozesses mithilfe der Protokolldateien:

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

   Dieser Befehl liefert die Ausgabe in Echtzeit und erfasst die Daten in den folgenden Protokolldateien:

   `deployment.log`

   `telemetry_asup.log`

   Sie können den Namen dieser Protokolldateien ändern, indem Sie die Datei mithilfe der folgenden Umgebungsvariablen bearbeiten `.env`:

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   Die folgenden Beispiele zeigen, wie Sie die Protokolldateinamen ändern:

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Nachdem Sie fertig sind

Mit den folgenden Schritten können Sie die temporäre Umgebung entfernen und Elemente bereinigen, die während des Bereitstellungsprozesses erstellt wurden.

Schritte

1. Wenn Sie FlexCache bereitgestellt haben, legen Sie die folgende Option in der `terraform.tfvars` Datei fest. Dadurch werden FlexCache-Volumes bereinigt und die zuvor erstellte temporäre Umgebung wird entfernt.

   `flexcache_operation = "destroy"`

   Die möglichen Optionen sind `deploy` und `destroy`.

2. Wenn Sie FlexClone bereitgestellt haben, legen Sie die folgende Option in der `terraform.tfvars` Datei fest. Dadurch werden FlexClone-Volumes bereinigt und die zuvor erstellte temporäre Umgebung wird entfernt.

```
flexclone_operation = "destroy"
```

Die möglichen Optionen sind `deploy` und `destroy`.

Cloud Volumes ONTAP für Google Cloud

Cloud Volumes ONTAP für Google Cloud – Burst in die Cloud

Dieser Artikel unterstützt die NetApp Cloud Volumes ONTAP für Google Cloud Automation Solution, die NetApp -Kunden über den NetApp Console Automation Hub zur Verfügung steht.

Die Automatisierungslösung Cloud Volumes ONTAP für Google Cloud automatisiert die Container-Implementierung von Cloud Volumes ONTAP für Google Cloud, sodass Sie Cloud Volumes ONTAP für Google Cloud schnell und ohne manuelle Eingriffe implementieren können.

Bevor Sie beginnen

- Sie müssen die "Cloud Volumes ONTAP für Google Cloud – Burst in die Cloud" Automatisierungslösung über die Web-Benutzeroberfläche der Konsole. Die Lösung ist verpackt als `cvo_gcp_flexcache.zip`Die
- Sie müssen eine Linux-VM im gleichen Netzwerk wie Cloud Volumes ONTAP installieren.
- Nach der Installation der Linux-VM müssen Sie die Schritte in dieser Lösung befolgen, um die erforderlichen Abhängigkeiten zu installieren.

Schritt: Installieren Sie Docker und Docker Compose

Installation Von Docker

Die folgenden Schritte verwenden Ubuntu 20.04 Debian Linux-Distributionssoftware als Beispiel. Die Befehle, die Sie ausführen, hängen von der Linux-Distributionssoftware ab, die Sie verwenden. Informationen zur Konfiguration finden Sie in der Dokumentation der jeweiligen Linux-Distributionssoftware.

Schritte

1. Installieren Sie Docker, indem Sie die folgenden Befehle ausführen:

```
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. Überprüfen Sie die Installation:

```
docker –version
```

3. Vergewissern Sie sich, dass auf Ihrem Linux-System eine Gruppe namens „Docker“ erstellt wurde. Erstellen Sie bei Bedarf die Gruppe:

```
sudo groupadd docker
```

4. Fügen Sie den Benutzer hinzu, der der Gruppe Zugriff auf Docker benötigt:

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

5. Ihre Änderungen werden übernommen, nachdem Sie sich beim Terminal abmelden und wieder anmelden. Alternativ können Sie die Änderungen sofort anwenden:

```
newgrp docker
```

Installieren Sie Docker Compose

Schritte

1. Installieren Sie Docker Compose, indem Sie die folgenden Befehle ausführen `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. Überprüfen Sie die Installation:

```
docker-compose –version
```

Schritt 2: Vorbereiten des Docker Images

Schritte

1. Kopieren Sie den `cvo_gcp_flexcache.zip` Ordner auf die Linux-VM, die Sie zum Bereitstellen von Cloud Volumes ONTAP verwenden möchten:

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

- `private-key.pem` Ist Ihre private Schlüsseldatei für die Anmeldung ohne Kennwort.
- `gcpuser` Ist der VM-Benutzername.
- `IP_ADDRESS_OF_VM` Ist die VM IP-Adresse.
- `LOCATION_TO_BE_COPIED` Ist der Speicherort, an den der Ordner kopiert werden soll.

2. Extrahieren Sie den `cvo_gcp_flexcache.zip` Ordner. Sie können den Ordner im aktuellen Verzeichnis oder an einem benutzerdefinierten Speicherort extrahieren.

   Um den Ordner im aktuellen Verzeichnis zu extrahieren, führen Sie Folgendes aus:

   ```
   unzip cvo_gcp_flexcache.zip
   ```

   Um den Ordner an einem benutzerdefinierten Speicherort zu extrahieren, führen Sie Folgendes aus:

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

3. Führen Sie nach dem Extrahieren des Inhalts den folgenden Befehl aus, um die Dateien anzuzeigen:

   ```
    ls -la
   ```

   Sie sollten eine Liste von Dateien sehen, ähnlich wie das folgende Beispiel:

   ```
       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. Suchen Sie die `cvo_gcp_flexcache_ubuntu_image.tar` Datei. Dieses enthält das Docker Image, das für die Implementierung von Cloud Volumes ONTAP für Google Cloud erforderlich ist.
5. Enttaren Sie die Datei:

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

6. Warten Sie einige Minuten, bis das Docker-Image geladen ist, und überprüfen Sie dann, ob das Docker-Image erfolgreich geladen wurde:

```
docker images
```

Sie sollten ein Docker-Image mit dem `latest` Tag sehen `cvo_gcp_flexcache_ubuntu_image`, wie im folgenden Beispiel gezeigt:

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

Bei Bedarf können Sie den Docker-Image-Namen ändern. Wenn Sie den Docker-Image-Namen ändern, müssen Sie den Docker-Image-Namen in den Dateien und `docker-compose-destroy` aktualisieren `docker-compose-deploy`.

Schritt 3: Aktualisieren Sie die JSON-Datei

In dieser Phase müssen Sie die Datei mit einem Servicekontoschlüssel aktualisieren `cxo-automation-gcp.json`, um den Google Cloud-Provider zu authentifizieren.

1. Erstellen Sie ein Dienstkonto mit Berechtigungen zum Bereitstellen von Cloud Volumes ONTAP und eines Konsolenagenten"Erfahren Sie mehr über das Erstellen von Servicekonten."
2. Laden Sie die Schlüsseldatei für das Konto herunter, und aktualisieren Sie die `cxo-automation-gcp.json` Datei mit den Informationen zur Schlüsseldatei. Die `cxo-automation-gcp.json` Datei befindet sich im `cvo_gcp_variables` Ordner.

Beispiel

```
{
  "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"
}
```

Das Dateiformat muss genau wie oben dargestellt sein.

Schritt 4: Registrieren Sie sich für NetApp Intelligent Services

Melden Sie sich über Ihren Cloud-Anbieter für NetApp Intelligent Services an und zahlen Sie stundenweise (PAYGO) oder über einen Jahresvertrag. Zu den intelligenten Diensten von NetApp gehören NetApp Backup und Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience und NetApp Disaster Recovery. Die NetApp Datenklassifizierung ist ohne zusätzliche Kosten in Ihrem Abonnement enthalten.

Schritte

1. Navigieren Sie zum"Google Cloud-Konsole" und wählen Sie * NetApp Intelligent Services abonnieren*.
2. Konfigurieren Sie das NetApp Konsolenportal, um das SaaS-Abonnement in die Konsole zu importieren.

   Sie können dies direkt über die Google Cloud Platform konfigurieren. Sie werden zum Konsolenportal weitergeleitet, um die Konfiguration zu bestätigen.

3. Bestätigen Sie die Konfiguration im Konsolenportal, indem Sie **Speichern** auswählen.

Weitere Informationen finden Sie unter "Verwalten Sie Google Cloud-Anmeldeinformationen und Abonnements für die NetApp Konsole" .

Schritt 5: Aktivieren Sie die erforderlichen Google Cloud APIs

Sie müssen die folgenden Google Cloud-APIs in Ihrem Projekt aktivieren, um Cloud Volumes ONTAP und den Konsolenagenten bereitzustellen.

- Cloud Deployment Manager V2-API
- Cloud-ProtokollierungsAPI
- Cloud Resource Manager API
- Compute Engine-API

- IAM-API (Identitäts- und Zugriffsmanagement

"Erfahren Sie mehr über die Aktivierung von APIs"

Schritt 6: Erstellen Sie ein externes Volume

Sie sollten ein externes Volume erstellen, damit die Terraform-Statusdateien und andere wichtige Dateien erhalten bleiben. Sie müssen sicherstellen, dass die Dateien für Terraform verfügbar sind, um den Workflow und die Implementierungen auszuführen.

Schritte

1. Externes Volume außerhalb von Docker Compose erstellen:

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

   Beispiel:

   ```
   docker volume create cvo_gcp_volume_dst
   ```

2. Verwenden Sie eine der folgenden Optionen:

   a. Fügen Sie einen externen Volume-Pfad zur Umgebungsdatei hinzu `.env`.

      Sie müssen das genaue unten dargestellte Format einhalten.

      Format:

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

      Beispiel:
      `PERSISTENT_VOL=cvo_gcp_volume_dst:/cvo_gcp`

   b. Fügen Sie NFS-Freigaben als externes Volume hinzu.

      Stellen Sie sicher, dass der Docker Container mit den NFS-Freigaben kommunizieren kann und dass die korrekten Berechtigungen wie Lese-/Schreibvorgänge konfiguriert sind.

      i. Fügen Sie den Pfad der NFS-Freigaben als Pfad zum externen Volume in der Docker Compose-Datei hinzu, wie unten gezeigt: Format:

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

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

3. Navigieren Sie zum `cvo_gcp_variables` Ordner.

   Folgende Dateien sollten im Ordner angezeigt werden:

   ◦ `terraform.tfvars`

- variables.tf

4. Ändern Sie die Werte innerhalb der `terraform.tfvars` Datei entsprechend Ihren Anforderungen.

   Sie müssen die spezifische Begleitdokumentation lesen, wenn Sie einen der Variablenwerte in der Datei ändern `terraform.tfvars`. Die Werte können je nach Region, Verfügbarkeitszonen und anderen von Cloud Volumes ONTAP für Google Cloud unterstützten Faktoren variieren. Dies umfasst Lizenzen, Festplattengröße und VM-Größe für einzelne Nodes sowie Hochverfügbarkeitspaare (HA).

   Alle unterstützenden Variablen für den Konsolenagenten und die Cloud Volumes ONTAP Terraform-Module sind bereits in der `variables.tf` Datei. Sie müssen auf die Variablennamen in der `variables.tf` Datei vor dem Hinzufügen zur `terraform.tfvars` Datei.

5. Je nach Ihren Anforderungen können Sie FlexCache und FlexClone aktivieren oder deaktivieren, indem Sie die folgenden Optionen auf oder `false` einstellen `true`.

   Die folgenden Beispiele aktivieren FlexCache und FlexClone:

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

Schritt 7: Implementierung von Cloud Volumes ONTAP für Google Cloud

Führen Sie die folgenden Schritte zur Implementierung von Cloud Volumes ONTAP für Google Cloud durch.

Schritte

1. Führen Sie im Stammordner den folgenden Befehl aus, um die Bereitstellung auszulösen:

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

   Zwei Container werden ausgelöst, der erste Container implementiert Cloud Volumes ONTAP und der zweite Container sendet Telemetriedaten an AutoSupport.

   Der zweite Container wartet, bis der erste Container alle Schritte erfolgreich abgeschlossen hat.

2. Überwachen Sie den Fortschritt des Bereitstellungsprozesses mithilfe der Protokolldateien:

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

   Dieser Befehl liefert die Ausgabe in Echtzeit und erfasst die Daten in den folgenden Protokolldateien: `deployment.log`

   `telemetry_asup.log`

   Sie können den Namen dieser Protokolldateien ändern, indem Sie die Datei mithilfe der folgenden Umgebungsvariablen bearbeiten `.env`:

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

Die folgenden Beispiele zeigen, wie Sie die Protokolldateinamen ändern:

```
DEPLOYMENT_LOGS=<your_deployment_log_filename>.log

TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log
```

Nachdem Sie fertig sind

Mit den folgenden Schritten können Sie die temporäre Umgebung entfernen und Elemente bereinigen, die während des Bereitstellungsprozesses erstellt wurden.

Schritte

1. Wenn Sie FlexCache bereitgestellt haben, legen Sie die folgende Option in der `terraform.tfvars` Datei fest. Dadurch werden FlexCache-Volumes bereinigt und die zuvor erstellte temporäre Umgebung wird entfernt.

   ```
   flexcache_operation = "destroy"
   ```

   Die möglichen Optionen sind `deploy` und `destroy`.

2. Wenn Sie FlexClone bereitgestellt haben, legen Sie die folgende Option in der `terraform.tfvars` Datei fest. Dadurch werden FlexClone-Volumes bereinigt und die zuvor erstellte temporäre Umgebung wird entfernt.

   ```
   flexclone_operation = "destroy"
   ```

   Die möglichen Optionen sind `deploy` und `destroy`.

ONTAP

Tag 0/1

Überblick über die ONTAP-Tag-0/1-Lösung

Mit der ONTAP Day 0/1 Automatisierungslösung können Sie einen ONTAP Cluster mithilfe von Ansible bereitstellen und konfigurieren. Die Lösung ist erhältlich bei "NetApp Console Automatisierungszentrale"Die

Flexible Implementierungsoptionen für ONTAP

Je nach Ihren Anforderungen können Sie lokale Hardware verwenden oder ONTAP simulieren, um einen ONTAP Cluster mithilfe von Ansible zu implementieren und zu konfigurieren.

On-Premises-Hardware

Sie können diese Lösung mit On-Premises-Hardware mit ONTAP wie einem FAS oder einem AFF System implementieren. Sie müssen eine Linux VM verwenden, um den ONTAP-Cluster mit Ansible zu implementieren und zu konfigurieren.

ONTAP simulieren

Um diese Lösung mit einem ONTAP-Simulator implementieren zu können, müssen Sie die aktuellste Version

von Simulate ONTAP von der NetApp Support-Website herunterladen. Simulieren ONTAP ist ein virtueller Simulator für ONTAP Software. Simulieren Sie ONTAP in einem VMware Hypervisor auf einem Windows-, Linux- oder Mac-System. Für Windows- und Linux-Hosts müssen Sie den VMware Workstation-Hypervisor verwenden, um diese Lösung auszuführen. Wenn Sie über ein Mac-Betriebssystem verfügen, verwenden Sie den VMware Fusion-Hypervisor.

Mehrlagiges Design

Das Ansible-Framework vereinfacht die Entwicklung und Wiederverwendung von Automatisierungsausführung und logischen Aufgaben. Das Framework unterscheidet zwischen den Entscheidungsaufgaben (Logikschicht) und den Ausführungsschritten (Ausführungsebene) in der Automatisierung. Wenn Sie verstehen, wie diese Ebenen funktionieren, können Sie die Konfiguration anpassen.

In einem Ansible-„Playbook“ werden verschiedene Aufgaben vom Anfang bis zum Ende ausgeführt. Das `site.yml` Playbook enthält das `logic.yml` Playbook und das `execution.yml` Playbook.

Wenn eine Anfrage ausgeführt wird, ruft das `site.yml` Playbook zuerst in das `logic.yml` Playbook auf und ruft dann das Playbook zur Ausführung der Service-Anfrage auf `execution.yml`.

Sie müssen die logische Schicht des Frameworks nicht verwenden. Die Logikebene bietet Optionen zur Erweiterung der Funktionalität des Frameworks über die hartcodierten Werte für die Ausführung hinaus. Auf diese Weise können Sie die Framework-Funktionen bei Bedarf anpassen.

Logische Ebene

Die Logikschicht besteht aus folgenden Komponenten:

- Das Playbook `logic.yml`
- Logische Aufgabendateien im `logic-tasks` Verzeichnis

Die Logikebene bietet die Möglichkeit für komplexe Entscheidungen, ohne dass eine umfassende benutzerdefinierte Integration erforderlich ist (z. B. eine Verbindung zu ServiceNow). Die Logikebene ist konfigurierbar und liefert die Eingabe zu Microservices.

Die Möglichkeit, die Logikschicht zu umgehen, wird ebenfalls bereitgestellt. Wenn Sie die logische Ebene umgehen möchten, definieren Sie die Variable nicht `logic_operation`. Der direkte Aufruf des `logic.yml` Playbooks ermöglicht es, ein gewisses Maß an Debugging ohne Ausführung durchzuführen. Sie können eine „Debug“-Anweisung verwenden, um zu überprüfen, ob der Wert des `raw_service_request` korrekt ist.

Wichtige Überlegungen:

- Das `logic.yml` Playbook sucht nach der `logic_operation` Variablen. Wenn die Variable in der Anfrage definiert ist, wird eine Aufgabendatei aus dem Verzeichnis geladen `logic-tasks`. Die Task-Datei muss eine .yml-Datei sein. Wenn keine passende Task-Datei vorhanden ist und die `logic_operation` Variable definiert ist, schlägt die Logikebene fehl.
- Der Standardwert der `logic_operation` Variable ist `no-op`. Wenn die Variable nicht explizit definiert ist, wird standardmäßig auf, gesetzt `no-op`, das keine Operationen ausführt.
- Wenn die `raw_service_request` Variable bereits definiert ist, wird die Ausführung zur Ausführungsebene fortgesetzt. Wenn die Variable nicht definiert ist, schlägt die logische Ebene fehl.

Ausführungsebene

Die Ausführungsebene besteht aus folgenden Komponenten:

- Das Playbook `execution.yml`

Die Ausführungsebene führt die API-Aufrufe zum Konfigurieren eines ONTAP-Clusters durch. Das `execution.yml` Playbook setzt voraus, dass die `raw_service_request` Variable bei der Ausführung definiert ist.

Unterstützung für Anpassungen

Sie können diese Lösung auf verschiedene Weise an Ihre Anforderungen anpassen.

Die Anpassungsoptionen umfassen:

- Ändern von Ansible Playbooks
- Hinzufügen von Rollen

Ansible-Dateien anpassen

In der folgenden Tabelle werden die in dieser Lösung enthaltenen anpassbaren Ansible-Dateien beschrieben.

Standort	Beschreibung
`playbooks/inventory/hosts`	Enthält eine einzelne Datei mit einer Liste von Hosts und Gruppen.
`playbooks/group_vars/all/*`	Ansible bietet eine praktische Möglichkeit, Variablen auf mehrere Hosts gleichzeitig anzuwenden. Sie können alle oder alle Dateien in diesem Ordner ändern, einschließlich `cfg.yml`, , , `clusters.yml` `defaults.yml` , `services.yml` `standards.yml` und `vault.yml`.
`playbooks/logic-tasks`	Unterstützung von Entscheidungsaufgaben innerhalb von Ansible und Beibehaltung der Trennung von Logik und Ausführung Sie können diesem Ordner Dateien hinzufügen, die dem entsprechenden Dienst entsprechen.
`playbooks/vars/*`	Dynamische Werte, die in Ansible Playbooks und Rollen verwendet werden, um Anpassungen, Flexibilität und Wiederverwendbarkeit von Konfigurationen zu ermöglichen. Bei Bedarf können Sie alle oder alle Dateien in diesem Ordner ändern.

Anpassen von Rollen

Sie können die Lösung auch anpassen, indem Sie Ansible-Rollen, auch Microservices genannt, hinzufügen oder ändern. Weitere Informationen finden Sie unter "Anpassen".

Bereiten Sie sich auf die Verwendung der Lösung für den ONTAP Tag 0/1 vor

Vor der Implementierung der Automatisierungslösung müssen Sie die ONTAP-Umgebung vorbereiten und Ansible installieren und konfigurieren.

Erste Überlegungen zur Planung

Lesen Sie sich die folgenden Anforderungen und Überlegungen durch, bevor Sie diese Lösung zum Bereitstellen eines ONTAP-Clusters verwenden.

Grundvoraussetzungen

Sie müssen die folgenden grundlegenden Anforderungen erfüllen, um diese Lösung verwenden zu können:

- Sie müssen auf die ONTAP-Software zugreifen können – entweder vor Ort oder über einen ONTAP-Simulator.
- Sie müssen wissen, wie Sie die ONTAP Software nutzen.
- Sie müssen wissen, wie Sie die Automatisierungssoftware-Tools von Ansible verwenden können.

Überlegungen zur Planung

Vor der Implementierung dieser Automatisierungslösung müssen Sie folgende Entscheidungen treffen:

- Der Speicherort, an dem der Ansible-Steuerungsknoten ausgeführt werden soll.
- Dem ONTAP System, entweder vor Ort Hardware oder einem ONTAP Simulator.
- Ob Sie eine Anpassung benötigen.

Bereiten Sie das ONTAP-System vor

Unabhängig davon, ob Sie ein lokales ONTAP System nutzen oder ONTAP simulieren, müssen Sie die Umgebung vorbereiten, bevor die Automatisierungslösung implementiert werden kann.

Optional können Sie Simulate ONTAP installieren und konfigurieren

Wenn Sie diese Lösung über einen ONTAP Simulator bereitstellen möchten, müssen Sie Simulate ONTAP herunterladen und ausführen.

Bevor Sie beginnen

- Sie müssen den VMware Hypervisor herunterladen und installieren, den Sie verwenden werden, um Simulate ONTAP auszuführen.
  - Wenn Sie über ein Windows- oder Linux-Betriebssystem verfügen, verwenden Sie VMware Workstation.
  - Wenn Sie ein Mac-Betriebssystem verwenden, verwenden Sie VMware Fusion.

  Wenn Sie ein Mac-Betriebssystem verwenden, benötigen Sie einen Intel-Prozessor.

Schritte

Gehen Sie wie folgt vor, um zwei ONTAP Simulatoren in Ihrer lokalen Umgebung zu installieren:

1. Laden Sie Simulate ONTAP aus dem "NetApp Support Website".

   Obwohl Sie zwei ONTAP Simulatoren installieren, müssen Sie nur eine Kopie der Software herunterladen.

2. Wenn die Anwendung noch nicht ausgeführt wird, starten Sie die VMware-Anwendung.
3. Suchen Sie die heruntergeladene Simulatordatei, und klicken Sie mit der rechten Maustaste, um sie mit der VMware-Anwendung zu öffnen.
4. Legen Sie den Namen der ersten ONTAP-Instanz fest.
5. Warten Sie, bis der Simulator hochgefahren ist, und befolgen Sie die Anweisungen zum Erstellen eines einzelnen Node-Clusters.

   Wiederholen Sie die Schritte für die zweite ONTAP-Instanz.

6. Fügen Sie optional eine vollständige Datenträgerergänzung hinzu.

   Führen Sie in jedem Cluster die folgenden Befehle aus:

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

Der Status des ONTAP Systems

Sie müssen den Anfangsstatus des ONTAP Systems überprüfen, unabhängig davon, ob es sich vor Ort befindet oder über einen ONTAP Simulator ausgeführt wird.

Stellen Sie sicher, dass die folgenden ONTAP-Systemanforderungen erfüllt sind:

- ONTAP ist installiert und läuft ohne Cluster definiert.
- ONTAP wird gebootet und zeigt die IP-Adresse für den Zugriff auf das Cluster an.
- Das Netzwerk ist erreichbar.
- Sie haben Admin-Anmeldedaten.
- Das MOTD-Banner (Message of the Day) wird mit der Managementadresse angezeigt.

Installieren Sie die erforderliche Automatisierungssoftware

Dieser Abschnitt enthält Informationen über die Installation von Ansible und die Vorbereitung der Automatisierungslösung für die Implementierung.

Installation Von Ansible

Ansible kann auf Linux oder Windows Systemen installiert werden.

Die standardmäßige Kommunikationsmethode, die Ansible für die Kommunikation mit einem ONTAP-Cluster verwendet, ist SSH.

Informationen zur Installation von Ansible finden Sie unter"Erste Schritte mit NetApp und Ansible – Installation von Ansible".

Ansible muss auf dem Steuerungsknoten des Systems installiert sein.

Laden Sie die Automatisierungslösung herunter und bereiten Sie sie vor

Sie können die Automatisierungslösung mit den folgenden Schritten herunterladen und für die Implementierung vorbereiten.

1. Laden Sie die "ONTAP - Tag 0/1 Health Checks" Automatisierungslösung über die Web-Benutzeroberfläche der Konsole. Die Lösung ist verpackt als `ONTAP_DAY0_DAY1.zip`Die
2. Extrahieren Sie den ZIP-Ordner und kopieren Sie die Dateien an den gewünschten Speicherort auf dem Steuerknoten in Ihrer Ansible-Umgebung.

Anfängliche Ansible-Framework-Konfiguration

Führen Sie die Erstkonfiguration des Ansible-Frameworks durch:

1. Navigieren Sie zu `playbooks/inventory/group_vars/all`.
2. Entschlüsseln der `vault.yml` Datei:

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

   Wenn Sie zur Eingabe des Vault-Passworts aufgefordert werden, geben Sie das folgende temporäre Passwort ein:

   ```
   NetApp123!
   ```

   > ⓘ „NetApp123!“ ist ein temporäres Kennwort zum Entschlüsseln der `vault.yml` Datei und des entsprechenden Vault-Passworts. Nach der ersten Verwendung müssen Sie die Datei mit Ihrem eigenen Passwort verschlüsseln.

3. Ändern Sie die folgenden Ansible-Dateien:

   - `clusters.yml` - Ändern Sie die Werte in dieser Datei, um Ihre Umgebung anzupassen.
   - `vault.yml` - Nach der Entschlüsselung der Datei, ändern Sie die ONTAP-Cluster, Benutzername und Passwort-Werte, um Ihre Umgebung anzupassen.
   - `cfg.yml` - Setzen Sie den Dateipfad für `log2file` und `show_request` unter `cfg` auf `True`, um die anzuzeigen `raw_service_request`.

     Die `raw_service_request` Variable wird in den Protokolldateien und während der Ausführung angezeigt.

   > ⓘ Jede aufgeführte Datei enthält Kommentare mit Anweisungen, wie sie entsprechend Ihren Anforderungen geändert werden kann.

4. Verschlüsseln Sie die Datei erneut `vault.yml`:

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

   > ⓘ Sie werden bei der Verschlüsselung aufgefordert, ein neues Passwort für den Tresor auszuwählen.

5. Navigieren Sie zu `playbooks/inventory/hosts` einem gültigen Python Interpreter und legen Sie ihn fest.
6. Bereitstellung des `framework_test` Service:

   Mit dem folgenden Befehl wird das Modul mit dem `gather_subset` Wert `cluster_identity_info` ausgeführt `na_ontap_info`. Dadurch wird überprüft, ob die Grundkonfiguration korrekt ist und ob Sie mit dem Cluster kommunizieren können.

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

Führen Sie den Befehl für jedes Cluster aus.

Wenn der Erfolg erfolgreich ist, sollte die Ausgabe wie im folgenden Beispiel angezeigt werden:

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

Implementieren Sie den ONTAP Cluster mit der Lösung

Nach Abschluss der Vorbereitung und Planung sind Sie bereit, mit der ONTAP Day 0/1 Lösung schnell einen ONTAP Cluster mithilfe von Ansible zu konfigurieren.

Sie können jederzeit während der Schritte in diesem Abschnitt auswählen, ob Sie eine Anforderung testen möchten, anstatt sie tatsächlich auszuführen. Um eine Anforderung zu testen, ändern Sie das `site.yml` Playbook in der Befehlszeile in `logic.yml`.

> Der `docs/tutorial-requests.txt` Speicherort enthält die endgültige Version aller Service-Requests, die während dieses Verfahrens verwendet werden. Wenn Sie Schwierigkeiten bei der Ausführung einer Service-Anfrage haben, können Sie die entsprechende Anforderung aus der Datei an den `playbooks/inventory/group_vars/all/tutorial-requests.yml` Speicherort kopieren `tutorial-requests.txt` und die hartcodierten Werte nach Bedarf ändern (IP-Adresse, Aggregatnamen usw.). Die Anforderung sollte dann erfolgreich ausgeführt werden können.

Bevor Sie beginnen

- Ansible muss installiert sein.
- Sie müssen die ONTAP Day 0/1-Lösung heruntergeladen und den Ordner an den gewünschten Speicherort auf dem Ansible-Steuerungsknoten extrahiert haben.
- Der ONTAP-Systemstatus muss die Anforderungen erfüllen und Sie müssen über die erforderlichen Anmeldedaten verfügen.
- Sie müssen alle erforderlichen Aufgaben abgeschlossen haben, die im Abschnitt beschrieben "Vorbereiten"sind.

> Die Beispiele in dieser Lösung verwenden „Cluster_01“ und „Cluster_02“ als Namen für die beiden Cluster. Sie müssen diese Werte durch die Namen der Cluster in Ihrer Umgebung ersetzen.

Schritt: Erstkonfiguration des Clusters

In dieser Phase müssen Sie zunächst einige Schritte zur Cluster-Konfiguration durchführen.

Schritte

1. Navigieren Sie zum `playbooks/inventory/group_vars/all/tutorial-requests.yml` Speicherort und prüfen Sie die `cluster_initial` Anforderung in der Datei. Nehmen Sie alle erforderlichen Änderungen an Ihrer Umgebung vor.
2. Erstellen Sie eine Datei im `logic-tasks` Ordner für die Service-Anfrage. Erstellen Sie beispielsweise eine Datei mit dem Namen `cluster_initial.yml`.

   Kopieren Sie die folgenden Zeilen in die neue Datei:

   ```
   - 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. Definieren Sie die `raw_service_request` Variable.

   Sie können eine der folgenden Optionen verwenden, um die Variable in der Datei zu `cluster_initial.yml` definieren `raw_service_request`, die Sie im Ordner erstellt `logic-tasks` haben:

   - **Option 1**: Variable manuell definieren `raw_service_request`.

     Öffnen Sie die `tutorial-requests.yml` Datei mit einem Editor und kopieren Sie den Inhalt von Zeile 11 in Zeile 165. Fügen Sie den Inhalt unter der Variablen in der neuen `cluster_initial.yml` Datei ein `raw service request`, wie in den folgenden Beispielen gezeigt:


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
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"

Beispiel anzeigen

Beispieldatei `cluster_initial.yml`:

```
- 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**: Verwenden Sie eine Jinja-Vorlage, um die Anforderung zu definieren:

  Sie können auch das folgende Jinja-Vorlagenformat verwenden, um den Wert zu erhalten `raw_service_request`.

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

4. Führen Sie die Erstkonfiguration des Clusters für das erste Cluster durch:

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

Vergewissern Sie sich, dass keine Fehler vorliegen, bevor Sie fortfahren.

5. Wiederholen Sie den Befehl für das zweite Cluster:

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

Vergewissern Sie sich, dass beim zweiten Cluster keine Fehler auftreten.

Wenn Sie zu Beginn der Ansible-Ausgabe nach oben scrollen, sollten Sie die an das Framework gesendete Anforderung sehen, wie im folgenden Beispiel gezeigt:

Beispiel anzeigen

```
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",
```

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

6. Melden Sie sich bei jeder ONTAP-Instanz an und überprüfen Sie, ob die Anforderung erfolgreich war.

Schritt 2: Konfigurieren der Intercluster LIFs

Sie können jetzt die Intercluster LIFs konfigurieren, indem Sie der Anforderung die LIF-Definitionen hinzufügen `cluster_initial` und den Microservice definieren `ontap_interface`.

Die Servicedefinition und die Anforderung arbeiten zusammen, um die Aktion zu bestimmen:

- Wenn Sie eine Service-Anfrage für einen Microservice bereitstellen, der nicht in den Servicedefinitionen enthalten ist, wird die Anforderung nicht ausgeführt.
- Wenn Sie eine Service-Anfrage mit einer oder mehreren in den Servicedefinitionen definierten Microservices bereitstellen, aber aus der Anfrage ausgelassen werden, wird die Anforderung nicht ausgeführt.

Das `execution.yml` Playbook wertet die Dienstdefinition aus, indem die Liste der Microservices in der aufgeführten Reihenfolge durchsucht wird:

- Wenn in der Anfrage ein Eintrag mit einem Wörterbuchschlüssel vorhanden ist, der dem Eintrag in den Microservice-Definitionen entspricht `args`, wird die Anforderung ausgeführt.
- Wenn in der Service-Anfrage kein übereinstimmender Eintrag vorhanden ist, wird die Anforderung

fehlerfrei übersprungen.

Schritte

1. Navigieren Sie zu der `cluster_initial.yml` zuvor erstellten Datei, und ändern Sie die Anforderung, indem Sie den Anforderungsdefinitionen die folgenden Zeilen hinzufügen:

```
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. Führen Sie den Befehl aus:

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

3. Melden Sie sich bei jeder Instanz an, um zu überprüfen, ob die LIFs dem Cluster hinzugefügt wurden:

Beispiel anzeigen

```
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.
```

Die Ausgabe zeigt an, dass die LIFs **nicht** hinzugefügt wurden. Der Grund dafür ist, dass der `ontap_interface` Microservice noch in der Datei definiert werden `services.yml` muss.

4. Vergewissern Sie sich, dass die LIFs der Variable hinzugefügt wurden `raw_service_request`.

Beispiel anzeigen

Im folgenden Beispiel werden die LIFs zur Anforderung hinzugefügt:

```
            "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. Definieren Sie den `ontap_interface` Microservice unter `cluster_initial` in der `services.yml` Datei.

   Kopieren Sie die folgenden Zeilen in die Datei, um den Microservice zu definieren:

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

6. Nachdem nun der `ontap_interface` Microservice in der Anfrage und der Datei definiert wurde `services.yml`, führen Sie die Anforderung erneut aus:

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

7. Loggen Sie sich bei jeder ONTAP Instanz ein und überprüfen Sie, ob die LIFs hinzugefügt wurden.

Schritt 3: Optional mehrere Cluster konfigurieren

Bei Bedarf können Sie mehrere Cluster in derselben Anforderung konfigurieren. Sie müssen beim Definieren der Anforderung für jedes Cluster Variablennamen angeben.

Schritte

1. Fügen Sie einen Eintrag für das zweite Cluster in der Datei hinzu `cluster_initial.yml`, um beide Cluster in derselben Anforderung zu konfigurieren.

   Im folgenden Beispiel wird das Feld angezeigt `ontap_aggr`, nachdem der zweite Eintrag hinzugefügt wurde.

```
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. Übernehmen Sie die Änderungen für alle anderen Elemente unter `cluster_initial`.
3. Fügen Sie Cluster-Peering zur Anforderung hinzu, indem Sie die folgenden Zeilen in die Datei kopieren:

```
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. Ansible-Anforderung ausführen:

```
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>
```

Schritt 4: Anfängliche SVM-Konfiguration

An dieser Stelle des Verfahrens konfigurieren Sie die SVMs im Cluster.

Schritte

1. Anforderung `tutorial-requests.yml` zur Konfiguration einer SVM- und SVM-Peer-Beziehung aktualisieren `svm_initial`

   Sie müssen Folgendes konfigurieren:

   - Das SVM

- Die SVM-Peer-Beziehung
- Die SVM-Schnittstelle für jede SVM

2. Aktualisieren Sie die Variablendefinitionen in den `svm_initial` Anforderungsdefinitionen. Sie müssen die folgenden Variablendefinitionen ändern:
   - `cluster_name`
   - `vserver_name`
   - `peer_cluster_name`
   - `peer_vserver`

   Um die Definitionen zu aktualisieren, entfernen Sie das **'{}'** nach `req_details` für die `svm_initial` Definition und fügen Sie die korrekte Definition hinzu.

3. Erstellen Sie eine Datei im `logic-tasks` Ordner für die Service-Anfrage. Erstellen Sie beispielsweise eine Datei mit dem Namen `svm_initial.yml`.

   Kopieren Sie die folgenden Zeilen in die Datei:

```
- 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. Definieren Sie die `raw_service_request` Variable.

   Sie können eine der folgenden Optionen verwenden, um die Variable für `svm_initial` im Ordner zu `logic-tasks` definieren `raw_service_request`:

   - **Option 1**: Variable manuell definieren `raw_service_request`.

     Öffnen Sie die `tutorial-requests.yml` Datei mit einem Editor und kopieren Sie den Inhalt von

Zeile 179 in Zeile 222. Fügen Sie den Inhalt unter der Variablen in der neuen `svm_initial.yml` Datei ein `raw service request`, wie in den folgenden Beispielen gezeigt:

Beispiel anzeigen

Beispieldatei `svm_initial.yml`:

```
- 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**: Verwenden Sie eine Jinja-Vorlage, um die Anforderung zu definieren:

  Sie können auch das folgende Jinja-Vorlagenformat verwenden, um den Wert zu erhalten `raw_service_request`.

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

5. Anforderung ausführen:

```
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. Melden Sie sich bei jeder ONTAP Instanz an und validieren Sie die Konfiguration.
7. Fügen Sie die SVM-Schnittstellen hinzu.

   Definieren Sie den `ontap_interface` Dienst unter `svm_initial` in der `services.yml` Datei und führen Sie die Anforderung erneut aus:

```
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. Loggen Sie sich bei jeder ONTAP Instanz ein und überprüfen Sie, ob die SVM-Schnittstellen konfiguriert sind.

Schritt 5: Optional können Sie eine Service-Anfrage dynamisch definieren

In den vorherigen Schritten ist die `raw_service_request` Variable hartcodiert. Dies ist nützlich für Lernen, Entwicklung und Tests. Sie können auch eine Serviceanfrage dynamisch generieren.

Der folgende Abschnitt bietet eine Option zum dynamischen Erstellen des erforderlichen `raw_service_request`, wenn Sie es nicht in höhere Systeme integrieren möchten.

- Wenn die `logic_operation` Variable im Befehl nicht definiert ist, importiert die `logic.yml` Datei keine Datei aus dem `logic-tasks` Ordner. Das bedeutet, dass die `raw_service_request` außerhalb von Ansible definiert und bei der Ausführung dem Framework zur Verfügung gestellt werden muss.
- Ein Aufgabendateiname im `logic-tasks` Ordner muss mit dem Wert der Variablen ohne die Erweiterung .yml übereinstimmen `logic_operation`.
- Die Aufgabendateien im `logic-tasks` Ordner definieren dynamisch ein `raw_service_request`. die einzige Voraussetzung ist, dass ein gültiges `raw_service_request` als letzte Aufgabe in der entsprechenden Datei definiert wird.

Dynamische Definition von Service-Anfragen

Es gibt mehrere Möglichkeiten, eine logische Aufgabe anzuwenden, um eine Service-Anfrage dynamisch zu definieren. Einige dieser Optionen sind unten aufgeführt:

- Verwenden einer Ansible-Aufgabendatei aus dem `logic-tasks` Ordner
- Aufrufen einer benutzerdefinierten Rolle, die Daten zurückgibt, die für die Konvertierung in eine Variable geeignet `raw_service_request` sind.
- Aufruf eines weiteren Tools außerhalb der Ansible-Umgebung, um die erforderlichen Daten bereitzustellen Beispielsweise ein REST-API-Aufruf an Active IQ Unified Manager.

Mit den folgenden Beispielbefehlen können Sie mithilfe der Datei für jedes Cluster eine Service-Anfrage dynamisch definieren `tutorial-requests.yml`:

```
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
```

Schritt 6: Implementierung der ONTAP-Lösung Tag 0/1

In dieser Phase sollten Sie bereits Folgendes abgeschlossen haben:

- Alle Dateien in wurden entsprechend Ihren Anforderungen überprüft und geändert `playbooks/inventory/group_vars/all`. Jede Datei enthält detaillierte Kommentare, mit denen Sie die Änderungen vornehmen können.
- Erforderliche Aufgabendateien wurden dem Verzeichnis hinzugefügt `logic-tasks`.
- Alle erforderlichen Datendateien wurden dem Verzeichnis hinzugefügt `playbook/vars`.

Verwenden Sie die folgenden Befehle, um die ONTAP Day 0/1-Lösung bereitzustellen und den Zustand Ihrer Bereitstellung zu überprüfen:

Zu diesem Zeitpunkt sollten Sie die Datei bereits entschlüsselt und geändert haben `vault.yml` und sie muss mit Ihrem neuen Passwort verschlüsselt werden.

- Führen Sie den ONTAP-Tag-0-Service aus:

```
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>
```

- Führen Sie den ONTAP Day 1-Service aus:

```
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>
```

- Clusterweite Einstellungen anwenden:

```
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>
```

- Führen Sie Zustandsprüfungen durch:

```
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>
```

Passen Sie die ONTAP Day 0/1-Lösung an

Zur Anpassung der ONTAP-Day-0/1-Lösung an Ihre Anforderungen können Sie Ansible-Rollen hinzufügen oder ändern.

Rollen stellen die Microservices im Ansible-Framework dar. Jeder Microservice führt einen Vorgang durch. Beispielsweise ist ONTAP Tag 0 ein Service, der mehrere Microservices umfasst.

Ansible-Rollen hinzufügen

Sie können Ansible-Rollen hinzufügen, um die Lösung an Ihre Umgebung anzupassen. Erforderliche Rollen werden durch Servicedefinitionen im Ansible Framework definiert.

Eine Rolle muss die folgenden Anforderungen erfüllen, um als Microservice verwendet werden zu können:

- Akzeptieren Sie eine Liste der Argumente in der `args` Variablen.
- Nutzen Sie die Ansible-Struktur „Block, Rescue, Always“ mit bestimmten Anforderungen für jeden Block.
- Verwenden Sie ein einzelnes Ansible-Modul und definieren Sie eine einzelne Aufgabe innerhalb des Blocks.
- Implementieren Sie alle verfügbaren Modulparameter gemäß den in diesem Abschnitt beschriebenen Anforderungen.

Erforderliche Microservice-Struktur

Jede Rolle muss die folgenden Variablen unterstützen:

- `mode`: Wenn der Modus auf die Rolle eingestellt ist `test`, versucht der zu importieren, der `test.yml` zeigt, was die Rolle tut, ohne sie tatsächlich auszuführen.

  Dies ist aufgrund bestimmter Abhängigkeiten nicht immer möglich.

- `status`: Der Gesamtstatus der Ausführung des Playbooks. Wenn der Wert nicht auf die Rolle gesetzt `success` ist, wird nicht ausgeführt.
- `args` : Eine Liste rollenspezifischer Wörterbücher mit Schlüsseln, die den Rollenparameternamen entsprechen.
- `global_log_messages`: Sammelt Protokollmeldungen während der Ausführung des Playbooks. Bei jeder Ausführung der Rolle wird ein Eintrag generiert.
- `log_name`: Der Name, der auf die Rolle innerhalb der Einträge verweist `global_log_messages`.
- `task_descr`: Eine kurze Beschreibung dessen, was die Rolle tut.
- `service_start_time`: Der Zeitstempel, der verwendet wird, um die Zeit zu verfolgen, zu der jede Rolle ausgeführt wird.
- `playbook_status`: Der Status des Ansible-Playbooks.
- `role_result`: Die Variable, die die Rollenausgabe enthält und in jeder Nachricht innerhalb der Einträge enthalten `global_log_messages` ist.

Beispiel für eine Rollenstruktur

Das folgende Beispiel zeigt die grundlegende Struktur einer Rolle, die einen Microservice implementiert. Sie müssen die Variablen in diesem Beispiel für Ihre Konfiguration ändern.

Beispiel anzeigen

Grundlegende Rollenstruktur:

```
- 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 ] }}"
```

Variablen, die in der Beispielrolle verwendet werden:

- `<NAME>`: Ein austauschbarer Wert, der für jeden Microservice bereitgestellt werden muss.
- `<LOG_NAME>`: Der Kurzname der Rolle, die für Protokollierungszwecke verwendet wird. `ONTAP_VOLUME`Beispiel: .
- `<TASK_DESCRIPTION>`: Eine kurze Beschreibung dessen, was der Microservice tut.
- `<MODULE_NAME>`: Der Ansible-Modulname für die Aufgabe.

> ⓘ Im Playbook der obersten Ebene `execute.yml` wird die Sammlung angegeben `netapp.ontap`. Wenn das Modul Teil der Sammlung ist `netapp.ontap`, muss der Modulname nicht vollständig angegeben werden.

- `<MODULE_SPECIFIC_PARAMETERS>`: Ansible-Modulparameter, die spezifisch für das Modul sind, das zur Implementierung des Microservices verwendet wird. In der folgenden Liste werden die Parametertypen und deren Gruppierung beschrieben.
  - Erforderliche Parameter: Alle erforderlichen Parameter werden ohne Standardwert angegeben.
  - Parameter, die einen für den Microservice spezifischen Standardwert haben (nicht der gleiche Wert wie ein in der Moduldokumentation spezifizierter Standardwert).
  - Alle verbleibenden Parameter werden als Standardwert verwendet `default(omit)`.

Verwendung von mehrstufigen Wörterbüchern als Modulparameter

Einige von NetApp bereitgestellte Ansible-Module verwenden mehrstufige Wörterbücher für Modulparameter (z. B. feste und adaptive QoS-Richtliniengruppen).

Allein zu verwenden `default(omit)` funktioniert nicht, wenn diese Wörterbücher verwendet werden, besonders wenn es mehrere gibt und sie sich gegenseitig ausschließen.

Wenn Sie Multi-Level-Wörterbücher als Modulparameter verwenden müssen, sollten Sie die Funktionalität in mehrere Microservices (Rollen) aufteilen, so dass jeder garantiert mindestens einen Second-Level-Wörterbuchwert für das jeweilige Wörterbuch liefern kann.

Die folgenden Beispiele zeigen feste und anpassungsfähige QoS-Richtliniengruppen, die sich auf zwei Microservices verteilen.

Der erste Microservice enthält feste QoS-Richtliniengruppenwerte:

```
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)
}}"
```

Der zweite Microservice enthält die Werte der adaptiven QoS-Richtliniengruppe:

```
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) }}"
```