Centro de automatización de la NetApp Console

NetApp Automation

NetApp
November 18, 2025

Tabla de contenidos

Centro de automatización de la NetApp Console

Descripción general del centro de automatización de la NetApp Console

El centro de automatización de NetApp Console es una colección de soluciones de automatización disponibles para clientes, socios y empleados de NetApp . El centro de automatización cuenta con diversas características y ventajas.

Ubicación única para sus necesidades de automatización

Puedes acceder a "Centro de automatización de la NetApp Console" a través de la interfaz de usuario web de la consola. Esto proporciona una ubicación única para los scripts, playbooks y módulos necesarios para mejorar la automatización y el funcionamiento de sus productos y servicios de NetApp .

NetApp crea y prueba las soluciones

Todas las soluciones de automatización y scripts han sido creadas y probadas por NetApp. Cada solución está dirigida a un caso de uso o solicitud específica de cliente. La mayor prioridad es la integración con los servicios de datos y archivos de NetApp.

Documentación

Cada una de las soluciones de automatización incluye documentación asociada para ayudarle a comenzar. Si bien se accede a las soluciones a través de la interfaz web de la Consola, toda la documentación está disponible en este sitio. La documentación está organizada en función de los productos y servicios en la nube de NetApp .

Una base sólida para el futuro

NetApp se compromete a ayudar a sus clientes a mejorar y optimizar la automatización de sus centros de datos y entornos en la nube. Prevemos continuar mejorando el centro de automatización de la consola para dar respuesta a las necesidades de los clientes, los cambios tecnológicos y la continua integración de productos.

Queremos tener noticias suyas

El equipo de automatización de la Oficina de la experiencia del cliente de NetApp (CXO) le gustaría recibir noticias suyas. Si tiene algún comentario, problema o solicitud de características, envíe un correo electrónico a cxo automation team.

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP : Explosión a la nube

Puede utilizar esta solución de automatización para aprovisionar Amazon FSx for NetApp ONTAP con volúmenes y un FlexCache asociado.

> La administración de Amazon FSx for NetApp ONTAP también se conoce como **FSx para ONTAP**.

Acerca de esta solución

En general, el código de automatización proporcionado con esta solución realiza las siguientes acciones:

- Aprovisionar un sistema de archivos FSx para ONTAP de destino
- Aprovisione las máquinas virtuales de almacenamiento (SVM) para el sistema de archivos
- Cree una relación de paridad de clústeres entre los sistemas de origen y destino
- Cree una relación entre iguales de SVM entre el sistema de origen y el de destino para FlexCache
- Opcionalmente, crea volúmenes de FlexVol mediante FSx para ONTAP
- Crea un volumen de FlexCache en FSx para ONTAP con el origen que apunta al almacenamiento on-premises

La automatización se basa en Docker y Docker Compose que deben instalarse en la máquina virtual Linux como se describe a continuación.

Antes de empezar

Debe tener lo siguiente para completar el aprovisionamiento y la configuración:

- Necesitas descargar el "Amazon FSx for NetApp ONTAP : Explosión a la nube" Solución de automatización a través de la interfaz web de la NetApp Console . La solución se empaqueta como archivo `AWS_FSxN_BTC.zip`.
- Conectividad de red entre los sistemas de origen y destino.
- Una VM de Linux con las siguientes características:
  - Distribución de Linux basada en Debian
  - Puesto en marcha en el mismo subconjunto de VPC utilizado para el aprovisionamiento de FSx para ONTAP
- Cuenta de AWS.

Paso 1: Instale y configure Docker

Instalar y configurar Docker en una máquina virtual Linux basada en Debian.

Pasos

1. Preparar el entorno.

```
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. Instale Docker y verifique la instalación.

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

3. Añada el grupo Linux requerido con un usuario asociado.

   Primero comprueba si el grupo **docker** existe en tu sistema Linux. Si no es así, cree el grupo y agregue el usuario. De forma predeterminada, el usuario de shell actual se agrega al grupo.

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

4. Active las nuevas definiciones de grupo y usuario

   Si ha creado un nuevo grupo con un usuario, debe activar las definiciones. Para ello, puede cerrar la sesión de Linux y volver a iniciarla. O bien puede ejecutar el siguiente comando.

   ```
   newgrp docker
   ```

Paso 2: Instale Docker Compose

Instale Docker Compose en una máquina virtual Linux basada en Debian.

Pasos

1. Instale 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. Compruebe que la instalación se ha realizado correctamente.

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

Paso 3: Preparar la imagen de Docker

Debe extraer y cargar la imagen de Docker proporcionada con la solución de automatización.

Pasos

1. Copie el archivo de la solución `AWS_FSxN_BTC.zip` en la máquina virtual donde se ejecutará el código de automatización.

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

   El parámetro de entrada `private-key.pem` es el archivo de claves privadas utilizado para la autenticación de máquinas virtuales de AWS (instancia de EC2).

2. Desplácese a la carpeta correcta con el archivo de solución y descomprima el archivo.

```
unzip AWS_FSxN_BTC.zip
```

3. Navegue a la nueva carpeta `AWS_FSxN_BTC` creada con la operación de descompresión y enumere los archivos. Debería ver el archivo `aws_fsxn_flexcache_image_latest.tar.gz`.

```
ls -la
```

4. Cargue el archivo de imagen de Docker. La operación de carga debería completarse normalmente en unos segundos.

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

5. Confirme que se ha cargado la imagen de Docker.

```
docker images
```

   Deberías ver la imagen de Docker `aws_fsxn_flexcache_image` con la etiqueta `latest`.

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

Paso 4: Crear un archivo de entorno para las credenciales de AWS

Se debe crear un archivo de variable local para la autenticación mediante la clave secreta y de acceso. A continuación, agregue el archivo al `.env` archivo.

Pasos

1. Cree el `awsauth.env` archivo en la siguiente ubicación:

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

2. Agregue el siguiente contenido al archivo:

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

   El formato **debe** ser exactamente como se muestra arriba sin ningún espacio entre `key` y. `value`

3. Agregue la ruta de acceso absoluta al `.env` archivo mediante la `AWS_CREDS` variable. Por ejemplo:

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

Paso 5: Cree un volumen externo

Necesita un volumen externo para asegurarse de que los archivos de estado de Terraform y otros archivos importantes son persistentes. Estos archivos deben estar disponibles para que Terraform ejecute el flujo de trabajo y las implementaciones.

Pasos

1. Cree un volumen externo fuera de Docker Compose.

   Asegúrese de actualizar el nombre del volumen (último parámetro) al valor apropiado antes de ejecutar el comando.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Añada la ruta al volumen externo al `.env` archivo de entorno mediante el comando:

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

   Recuerde mantener el contenido del archivo existente y el formato de dos puntos. Por ejemplo:

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

   En su lugar, se puede agregar un recurso compartido de NFS como volumen externo mediante un comando, como el siguiente:

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

3. Actualice las variables de Terraform.
   a. Navegue a la carpeta `aws_fsxn_variables`.
   b. Confirme que existen los dos archivos siguientes `terraform.tfvars`: Y `variables.tf`.
   c. Actualice los valores en `terraform.tfvars` según sea necesario para el entorno.

      Consulte "Recurso de Terraform: aws_fsx_ONTAP_file_system" para obtener más información.

Paso 6: Aprovisionar Amazon FSx for NetApp ONTAP y FlexCache

Puede aprovisionar Amazon FSx for NetApp ONTAP y FlexCache.

Pasos

1. Navegue hasta la raíz de la carpeta (AWS_FSXN_BTC) y ejecute el comando de aprovisionamiento.

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

   Este comando crea dos contenedores. El primer contenedor pone en marcha FSx para ONTAP y el

segundo contenedor crea las relaciones entre iguales de clústeres, las relaciones entre iguales de SVM, el volumen de destino y FlexCache.

2. Supervisar el proceso de aprovisionamiento.

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

Este comando le da la salida en tiempo real, pero se ha configurado para capturar los logs a través del archivo `deployment.log`. Puede cambiar el nombre de estos archivos log editando el `.env` archivo y actualizando las variables `DEPLOYMENT_LOGS`.

Paso 7: Destruya Amazon FSx for NetApp ONTAP y FlexCache

Opcionalmente, puede eliminar y quitar Amazon FSx for NetApp ONTAP y FlexCache.

1. Defina la variable `flexcache_operation` del `terraform.tfvars` archivo en Destruir.
2. Navegue hasta la raíz de la carpeta (AWS_FSXN_BTC) y ejecute el siguiente comando.

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

Este comando crea dos contenedores. El primer contenedor elimina FlexCache y el segundo contenedor elimina FSx para ONTAP.

3. Supervisar el proceso de aprovisionamiento.

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

Amazon FSx for NetApp ONTAP : recuperación ante desastres

Puede utilizar esta solución de automatización para realizar una copia de seguridad de recuperación ante desastres de un sistema de origen mediante la administración de Amazon FSx for NetApp ONTAP .

> La administración de Amazon FSx for NetApp ONTAP también se conoce como **FSx para ONTAP**.

Acerca de esta solución

En general, el código de automatización proporcionado con esta solución realiza las siguientes acciones:

- Aprovisionar un sistema de archivos FSx para ONTAP de destino
- Aprovisione las máquinas virtuales de almacenamiento (SVM) para el sistema de archivos
- Cree una relación de paridad de clústeres entre los sistemas de origen y destino
- Cree una relación entre iguales de SVM entre el sistema de origen y el de destino para SnapMirror
- Crear volúmenes de destino

- Crear una relación de SnapMirror entre los volúmenes de origen y de destino
- Inicie la transferencia de SnapMirror entre los volúmenes de origen y destino

La automatización se basa en Docker y Docker Compose que deben instalarse en la máquina virtual Linux como se describe a continuación.

Antes de empezar

Debe tener lo siguiente para completar el aprovisionamiento y la configuración:

- Necesitas descargar el "Amazon FSx for NetApp ONTAP : recuperación ante desastres" Solución de automatización a través de la interfaz web de la NetApp Console . La solución se presenta empaquetada como `FSxN_DR.zip`. Este archivo zip contiene el `AWS_FSxN_Bck_Prov.zip` archivo que utilizará para implementar la solución descrita en este documento.
- Conectividad de red entre los sistemas de origen y destino.
- Una VM de Linux con las siguientes características:
  - Distribución de Linux basada en Debian
  - Puesto en marcha en el mismo subconjunto de VPC utilizado para el aprovisionamiento de FSx para ONTAP
- Una cuenta de AWS.

Paso 1: Instale y configure Docker

Instalar y configurar Docker en una máquina virtual Linux basada en Debian.

Pasos

1. Preparar el entorno.

```
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. Instale Docker y verifique la instalación.

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

3. Añada el grupo Linux requerido con un usuario asociado.

   Primero comprueba si el grupo **docker** existe en tu sistema Linux. Si no existe, cree el grupo y agregue el usuario. De forma predeterminada, el usuario de shell actual se agrega al grupo.

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

4. Active las nuevas definiciones de grupo y usuario

   Si ha creado un nuevo grupo con un usuario, debe activar las definiciones. Para ello, puede cerrar la sesión de Linux y volver a iniciarla. O bien puede ejecutar el siguiente comando.

```
newgrp docker
```

Paso 2: Instale Docker Compose

Instale Docker Compose en una máquina virtual Linux basada en Debian.

Pasos

1. Instale 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. Compruebe que la instalación se ha realizado correctamente.

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

Paso 3: Preparar la imagen de Docker

Debe extraer y cargar la imagen de Docker proporcionada con la solución de automatización.

Pasos

1. Copie el archivo de la solución `AWS_FSxN_Bck_Prov.zip` en la máquina virtual donde se ejecutará el código de automatización.

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

   El parámetro de entrada `private-key.pem` es el archivo de claves privadas utilizado para la autenticación de máquinas virtuales de AWS (instancia de EC2).

2. Desplácese a la carpeta correcta con el archivo de solución y descomprima el archivo.

```
unzip AWS_FSxN_Bck_Prov.zip
```

3. Navegue a la nueva carpeta `AWS_FSxN_Bck_Prov` creada con la operación de descompresión y enumere los archivos. Debería ver el archivo `aws_fsxn_bck_image_latest.tar.gz`.

```
ls -la
```

4. Cargue el archivo de imagen de Docker. La operación de carga debería completarse normalmente en unos segundos.

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

5. Confirme que se ha cargado la imagen de Docker.

```
docker images
```

Deberías ver la imagen de Docker `aws_fsxn_bck_image` con la etiqueta `latest`.

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

Paso 4: Crear un archivo de entorno para las credenciales de AWS

Se debe crear un archivo de variable local para la autenticación mediante la clave secreta y de acceso. A continuación, agregue el archivo al `.env` archivo.

Pasos

1. Cree el `awsauth.env` archivo en la siguiente ubicación:

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

2. Agregue el siguiente contenido al archivo:

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

   El formato **debe** ser exactamente como se muestra arriba sin ningún espacio entre `key` y. `value`

3. Agregue la ruta de acceso absoluta al `.env` archivo mediante la `AWS_CREDS` variable. Por ejemplo:

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

Paso 5: Cree un volumen externo

Necesita un volumen externo para asegurarse de que los archivos de estado de Terraform y otros archivos importantes son persistentes. Estos archivos deben estar disponibles para que Terraform ejecute el flujo de trabajo y las implementaciones.

Pasos

1. Cree un volumen externo fuera de Docker Compose.

   Asegúrese de actualizar el nombre del volumen (último parámetro) al valor apropiado antes de ejecutar el comando.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Añada la ruta al volumen externo al `.env` archivo de entorno mediante el comando:

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

   Recuerde mantener el contenido del archivo existente y el formato de dos puntos. Por ejemplo:

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

   En su lugar, se puede agregar un recurso compartido de NFS como volumen externo mediante un comando, como el siguiente:

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

3. Actualice las variables de Terraform.
   a. Navegue a la carpeta `aws_fsxn_variables`.
   b. Confirme que existen los dos archivos siguientes `terraform.tfvars`: Y `variables.tf`.
   c. Actualice los valores en `terraform.tfvars` según sea necesario para el entorno.

      Consulte "Recurso de Terraform: aws_fsx_ONTAP_file_system" para obtener más información.

Paso 6: Implemente la solución de backup

Puede poner en marcha y aprovisionar la solución de backup de recuperación ante desastres.

Pasos

1. Navegue hasta la raíz de la carpeta (aws_FSxN_Bck_Prov) y ejecute el comando de provisionamiento.

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

   Este comando crea tres contenedores. El primer contenedor pone en marcha FSx para ONTAP. El segundo contenedor crea la relación de iguales de clústeres, la relación entre iguales de SVM y el volumen de destino. El tercer contenedor crea la relación de SnapMirror e inicia la transferencia de SnapMirror.

2. Supervisar el proceso de aprovisionamiento.

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

Este comando le da la salida en tiempo real, pero se ha configurado para capturar los logs a través del archivo `deployment.log`. Puede cambiar el nombre de estos archivos log editando el `.env` archivo y actualizando las variables `DEPLOYMENT_LOGS`.

Azure NetApp Files

Instalar Oracle mediante Azure NetApp Files

Puede usar esta solución de automatización para aprovisionar volúmenes Azure NetApp Files e instalar Oracle en una máquina virtual disponible. A continuación, Oracle utiliza los volúmenes para el almacenamiento de datos.

Acerca de esta solución

En general, el código de automatización proporcionado con esta solución realiza las siguientes acciones:

- Configure una cuenta de NetApp en Azure
- Configure un pool de capacidad de almacenamiento en Azure
- Aprovisione los volúmenes Azure NetApp Files según la definición
- Cree los puntos de montaje
- Monte los volúmenes Azure NetApp Files en los puntos de montaje
- Instale Oracle en el servidor Linux
- Cree los listeners y la base de datos
- Crear Bases de Datos de Conexión (PDB)
- Inicie el listener y la instancia de Oracle
- Instale y configure la `azacsnap` utilidad para tomar una instantánea

Antes de empezar

Debe tener lo siguiente para completar la instalación:

- Necesitas descargar el "Oracle con Azure NetApp Files" Solución de automatización a través de la interfaz web de la NetApp Console . La solución se empaqueta como archivo `na_oracle19c_deploy-master.zip`.
- Una VM de Linux con las siguientes características:
  - RHEL 8 (Standard_D8s_v3-RHEL-8)
  - Se implementa en la misma red virtual de Azure utilizada para el aprovisionamiento de Azure NetApp Files
- Una cuenta de Azure

La solución de automatización se proporciona como una imagen y se ejecuta con Docker y Docker Compose. Debe instalar ambos en la máquina virtual Linux como se describe a continuación.

También debe registrar la VM con RedHat mediante el comando `sudo subscription-manager register`. El comando le solicitará las credenciales de su cuenta. Si es necesario, puede crear una cuenta en https://developers.redhat.com/.

Paso 1: Instale y configure Docker

Instalar y configurar Docker en una máquina virtual de RHEL 8 Linux.

Pasos

1. Instale el software Docker con los siguientes comandos.

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

2. Inicie Docker y muestre la versión para confirmar que la instalación se ha realizado correctamente.

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

3. Añada el grupo Linux requerido con un usuario asociado.

   Primero comprueba si el grupo **docker** existe en tu sistema Linux. Si no es así, cree el grupo y agregue el usuario. De forma predeterminada, el usuario de shell actual se agrega al grupo.

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

4. Active las nuevas definiciones de grupo y usuario

   Si ha creado un nuevo grupo con un usuario, debe activar las definiciones. Para ello, puede cerrar la sesión de Linux y volver a iniciarla. O bien puede ejecutar el siguiente comando.

```
newgrp docker
```

Paso 2: Instale Docker Compose y las utilidades NFS

Instale y configure Docker Compose junto con el paquete de utilidades NFS.

Pasos

1. Instale Docker Compose y muestre la versión para confirmar que la instalación se ha realizado correctamente.

```
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. Instale el paquete de utilidades NFS.

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

Paso 3: Descargue los archivos de instalación de Oracle

Descargue la instalación y los archivos de parches de Oracle necesarios, así como la `azacsnap` utilidad.

Pasos

1. Inicie sesión en su cuenta de Oracle según sea necesario.
2. Descargue los siguientes archivos.

Archivo	Descripción
LINUX.X64_193000_db_home.zip	instalador de base 19,3
p31281355_190000_Linux-x86-64.zip	19,8 parche RU
p6880880_190000_Linux-x86-64.zip	opatch versión 12.2.0.1.23
azacsnap_installer_v5.0.run	instalador de azacsnap

3. Coloque todos los archivos de instalación en la carpeta `/tmp/archive`.
4. Asegúrese de que todos los usuarios del servidor de bases de datos tengan acceso completo (lectura, escritura, ejecución) a la carpeta `/tmp/archive`.

Paso 4: Preparar la imagen de Docker

Debe extraer y cargar la imagen de Docker proporcionada con la solución de automatización.

Pasos

1. Copie el archivo de la solución `na_oracle19c_deploy-master.zip` en la máquina virtual donde se ejecutará el código de automatización.

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

   El parámetro de entrada `private-key.pem` es el archivo de clave privada utilizado para la autenticación de máquinas virtuales de Azure.

2. Desplácese a la carpeta correcta con el archivo de solución y descomprima el archivo.

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

3. Navegue a la nueva carpeta `na_oracle19c_deploy-master` creada con la operación de descompresión y enumere los archivos. Debería ver el archivo `ora_anf_bck_image.tar`.

```
ls -lt
```

4. Cargue el archivo de imagen de Docker. La operación de carga debería completarse normalmente en unos segundos.

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

5. Confirme que se ha cargado la imagen de Docker.

```
docker images
```

   Deberías ver la imagen de Docker `ora_anf_bck_image` con la etiqueta `latest`.

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

Paso 5: Cree un volumen externo

Necesita un volumen externo para asegurarse de que los archivos de estado de Terraform y otros archivos importantes son persistentes. Estos archivos deben estar disponibles para que Terraform ejecute el flujo de trabajo y las implementaciones.

Pasos

1. Cree un volumen externo fuera de Docker Compose.

   Asegúrese de actualizar el nombre del volumen antes de ejecutar el comando.

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

2. Añada la ruta al volumen externo al `.env` archivo de entorno mediante el comando:

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

   Recuerde mantener el contenido del archivo existente y el formato de dos puntos. Por ejemplo:

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

3. Actualice las variables de Terraform.
   a. Navegue a la carpeta `ora_anf_variables`.
   b. Confirme que existen los dos archivos siguientes `terraform.tfvars`: Y `variables.tf`.
   c. Actualice los valores en `terraform.tfvars` según sea necesario para el entorno.

Paso 6: Instalar Oracle

Ahora puede provisionar e instalar Oracle.

Pasos

1. Instale Oracle con la siguiente secuencia de comandos.

```
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. Vuelva a cargar las variables Bash y confirme mostrando el valor para `ORACLE_HOME`.
   a. `cd /home/oracle`
   b. `source .bash_profile`
   c. `echo $ORACLE_HOME`
3. Debe poder conectarse a Oracle.

```
sudo su oracle
```

Paso 7: Validar la instalación de Oracle

Debe confirmar que la instalación de Oracle se ha realizado correctamente.

Pasos

1. Conéctese al servidor Oracle de Linux y muestre una lista de los procesos de Oracle. Esto confirma que la instalación se ha completado como se esperaba y que la base de datos Oracle se está ejecutando.

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

2. Conéctese a la base de datos para examinar la configuración de la base de datos y confirmar que las PDB se han creado correctamente.

```
sqlplus / as sysdba
```

Debería ver una salida similar a la siguiente:

```
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. Ejecute unos sencillos comandos SQL para confirmar que la base de datos está disponible.

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

Paso 8: Instale la utilidad azacsnap y realice una copia de seguridad de instantáneas

Debe instalar y ejecutar la `azacsnap` utilidad para realizar un backup de snapshot.

Pasos

1. Instale el contenedor.

```
docker-compose up azacsnap_install
```

2. Cambie a la cuenta de usuario de instantánea.

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

3. Configurar un archivo de detalles de copia de seguridad de almacenamiento. Esto creará el `azacsnap.json` archivo de configuración.

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

4. Realizar un backup de snapshot.

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

Paso 9: Opcionalmente, migre una PDB local a la nube

Opcionalmente, puede migrar la PDB local a la nube.

Pasos

1. Configure las variables en `tfvars` los archivos según sea necesario para su entorno.
2. Migre la PDB.

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

Cloud Volumes ONTAP para AWS

Cloud Volumes ONTAP para AWS: Estallido a la nube

Este artículo es compatible con la solución de automatización NetApp Cloud Volumes ONTAP para AWS, que está disponible para los clientes de NetApp desde el centro de automatización de la NetApp Console .

La solución de automatización de Cloud Volumes ONTAP para AWS automatiza la implementación en contenedores de Cloud Volumes ONTAP para AWS mediante Terraform, lo que le permite implementar Cloud Volumes ONTAP para AWS rápidamente, sin intervención manual.

Antes de empezar

- Debes descargar el "AWS de Cloud Volumes ONTAP: Ráfagas en la nube" Solución de automatización a través de la interfaz web de la consola. La solución se presenta empaquetada como `cvo_aws_flexcache.zip`.
- Debe instalar una máquina virtual Linux en la misma red que Cloud Volumes ONTAP.
- Después de instalar la máquina virtual Linux, debe seguir los pasos de esta solución para instalar las dependencias necesarias.

Paso 1: Instalar Docker y Docker Compose

Instale Docker

Los siguientes pasos usan el software de distribución Linux Ubuntu 20,04 como ejemplo. Los comandos que ejecute dependen del software de distribución de Linux que utilice. Consulte la documentación específica del software de distribución de Linux para conocer la configuración.

Pasos

1. Instale Docker ejecutando los siguientes `sudo` comandos:

```
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. Compruebe la instalación:

```
docker –version
```

3. Compruebe que se ha creado un grupo denominado «docker» en su sistema Linux. Si es necesario, cree el grupo:

```
sudo groupadd docker
```

4. Agregue el usuario que necesita acceder a Docker al grupo:

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

5. Los cambios se aplican después de cerrar la sesión y volver a conectarse al terminal. Como alternativa, puede aplicar los cambios inmediatamente:

```
newgrp docker
```

Instale Docker Compose

Pasos

1. Instale Docker Compose ejecutando los siguientes `sudo` comandos:

```
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. Compruebe la instalación:

```
docker-compose –version
```

Paso 2: Preparar la imagen de Docker

Pasos

1. Copie `cvo_aws_flexcache.zip` la carpeta en la máquina virtual Linux que desee utilizar para implementar Cloud Volumes ONTAP:

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

   - `private-key.pem` es su archivo de clave privada para iniciar sesión sin contraseña.
   - `awsuser` Es el nombre de usuario de la máquina virtual.
   - `IP_ADDRESS_OF_VM` Es la dirección IP del equipo virtual.
   - `LOCATION_TO_BE_COPIED` es la ubicación donde se copiará la carpeta.

2. Extraiga la `cvo_aws_flexcache.zip` carpeta. Puede extraer la carpeta en el directorio actual o en una ubicación personalizada.

   Para extraer la carpeta del directorio actual, ejecute:

```
unzip cvo_aws_flexcache.zip
```

   Para extraer la carpeta en una ubicación personalizada, ejecute:

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

3. Después de extraer el contenido, desplácese a la `CVO_Aws_Deployment` carpeta y ejecute el siguiente comando para ver los archivos:

```
 ls -la
```

   Debería ver una lista de archivos, similar al siguiente ejemplo:

```
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. Busque el `cvo_aws_flexcache_ubuntu_image.tar` archivo. Esto contiene la imagen de Docker necesaria para poner en marcha Cloud Volumes ONTAP para AWS.
5. Abra el archivo:

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

6. Espere unos minutos hasta que se cargue la imagen de Docker y, a continuación, valide que la imagen de Docker se haya cargado correctamente:

```
docker images
```

Debería ver una imagen de Docker llamada `cvo_aws_flexcache_ubuntu_image` con la `latest` etiqueta, como se muestra en el siguiente ejemplo:

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

Puede cambiar el nombre de la imagen de Docker si es necesario. Si cambia el nombre de la imagen de Docker, asegúrese de actualizar el nombre de la imagen de Docker en los `docker-compose-deploy` archivos y. `docker-compose-destroy`

Paso 3: Crear archivos de variables de entorno

En esta etapa, debes crear dos archivos de variables de entorno. Un archivo es para la autenticación de las API de AWS Resource Manager mediante las claves secretas y de acceso de AWS. El segundo archivo sirve para configurar variables de entorno para permitir que los módulos Terraform de la consola localicen y autentiquen las API de AWS.

Pasos

1. Cree el `awsauth.env` archivo en la siguiente ubicación:

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

   a. Agregue el siguiente contenido al `awsauth.env` archivo:

      access_key=<> clave_secreta=<>

   El formato **debe** ser exactamente como se muestra arriba.

2. Agregue la ruta de acceso absoluta al `.env` archivo.

   Introduzca la ruta de acceso absoluta para `awsauth.env` el archivo de entorno que corresponda a la `AWS_CREDS` variable de entorno.

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

3. Desplácese a `cvo_aws_variable` la carpeta y actualice el acceso y la clave secreta en el archivo de credenciales.

   Agregue el siguiente contenido al archivo:

   aws_access_key_id=<> aws_secret_access_key=<>

   El formato **debe** ser exactamente como se muestra arriba.

Paso 4: Regístrese en NetApp Intelligent Services

Regístrese en NetApp Intelligent Services a través de su proveedor de nube para pagar por hora (PAYGO) o mediante un contrato anual. Los servicios inteligentes de NetApp incluyen NetApp Backup and Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience y NetApp Disaster Recovery. La clasificación de datos de NetApp está incluida en su suscripción sin costo adicional.

Pasos

1. Desde el portal de Amazon Web Services (AWS), navegue a **SaaS** y seleccione **Suscribirse a NetApp Intelligent Services**.

   Puede usar el mismo grupo de recursos que Cloud Volumes ONTAP o uno diferente.

2. Configure el portal de la consola de NetApp para importar la suscripción de SaaS a la consola.

   Puede configurarlo directamente desde el portal de AWS.

   Serás redirigido al portal de la consola para confirmar la configuración.

3. Confirme la configuración en el portal de la consola seleccionando **Guardar**.

Paso 5: Cree un volumen externo

Debe crear un volumen externo para mantener los archivos de estado de Terraform y otros archivos importantes persistentes. Debe asegurarse de que los archivos están disponibles para Terraform para ejecutar el flujo de trabajo y las implementaciones.

Pasos

1. Cree un volumen externo fuera de Docker Compose:

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

Ejemplo:

```
docker volume create cvo_aws_volume_dst
```

2. Utilice una de las siguientes opciones:

   a. Añada una ruta de volumen externo al `.env` archivo de entorno.

      Debe seguir el formato exacto que se muestra a continuación.

      Formato:

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

      Ejemplo:
      `PERSISTENT_VOL=cvo_aws_volume_dst:/cvo_aws`

   b. Añada recursos compartidos NFS como volumen externo.

      Asegúrese de que el contenedor de Docker se pueda comunicar con los recursos compartidos NFS y de que los permisos correctos, como lectura/escritura, están configurados.

      i. Agregue la ruta de acceso de recursos compartidos NFS como la ruta al volumen externo en el archivo Docker Compose, como se muestra a continuación: Formato:

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

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

3. Navegue a la `cvo_aws_variables` carpeta.

   Debe ver el siguiente archivo de variables en la carpeta:

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

4. Cambie los valores dentro del `terraform.tfvars` archivo de acuerdo con sus requisitos.

   Debe leer la documentación de soporte específica cuando modifique cualquiera de los valores de variables del `terraform.tfvars` archivo. Los valores pueden variar según la región, las zonas de disponibilidad y otros factores compatibles con Cloud Volumes ONTAP para AWS. Esto incluye licencias, tamaño de disco y tamaño de máquina virtual para nodos individuales y pares de alta disponibilidad.

   Todas las variables de soporte para el agente de la consola y los módulos Terraform de Cloud Volumes ONTAP ya están definidas en el `variables.tf` archivo. Debes hacer referencia a los nombres de las

variables en el `variables.tf` archivo antes de agregarlo al `terraform.tfvars` archivo.

5. En función de sus requisitos, puede activar o desactivar FlexCache and FlexClone configurando las siguientes opciones en `true` o `false`.

   Los siguientes ejemplos habilitan FlexCache y FlexClone:

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

Paso 6: Ponga en marcha Cloud Volumes ONTAP para AWS

Utilice los siguientes pasos para poner en marcha Cloud Volumes ONTAP para AWS.

Pasos

1. Desde la carpeta raíz, ejecute el siguiente comando para activar el despliegue:

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

   Se activan dos contenedores, el primer contenedor pone en marcha Cloud Volumes ONTAP y el segundo contenedor envía datos de telemetría a AutoSupport.

   El segundo contenedor espera hasta que el primer contenedor complete todos los pasos correctamente.

2. Supervise el progreso del proceso de despliegue mediante los archivos log:

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

   Este comando proporciona resultados en tiempo real y captura los datos en los siguientes archivos de registro:
   `deployment.log`

   `telemetry_asup.log`

   Puede cambiar el nombre de estos archivos de registro editando `.env` el archivo mediante las siguientes variables de entorno:

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   Los siguientes ejemplos muestran cómo cambiar los nombres de los archivos log:

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Después de terminar

Puede utilizar los siguientes pasos para eliminar el entorno temporal y limpiar los elementos creados durante

el proceso de despliegue.

Pasos

1. Si implementó FlexCache, configure la siguiente opción en `terraform.tfvars` el archivo de variables, esto limpia los volúmenes de FlexCache y elimina el entorno temporal que se creó anteriormente.

   ```
   flexcache_operation = "destroy"
   ```

   Las opciones posibles son `deploy` y. `destroy`

2. Si implementó FlexClone, configure la siguiente opción en `terraform.tfvars` el archivo de variables, esto limpia los volúmenes de FlexClone y elimina el entorno temporal que se creó anteriormente.

   ```
   flexclone_operation = "destroy"
   ```

   Las opciones posibles son `deploy` y. `destroy`

Cloud Volumes ONTAP para Azure

Cloud Volumes ONTAP para Azure: Ráfagas en la nube

Este artículo es compatible con la solución de automatización NetApp Cloud Volumes ONTAP para Azure, que está disponible para los clientes de NetApp desde el centro de automatización de la NetApp Console .

La solución de automatización de Cloud Volumes ONTAP para Azure automatiza la implementación en contenedores de Cloud Volumes ONTAP para Azure mediante Terraform, lo que le permite implementar Cloud Volumes ONTAP para Azure rápidamente, sin ninguna intervención manual.

Antes de empezar

- Debes descargar el "Cloud Volumes ONTAP Azure: Ráfagas a la nube" Solución de automatización a través de la interfaz web de la consola. La solución se presenta empaquetada como `CVO-Azure-Burst-To-Cloud.zip`.
- Debe instalar una máquina virtual Linux en la misma red que Cloud Volumes ONTAP.
- Después de instalar la máquina virtual Linux, debe seguir los pasos de esta solución para instalar las dependencias necesarias.

Paso 1: Instalar Docker y Docker Compose

Instale Docker

Los siguientes pasos usan el software de distribución Linux Ubuntu 20,04 como ejemplo. Los comandos que ejecute dependen del software de distribución de Linux que utilice. Consulte la documentación específica del software de distribución de Linux para conocer la configuración.

Pasos

1. Instale Docker ejecutando los siguientes `sudo` comandos:

```
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. Compruebe la instalación:

```
docker –version
```

3. Compruebe que se ha creado un grupo denominado «docker» en su sistema Linux. Si es necesario, cree el grupo:

```
sudo groupadd docker
```

4. Agregue el usuario que necesita acceder a Docker al grupo:

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

5. Los cambios se aplican después de cerrar la sesión y volver a conectarse al terminal. Como alternativa, puede aplicar los cambios inmediatamente:

```
newgrp docker
```

Instale Docker Compose

Pasos

1. Instale Docker Compose ejecutando los siguientes `sudo` comandos:

```
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. Compruebe la instalación:

```
docker-compose –version
```

Paso 2: Preparar la imagen de Docker

Pasos

1. Copie `CVO-Azure-Burst-To-Cloud.zip` la carpeta en la máquina virtual Linux que desee utilizar para implementar Cloud Volumes ONTAP:

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

   - `private-key.pem` es su archivo de clave privada para iniciar sesión sin contraseña.
   - `azureuser` Es el nombre de usuario de la máquina virtual.
   - `IP_ADDRESS_OF_VM` Es la dirección IP del equipo virtual.
   - `LOCATION_TO_BE_COPIED` es la ubicación donde se copiará la carpeta.

2. Extraiga la `CVO-Azure-Burst-To-Cloud.zip` carpeta. Puede extraer la carpeta en el directorio actual o en una ubicación personalizada.

   Para extraer la carpeta del directorio actual, ejecute:

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

   Para extraer la carpeta en una ubicación personalizada, ejecute:

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

3. Después de extraer el contenido, desplácese a la `CVO_Azure_Deployment` carpeta y ejecute el siguiente comando para ver los archivos:

```
 ls -la
```

   Debería ver una lista de archivos, similar al siguiente ejemplo:

```
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. Busque el `cvo_azure_flexcache_ubuntu_image_latest.tar.gz` archivo. Esto contiene la imagen de Docker necesaria para poner en marcha Cloud Volumes ONTAP para Azure.
5. Abra el archivo:

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

6. Espere unos minutos hasta que se cargue la imagen de Docker y, a continuación, valide que la imagen de Docker se haya cargado correctamente:

```
docker images
```

Debería ver una imagen de Docker llamada `cvo_azure_flexcache_ubuntu_image_latest` con la `latest` etiqueta, como se muestra en el siguiente ejemplo:

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

Paso 3: Crear archivos de variables de entorno

En esta etapa, debes crear dos archivos de variables de entorno. Un archivo es para la autenticación de las API de Azure Resource Manager mediante credenciales de entidad de servicio. El segundo archivo sirve para configurar variables de entorno para permitir que los módulos Terraform de la consola localicen y autentiquen las API de Azure.

Pasos

1. Cree un principal de servicio.

Antes de crear los archivos de variables de entorno, debe crear un principal de servicio siguiendo los pasos de "Cree una aplicación de Azure Active Directory y un director de servicio que pueda acceder a los recursos".

2. Asigne el rol **Contributor** al principal de servicio recién creado.
3. Crear un rol personalizado.
    a. Localice el `sp_role.json` archivo y compruebe los permisos necesarios en las acciones enumeradas.
    b. Inserte estos permisos y adjunte el rol personalizado al principal de servicio recién creado.
4. Vaya a **Certificados y secretos** y seleccione **Nuevo secreto de cliente** para crear el secreto de cliente.

    Cuando creas el secreto del cliente, debes registrar los detalles de la columna **VALOR** porque no podrás ver este valor de nuevo. También debe registrar la siguiente información:

    - ID del cliente
    - ID de suscripción
    - ID de inquilino

    Necesitará esta información para crear las variables de entorno. Puede encontrar la información de ID de cliente e ID de inquilino en la sección **Overview** de la interfaz de usuario principal de servicio.
5. Cree los archivos de entorno.
    a. Cree el `azureauth.env` archivo en la siguiente ubicación:

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

        i. Agregue el siguiente contenido al archivo:

        ClientID=<> clientSecret=<> SubscriptionId=<> tenantId=<>

    El formato **debe** ser exactamente como se muestra arriba sin ningún espacio entre la clave y el valor.

    b. Cree el `credentials.env` archivo en la siguiente ubicación:

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

        i. Agregue el siguiente contenido al archivo:

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

    El formato **debe** ser exactamente como se muestra arriba sin ningún espacio entre la clave y el valor.
6. Agregue las rutas de acceso absolutas al `.env` archivo.

    Introduzca la ruta de acceso absoluta para `azureauth.env` el archivo de entorno en el `.env` archivo que corresponda a la `AZURE_RM_CREDS` variable de entorno.

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

    Introduzca la ruta de acceso absoluta para `credentials.env` el archivo de entorno en el `.env` archivo que corresponda a la `BLUEXP_TF_AZURE_CREDS` variable de entorno.

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

Paso 4: Regístrese en NetApp Intelligent Services

Regístrese en NetApp Intelligent Services a través de su proveedor de nube para pagar por hora (PAYGO) o mediante un contrato anual. Los servicios inteligentes de NetApp incluyen NetApp Backup and Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience y NetApp Disaster Recovery. La clasificación de datos de NetApp está incluida en su suscripción sin costo adicional

Pasos

1. Desde el portal de Azure, navegue hasta **SaaS** y seleccione **Suscribirse a NetApp Intelligent Services**.
2. Seleccione el plan **Cloud Manager (por Cap PYGO por hora, WORM y servicios de datos)**.

   Puede usar el mismo grupo de recursos que Cloud Volumes ONTAP o uno diferente.

3. Configure el portal de la consola para importar la suscripción de SaaS a la consola.

   Puede configurarlo directamente desde el portal de Azure navegando a **Detalles del producto y del plan** y seleccionando la opción **Configurar cuenta ahora**.

   Luego será redirigido al portal de la consola para confirmar la configuración.

4. Confirme la configuración en el portal de la consola seleccionando **Guardar**.

Paso 5: Cree un volumen externo

Debe crear un volumen externo para mantener los archivos de estado de Terraform y otros archivos importantes persistentes. Debe asegurarse de que los archivos están disponibles para Terraform para ejecutar el flujo de trabajo y las implementaciones.

Pasos

1. Cree un volumen externo fuera de Docker Compose:

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

   Ejemplo:

   ```
   docker volume create cvo_azure_volume_dst
   ```

2. Utilice una de las siguientes opciones:

   a. Añada una ruta de volumen externo al `.env` archivo de entorno.

      Debe seguir el formato exacto que se muestra a continuación.

      Formato:

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

      Ejemplo:
      `PERSISTENT_VOL=cvo_azure_volume_dst:/cvo_azure`

b. Añada recursos compartidos NFS como volumen externo.

   Asegúrese de que el contenedor de Docker se pueda comunicar con los recursos compartidos NFS y de que los permisos correctos, como lectura/escritura, están configurados.

   i. Agregue la ruta de acceso de recursos compartidos NFS como la ruta al volumen externo en el archivo Docker Compose, como se muestra a continuación: Formato:

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

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

3. Navegue a la `cvo_azure_variables` carpeta.

   Debe ver los siguientes archivos de variables en la carpeta:

   `terraform.tfvars`

   `variables.tf`

4. Cambie los valores dentro del `terraform.tfvars` archivo de acuerdo con sus requisitos.

   Debe leer la documentación de soporte específica cuando modifique cualquiera de los valores de variables del `terraform.tfvars` archivo. Los valores pueden variar según la región, las zonas de disponibilidad y otros factores compatibles con Cloud Volumes ONTAP para Azure. Esto incluye licencias, tamaño de disco y tamaño de máquina virtual para nodos individuales y pares de alta disponibilidad.

   Todas las variables de soporte para el agente de la consola y los módulos Terraform de Cloud Volumes ONTAP ya están definidas en el `variables.tf` archivo. Debes hacer referencia a los nombres de las variables en el `variables.tf` archivo antes de agregarlo al `terraform.tfvars` archivo.

5. En función de sus requisitos, puede activar o desactivar FlexCache and FlexClone configurando las siguientes opciones en `true` o `false`.

   Los siguientes ejemplos habilitan FlexCache y FlexClone:

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

6. Si es necesario, puede recuperar el valor de la variable Terraform `az_service_principal_object_id` desde Azure Active Directory Service:

   a. Vaya a **Enterprise Applications –> All Applications** y seleccione el nombre del Service Principal que creó anteriormente.

   b. Copie el ID del objeto e inserte el valor para la variable Terraform:

   ```
   az_service_principal_object_id
   ```

Paso 6: Implemente Cloud Volumes ONTAP para Azure

Utilice los siguientes pasos para implementar Cloud Volumes ONTAP para Azure.

Pasos

1. Desde la carpeta raíz, ejecute el siguiente comando para activar el despliegue:

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

   Se activan dos contenedores, el primer contenedor pone en marcha Cloud Volumes ONTAP y el segundo contenedor envía datos de telemetría a AutoSupport.

   El segundo contenedor espera hasta que el primer contenedor complete todos los pasos correctamente.

2. Supervise el progreso del proceso de despliegue mediante los archivos log:

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

   Este comando proporciona resultados en tiempo real y captura los datos en los siguientes archivos de registro:

   `deployment.log`

   `telemetry_asup.log`

   Puede cambiar el nombre de estos archivos de registro editando `.env` el archivo mediante las siguientes variables de entorno:

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   Los siguientes ejemplos muestran cómo cambiar los nombres de los archivos log:

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Después de terminar

Puede utilizar los siguientes pasos para eliminar el entorno temporal y limpiar los elementos creados durante el proceso de despliegue.

Pasos

1. Si implementó FlexCache, configure la siguiente opción en `terraform.tfvars` el archivo, esto limpia los volúmenes de FlexCache y elimina el entorno temporal creado anteriormente.

   `flexcache_operation = "destroy"`

   Las opciones posibles son `deploy` y. `destroy`

2. Si implementó FlexClone, configure la siguiente opción en `terraform.tfvars` el archivo, esto limpia los volúmenes de FlexClone y elimina el entorno temporal creado anteriormente.

```
flexclone_operation = "destroy"
```

Las opciones posibles son `deploy` y. `destroy`

Cloud Volumes ONTAP para Google Cloud

Cloud Volumes ONTAP para Google Cloud: Estallido a la nube

Este artículo es compatible con la solución de automatización NetApp Cloud Volumes ONTAP para Google Cloud, que está disponible para los clientes de NetApp desde el centro de automatización de la NetApp Console .

La solución de automatización de Cloud Volumes ONTAP para Google Cloud automatiza la puesta en marcha en contenedores de Cloud Volumes ONTAP para Google Cloud, lo que le permite poner en marcha Cloud Volumes ONTAP para Google Cloud rápidamente sin intervención manual.

Antes de empezar

- Debes descargar el "Cloud Volumes ONTAP para Google Cloud: Estallido a la nube" Solución de automatización a través de la interfaz web de la consola. La solución se presenta empaquetada como `cvo_gcp_flexcache.zip`.
- Debe instalar una máquina virtual Linux en la misma red que Cloud Volumes ONTAP.
- Después de instalar la máquina virtual Linux, debe seguir los pasos de esta solución para instalar las dependencias necesarias.

Paso 1: Instalar Docker y Docker Compose

Instale Docker

Los siguientes pasos usan el software de distribución Linux Ubuntu 20,04 como ejemplo. Los comandos que ejecute dependen del software de distribución de Linux que utilice. Consulte la documentación específica del software de distribución de Linux para conocer la configuración.

Pasos

1. Instale Docker ejecutando los siguientes comandos:

```
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. Compruebe la instalación:

```
docker –version
```

3. Compruebe que se ha creado un grupo denominado «docker» en su sistema Linux. Si es necesario, cree el grupo:

```
sudo groupadd docker
```

4. Agregue el usuario que necesita acceder a Docker al grupo:

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

5. Los cambios se aplican después de cerrar la sesión y volver a conectarse al terminal. Como alternativa, puede aplicar los cambios inmediatamente:

```
newgrp docker
```

Instale Docker Compose

Pasos

1. Instale Docker Compose ejecutando los siguientes `sudo` comandos:

```
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. Compruebe la instalación:

```
docker-compose –version
```

Paso 2: Preparar la imagen de Docker

Pasos

1. Copie `cvo_gcp_flexcache.zip` la carpeta en la máquina virtual Linux que desee utilizar para implementar Cloud Volumes ONTAP:

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

- `private-key.pem` es su archivo de clave privada para iniciar sesión sin contraseña.
- `gcpuser` Es el nombre de usuario de la máquina virtual.
- `IP_ADDRESS_OF_VM` Es la dirección IP del equipo virtual.
- `LOCATION_TO_BE_COPIED` es la ubicación donde se copiará la carpeta.

2. Extraiga la `cvo_gcp_flexcache.zip` carpeta. Puede extraer la carpeta en el directorio actual o en una ubicación personalizada.

   Para extraer la carpeta del directorio actual, ejecute:

   ```
   unzip cvo_gcp_flexcache.zip
   ```

   Para extraer la carpeta en una ubicación personalizada, ejecute:

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

3. Después de extraer el contenido, ejecute el siguiente comando para ver los archivos:

   ```
    ls -la
   ```

   Debería ver una lista de archivos, similar al siguiente ejemplo:

   ```
       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. Busque el `cvo_gcp_flexcache_ubuntu_image.tar` archivo. Esto contiene la imagen de Docker necesaria para poner en marcha Cloud Volumes ONTAP para Google Cloud.
5. Abra el archivo:

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

6. Espere unos minutos hasta que se cargue la imagen de Docker y, a continuación, valide que la imagen de Docker se haya cargado correctamente:

```
docker images
```

Debería ver una imagen de Docker llamada `cvo_gcp_flexcache_ubuntu_image` con la `latest` etiqueta, como se muestra en el siguiente ejemplo:

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

Puede cambiar el nombre de la imagen de Docker si es necesario. Si cambia el nombre de la imagen de Docker, asegúrese de actualizar el nombre de la imagen de Docker en los `docker-compose-deploy` archivos y. `docker-compose-destroy`

Paso 3: Actualice el archivo JSON

En esta etapa, debe actualizar `cxo-automation-gcp.json` el archivo con una clave de cuenta de servicio para autenticar al proveedor de Google Cloud.

1. Cree una cuenta de servicio con permisos para implementar Cloud Volumes ONTAP y un agente de consola"Obtenga más información sobre la creación de cuentas de servicio."
2. Descargue el archivo de claves de la cuenta y actualice `cxo-automation-gcp.json` el archivo con la información del archivo de claves. `cxo-automation-gcp.json`El archivo se encuentra en `cvo_gcp_variables` la carpeta.

Ejemplo

```
{
  "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"
}
```

El formato de archivo debe ser exactamente como se muestra anteriormente.

Paso 4: Regístrese en NetApp Intelligent Services

Regístrese en NetApp Intelligent Services a través de su proveedor de nube para pagar por hora (PAYGO) o mediante un contrato anual. Los servicios inteligentes de NetApp incluyen NetApp Backup and Recovery, Cloud Volumes ONTAP, NetApp Cloud Tiering, NetApp Ransomware Resilience y NetApp Disaster Recovery. La clasificación de datos de NetApp está incluida en su suscripción sin costo adicional.

Pasos

1. Navegar hasta el"Consola de Google Cloud" y seleccione *Suscribirse a los servicios inteligentes de NetApp *.
2. Configure el portal de la consola de NetApp para importar la suscripción de SaaS a la consola.

   Puede configurar esto directamente desde Google Cloud Platform. Serás redirigido al portal de la consola para confirmar la configuración.

3. Confirme la configuración en el portal de la consola seleccionando **Guardar**.

Para obtener más información, consulte "Administrar credenciales y suscripciones de Google Cloud para la consola de NetApp" .

Paso 5: Habilita las API de Google Cloud necesarias

Debe habilitar las siguientes API de Google Cloud en su proyecto para implementar Cloud Volumes ONTAP y el agente de la consola.

- API de Cloud Deployment Manager V2
- API de registro en la nube
- API de Cloud Resource Manager
- API del motor de computación
- API de gestión de acceso e identidad (IAM)

"Obtenga más información sobre cómo habilitar las API"

Paso 6: Cree un volumen externo

Debe crear un volumen externo para mantener los archivos de estado de Terraform y otros archivos importantes persistentes. Debe asegurarse de que los archivos están disponibles para Terraform para ejecutar el flujo de trabajo y las implementaciones.

Pasos

1. Cree un volumen externo fuera de Docker Compose:

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

   Ejemplo:

   ```
   docker volume create cvo_gcp_volume_dst
   ```

2. Utilice una de las siguientes opciones:

   a. Añada una ruta de volumen externo al `.env` archivo de entorno.

      Debe seguir el formato exacto que se muestra a continuación.

      Formato:

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

      Ejemplo:
      `PERSISTENT_VOL=cvo_gcp_volume_dst:/cvo_gcp`

   b. Añada recursos compartidos NFS como volumen externo.

      Asegúrese de que el contenedor de Docker se pueda comunicar con los recursos compartidos NFS y de que los permisos correctos, como lectura/escritura, están configurados.

      i. Agregue la ruta de acceso de recursos compartidos NFS como la ruta al volumen externo en el archivo Docker Compose, como se muestra a continuación: Formato:

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

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

3. Navegue a la `cvo_gcp_variables` carpeta.

   Debe ver los siguientes archivos en la carpeta:

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

4. Cambie los valores dentro del `terraform.tfvars` archivo de acuerdo con sus requisitos.

   Debe leer la documentación de soporte específica cuando modifique cualquiera de los valores de variables del `terraform.tfvars` archivo. Los valores pueden variar en función de la región, las zonas de disponibilidad y otros factores compatibles con Cloud Volumes ONTAP para Google Cloud. Esto incluye licencias, tamaño de disco y tamaño de máquina virtual para nodos individuales y pares de alta disponibilidad.

   Todas las variables de soporte para el agente de la consola y los módulos Terraform de Cloud Volumes ONTAP ya están definidas en el `variables.tf` archivo. Debes hacer referencia a los nombres de las variables en el `variables.tf` archivo antes de agregarlo al `terraform.tfvars` archivo.

5. En función de sus requisitos, puede activar o desactivar FlexCache and FlexClone configurando las siguientes opciones en `true` o `false`.

   Los siguientes ejemplos habilitan FlexCache y FlexClone:

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

Paso 7: Implementa Cloud Volumes ONTAP para Google Cloud

Siga estos pasos para implementar Cloud Volumes ONTAP para Google Cloud.

Pasos

1. Desde la carpeta raíz, ejecute el siguiente comando para activar el despliegue:

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

   Se activan dos contenedores, el primer contenedor pone en marcha Cloud Volumes ONTAP y el segundo contenedor envía datos de telemetría a AutoSupport.

   El segundo contenedor espera hasta que el primer contenedor complete todos los pasos correctamente.

2. Supervise el progreso del proceso de despliegue mediante los archivos log:

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

   Este comando proporciona resultados en tiempo real y captura los datos en los siguientes archivos de registro:
   `deployment.log`

   `telemetry_asup.log`

   Puede cambiar el nombre de estos archivos de registro editando `.env` el archivo mediante las siguientes variables de entorno:

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   Los siguientes ejemplos muestran cómo cambiar los nombres de los archivos log:

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Después de terminar

Puede utilizar los siguientes pasos para eliminar el entorno temporal y limpiar los elementos creados durante el proceso de despliegue.

Pasos

1. Si implementó FlexCache, configure la siguiente opción en `terraform.tfvars` el archivo, esto limpia los volúmenes de FlexCache y elimina el entorno temporal creado anteriormente.

   `flexcache_operation = "destroy"`

   Las opciones posibles son `deploy` y. `destroy`

2. Si implementó FlexClone, configure la siguiente opción en `terraform.tfvars` el archivo, esto limpia los volúmenes de FlexClone y elimina el entorno temporal creado anteriormente.

```
flexclone_operation = "destroy"
```

Las opciones posibles son `deploy` y. `destroy`

ONTAP

Día 0/1

Descripción general de la solución ONTAP Day 0/1

Puede utilizar la solución de automatización ONTAP día 0/1 para implementar y configurar un clúster ONTAP utilizando Ansible. La solución está disponible en "Centro de automatización de la NetApp Console".

Opciones flexibles de puesta en marcha de ONTAP

En función de sus requisitos, se puede usar hardware en las instalaciones o simular ONTAP para poner en marcha y configurar un clúster de ONTAP con Ansible.

Hardware en las instalaciones

Puede poner en marcha esta solución con hardware en las instalaciones que ejecute ONTAP, como un sistema FAS o AFF. Debe utilizar una máquina virtual Linux para poner en marcha y configurar el clúster de ONTAP mediante Ansible.

Simular ONTAP

Para implementar esta solución mediante un simulador de ONTAP, debe descargar la versión más reciente de Simulate ONTAP del sitio de soporte de NetApp. Simulate ONTAP es un simulador virtual para el software ONTAP. Simulate ONTAP se ejecuta en un hipervisor de VMware en un sistema Windows, Linux o Mac. Para hosts Windows y Linux, debe usar el hipervisor de VMware Workstation para ejecutar esta solución. Si tiene un sistema operativo Mac, utilice el hipervisor VMware Fusion.

Diseño en capas

El marco de Ansible simplifica el desarrollo y la reutilización de las tareas lógicas y de ejecución de automatización. El marco hace una distinción entre las tareas de toma de decisiones (capa lógica) y los pasos de ejecución (capa de ejecución) en la automatización. Comprender cómo funcionan estas capas le permite personalizar la configuración.

Un «libro de estrategia» de Ansible ejecuta una serie de tareas que van de principio a fin. `site.yml`el libro de estrategia contiene `logic.yml` el libro de estrategia y `execution.yml` el libro de estrategia.

Cuando se ejecuta una solicitud, `site.yml` el libro de aplicaciones realiza primero una llamada al `logic.yml` libro de aplicaciones y, a continuación, llama a él `execution.yml` para ejecutar la solicitud de servicio.

No es necesario que utilice la capa lógica del marco. La capa lógica proporciona opciones para ampliar la capacidad del marco más allá de los valores codificados para la ejecución. Esto le permite personalizar las

capacidades del marco si es necesario.

Capa lógica

La capa lógica consta de lo siguiente:

- El `logic.yml` libro de estrategia
- Archivos de tareas lógicas dentro del `logic-tasks` directorio

La capa lógica proporciona la capacidad de tomar decisiones complejas sin la necesidad de una integración personalizada significativa (por ejemplo, conectarse a ServiceNow). La capa lógica es configurable y proporciona la entrada a los microservicios.

También se proporciona la capacidad de omitir la capa lógica. Si desea omitir la capa lógica, no defina la `logic_operation` variable. La invocación directa `logic.yml` del libro de estrategia proporciona la capacidad de realizar algún nivel de depuración sin ejecución. Puede utilizar una sentencia de depuración para verificar que el valor de `raw_service_request` es correcto.

Consideraciones importantes:

- El `logic.yml` libro de estrategia buscará `logic_operation` la variable. Si la variable está definida en la solicitud, carga un archivo de tareas desde el `logic-tasks` directorio. El archivo de tarea debe ser un archivo .yml. Si no hay ningún archivo de tarea coincidente y la `logic_operation` variable está definida, la capa lógica falla.
- El valor por defecto de `logic_operation` la variable es `no-op`. Si la variable no está definida de forma explícita, se establece por defecto en `no-op`, que no ejecuta ninguna operación.
- Si la `raw_service_request` variable ya está definida, la ejecución continúa a la capa de ejecución. Si la variable no está definida, la capa lógica falla.

Capa de ejecución

La capa de ejecución consta de lo siguiente:

- El `execution.yml` libro de estrategia

La capa de ejecución realiza las llamadas API para configurar un clúster de ONTAP. El `execution.yml` playbook requiere que la `raw_service_request` variable se defina al ejecutarse.

Soporte para personalización

Puede personalizar esta solución de varias maneras en función de sus requisitos.

Las opciones de personalización incluyen:

- Modificar libros de estrategia de Ansible
- Agregar roles

Personalizar los archivos de Ansible

La siguiente tabla describe los archivos Ansible personalizables que contiene esta solución.

Ubicación	Descripción
`playbooks/inventory/hosts`	Contiene un único archivo con una lista de hosts y grupos.
`playbooks/group_vars/all/*`	Ansible proporciona una forma cómoda de aplicar variables a varios hosts a la vez. Puede modificar cualquiera o todos los archivos de esta carpeta, incluidos `cfg.yml`, `clusters.yml`, , `defaults.yml` `services.yml` `standards.yml` , , y `vault.yml`.
`playbooks/logic-tasks`	Admite tareas de toma de decisiones en Ansible y mantiene la separación de la lógica y la ejecución. Puede agregar archivos a esta carpeta que se correspondan con el servicio pertinente.
`playbooks/vars/*`	Los valores dinámicos utilizados en los libros de estrategia y roles de Ansible para permitir la personalización, flexibilidad y reutilización de las configuraciones. Si es necesario, puede modificar cualquiera o todos los archivos de esta carpeta.

Personalizar roles

También puede personalizar la solución agregando o cambiando funciones de Ansible, también denominadas microservicios. Para obtener más información, consulte "Personalizar".

Prepárese para utilizar la solución ONTAP day 0/1

Antes de poner en marcha la solución de automatización, debe preparar el entorno de ONTAP e instalar y configurar Ansible.

Consideraciones de planificación inicial

Debe revisar los siguientes requisitos y consideraciones antes de usar esta solución para poner en marcha un clúster de ONTAP.

Requisitos básicos

Debe cumplir los siguientes requisitos básicos para utilizar esta solución:

- Tiene que tener acceso al software ONTAP, ya sea en las instalaciones o a través de un simulador ONTAP.
- Debe saber cómo utilizar el software ONTAP.
- Debes saber cómo utilizar las herramientas de software de automatización de Ansible.

Consideraciones DE PLANIFICACIÓN

Antes de implementar esta solución de automatización, debe decidir:

- La ubicación donde se va a ejecutar el nodo de control de Ansible.
- El sistema ONTAP, ya sea hardware en las instalaciones o un simulador ONTAP.
- Independientemente de si necesitará personalización o no.

Prepare el sistema ONTAP

Tanto si utiliza un sistema ONTAP on-premises o Simulate ONTAP, debe preparar el entorno antes de poder implementar la solución de automatización.

Opcionalmente, instale y configure Simulate ONTAP

Si desea implementar esta solución mediante un simulador de ONTAP, debe descargar y ejecutar Simulate ONTAP.

Antes de empezar

- Debe descargar e instalar el hipervisor de VMware que va a utilizar para ejecutar Simulate ONTAP.
  - Si tiene un sistema operativo Windows o Linux, utilice VMware Workstation.
  - Si tiene un sistema operativo Mac, utilice VMware Fusion.

> Si utiliza un sistema operativo Mac, debe tener un procesador Intel.

Pasos

Utilice el siguiente procedimiento para instalar dos simuladores de ONTAP en su entorno local:

1. Descargar Simulate ONTAP desde la "Sitio de soporte de NetApp".

   > Aunque instale dos simuladores ONTAP, solo necesita descargar una copia del software.

2. Si todavía no está en ejecución, inicie su aplicación VMware.
3. Busque el archivo del simulador descargado y haga clic con el botón derecho del ratón para abrirlo con la aplicación VMware.
4. Defina el nombre de la primera instancia de ONTAP.
5. Espere a que se arranque el simulador y siga las instrucciones para crear un clúster de un solo nodo.

   Repita los pasos para la segunda instancia de ONTAP.

6. Opcionalmente, agregue un complemento de disco completo.

   Desde cada clúster, ejecute los siguientes comandos:

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

Estado del sistema ONTAP

Debe comprobar el estado inicial del sistema ONTAP, ya sea en las instalaciones o ejecutándose a través de un simulador de ONTAP.

Compruebe que se cumplen los siguientes requisitos del sistema ONTAP:

- ONTAP se ha instalado y se está ejecutando sin definir ningún clúster aún.
- La ONTAP se arranca y muestra la dirección IP para acceder al clúster.
- Se puede acceder a la red.

- Tiene credenciales de administrador.
- Se muestra el banner del mensaje del día (MOTD) con la dirección de administración.

Instale el software de automatización necesario

En esta sección se ofrece información sobre cómo instalar Ansible y preparar la solución de automatización para la puesta en marcha.

Instale Ansible

Ansible se puede instalar en sistemas Linux o Windows.

El método de comunicación predeterminado que utiliza Ansible para comunicarse con un clúster de ONTAP es SSH.

Consulte "Introducción a NetApp y Ansible: Instale Ansible"para instalar Ansible.

Ansible debe instalarse en el nodo de control del sistema.

Descarga y prepara la solución de automatización

Puedes usar los siguientes pasos para descargar y preparar la solución de automatización para la puesta en marcha.

1. Descarga el "ONTAP - Día 0/1 Comprobaciones de estado" Solución de automatización a través de la interfaz web de la consola. La solución se presenta empaquetada como `ONTAP_DAY0_DAY1.zip`.
2. Extraiga la carpeta zip y copie los archivos en la ubicación deseada en el nodo de control dentro del entorno de Ansible.

Configuración de marco de Ansible inicial

Realice la configuración inicial del marco Ansible:

1. Navegar a `playbooks/inventory/group_vars/all`.
2. Descifre el `vault.yml` archivo:

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

   Cuando se le solicite la contraseña del almacén, introduzca la siguiente contraseña temporal:

   ```
   NetApp123!
   ```

   “NetApp123!” es una contraseña temporal para descifrar el `vault.yml` archivo y la contraseña del almacén correspondiente. Después de su primer uso, * debe * cifrar el archivo con su propia contraseña.

3. Modifique los siguientes archivos de Ansible:

   - `clusters.yml` - Modificar los valores de este archivo para adaptarse a su entorno.
   - `vault.yml` - Después de descifrar el archivo, modifique el clúster de ONTAP, los valores de nombre de usuario y contraseña para adaptarse a su entorno.

- `cfg.yml` - Establecer la ruta del archivo para `log2file` y establecer `show_request` debajo de `cfg True` para mostrar el `raw_service_request`.

  `raw_service_request`La variable se muestra en los archivos log y durante la ejecución.

  Cada archivo enumerado contiene comentarios con instrucciones sobre cómo modificarlo de acuerdo con sus requisitos.

4. Vuelva a cifrar el `vault.yml` archivo:

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

   Se le pedirá que elija una nueva contraseña para el almacén tras el cifrado.

5. Navegue hasta `playbooks/inventory/hosts` y establezca un intérprete de Python válido.
6. Despliegue el `framework_test` servicio:

   El siguiente comando ejecuta el `na_ontap_info` módulo con un `gather_subset` valor de `cluster_identity_info`. Esto valida que la configuración básica es correcta y verifica que pueda comunicarse con el clúster.

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

   Ejecute el comando para cada clúster.

   Si es correcto, debería ver un resultado similar al siguiente ejemplo:

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

Ponga en marcha el clúster de ONTAP utilizando la solución

Después de completar la preparación y la planificación, estará listo para utilizar la solución ONTAP day 0/1 para configurar rápidamente un clúster ONTAP con Ansible.

En cualquier momento durante los pasos de esta sección, puede elegir probar una solicitud en lugar de ejecutarla realmente. Para probar una solicitud, cambie la `site.yml` tableta playbook en la línea de comandos a `logic.yml`.

`docs/tutorial-requests.txt`La ubicación contiene la versión final de todas las solicitudes de servicio utilizadas en este procedimiento. Si tiene dificultades para ejecutar una solicitud de servicio, puede copiar la solicitud pertinente del `tutorial-requests.txt` archivo a la `playbooks/inventory/group_vars/all/tutorial-requests.yml` ubicación y modificar los valores codificados según sea necesario (dirección IP, nombres agregados, etc.). A continuación, debería poder ejecutar correctamente la solicitud.

Antes de empezar

- Debe tener Ansible instalado.
- Debe haber descargado la solución ONTAP day 0/1 y extraído la carpeta a la ubicación deseada en el nodo de control de Ansible.
- El estado del sistema ONTAP debe cumplir con los requisitos y debe tener las credenciales necesarias.
- Debe haber completado todas las tareas requeridas descritas en la "Prepare" sección.

Los ejemplos de esta solución utilizan «Cluster_01» y «Cluster_02» como los nombres de los dos clústeres. Debe reemplazar estos valores por los nombres de los clústeres del entorno.

Paso 1: Configuración inicial del clúster

En esta etapa, debe realizar algunos pasos iniciales de configuración del clúster.

Pasos

1. Navegue a la `playbooks/inventory/group_vars/all/tutorial-requests.yml` ubicación y revise la `cluster_initial` solicitud en el archivo. Realice los cambios necesarios en su entorno.
2. Cree un archivo en `logic-tasks` la carpeta para la solicitud de servicio. Por ejemplo, cree un archivo `cluster_initial.yml` llamado .

   Copie las siguientes líneas en el nuevo archivo:

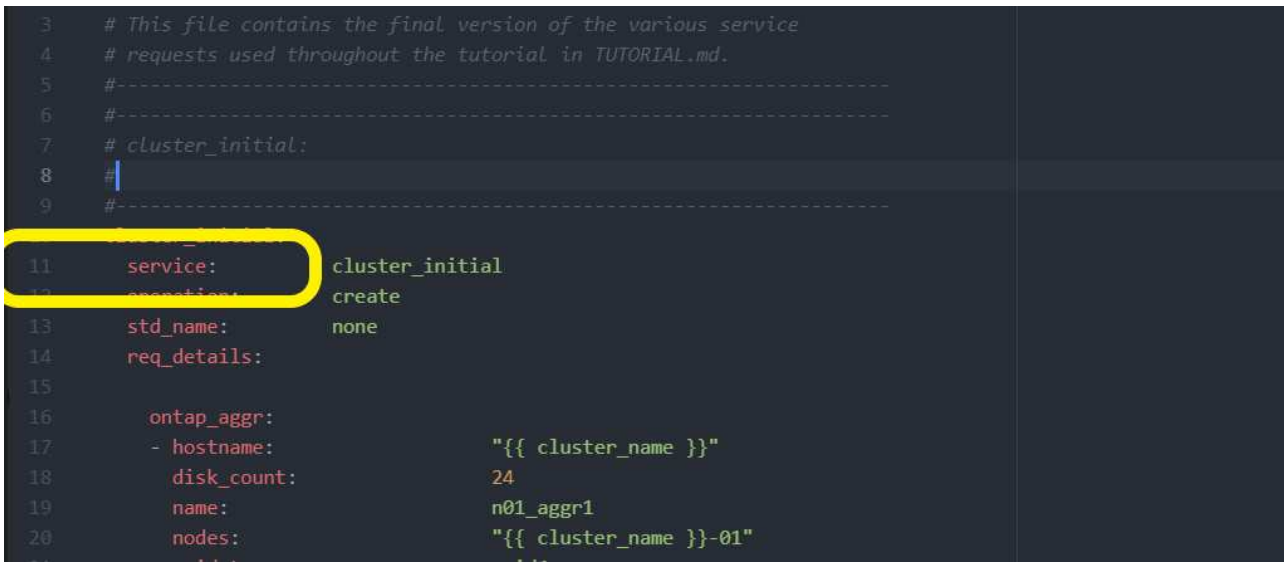
```
- name: Validate required inputs
  ansible.builtin.assert:
    that:
    - service is defined

- 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. Defina la `raw_service_request` variable.

   Puede utilizar una de las siguientes opciones para definir la `raw_service_request` variable en el `cluster_initial.yml` archivo creado en la `logic-tasks` carpeta:

   - **Opción 1**: Defina manualmente la `raw_service_request` variable.

     Abra el `tutorial-requests.yml` archivo con un editor y copie el contenido de la línea 11 a la línea 165. Pegue el contenido bajo `raw service request` la variable en el nuevo `cluster_initial.yml` archivo, como se muestra en los siguientes ejemplos:



Muestra el ejemplo

Archivo de ejemplo `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 }}"
```

- **Opción 2**: Usa una plantilla Jinja para definir la solicitud:

  También puede utilizar el siguiente formato de plantilla Jinja para obtener el `raw_service_request` valor.

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

4. Realice la configuración inicial del clúster para el primer clúster:

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

Compruebe que no haya errores antes de continuar.

5. Repita el comando para el segundo clúster:

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

Compruebe que no hay errores en el segundo clúster.

Cuando se desplaza hacia arriba hacia el principio de la salida de Ansible, debe ver la solicitud que se envió al marco, como se muestra en el siguiente ejemplo:

Muestra el ejemplo


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

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

6. Inicie sesión en cada instancia de ONTAP y compruebe que la solicitud se ha realizado correctamente.

Paso 2: Configurar las LIF de interconexión de clústeres

Ahora puede configurar las LIF de interconexión de clústeres añadiendo las definiciones de LIF a la `cluster_initial` solicitud y definiendo `ontap_interface` el microservicio.

La definición del servicio y la solicitud trabajan conjuntamente para determinar la acción:

- Si proporciona una solicitud de servicio para un microservicio que no está en las definiciones de servicio, la solicitud no se ejecuta.
- Si proporciona una solicitud de servicio con uno o más microservicios definidos en las definiciones de servicio, pero omitidos de la solicitud, la solicitud no se ejecuta.

El `execution.yml` libro de estrategia evalúa la definición del servicio escaneando la lista de microservicios en el orden indicado:

- Si hay una entrada en la solicitud con una clave de diccionario que coincida con la `args` entrada contenida en las definiciones de microservicio, se ejecuta la solicitud.
- Si no hay ninguna entrada coincidente en la solicitud de servicio, la solicitud se omite sin errores.

Pasos

1. Navegue hasta el `cluster_initial.yml` archivo que ha creado anteriormente y modifique la solicitud agregando las siguientes líneas a las definiciones de solicitud:

```
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. Ejecute el comando:

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

3. Inicie sesión en cada instancia para comprobar si las LIF se han agregado al clúster:

Muestra el ejemplo

```
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.
```

La salida muestra que las LIF fueron **NOT** agregadas. Esto se debe a que el `ontap_interface` microservicio todavía necesita definirse en el `services.yml` archivo.

4. Compruebe que las LIF se han añadido a `raw_service_request` la variable.

Muestra el ejemplo

En el ejemplo siguiente se muestra que las LIF se han agregado a la solicitud:


```
            "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. Defina el `ontap_interface` microservicio en en `cluster_initial` el `services.yml` archivo.

   Copie las siguientes líneas en el archivo para definir el microservicio:

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

6. Ahora que el `ontap_interface` microservicio se ha definido en la solicitud y en `services.yml` el archivo, vuelva a ejecutar la solicitud:

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

7. Inicie sesión en cada instancia de ONTAP y compruebe que las LIF se han añadido.

Paso 3: De manera opcional, configure varios clústeres

Si es necesario, puede configurar varios clústeres en la misma solicitud. Cuando defina la solicitud, debe proporcionar nombres de variables para cada clúster.

Pasos

1. Agregue una entrada para el segundo clúster del `cluster_initial.yml` archivo para configurar ambos clústeres en la misma solicitud.

   El siguiente ejemplo muestra el `ontap_aggr` campo después de agregar la segunda entrada.

```
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. Aplique los cambios para todos los demás elementos en `cluster_initial`.
3. Agregue la interconexión de clústeres a la solicitud copiando las siguientes líneas en el archivo:

```
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. Ejecute la solicitud de Ansible:

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

Paso 4: Configuración inicial de SVM

En esta etapa del procedimiento, configurará las SVM en el clúster.

Pasos

1. Actualice `svm_initial` la solicitud del `tutorial-requests.yml` archivo para configurar una relación entre iguales de SVM y SVM.

   Debe configurar lo siguiente:

   - La SVM

- La relación entre iguales de SVM
- La interfaz de SVM para cada SVM

2. Actualice las definiciones de variables en las `svm_initial` definiciones de solicitud. Debe modificar las siguientes definiciones de variables:
   - `cluster_name`
   - `vserver_name`
   - `peer_cluster_name`
   - `peer_vserver`

   Para actualizar las definiciones, elimine el **'{}'** después de `req_details` la `svm_initial` definición y agregue la definición correcta.

3. Cree un archivo en `logic-tasks` la carpeta para la solicitud de servicio. Por ejemplo, cree un archivo `svm_initial.yml` llamado .

   Copie las siguientes líneas en el archivo:

```
- 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:
```

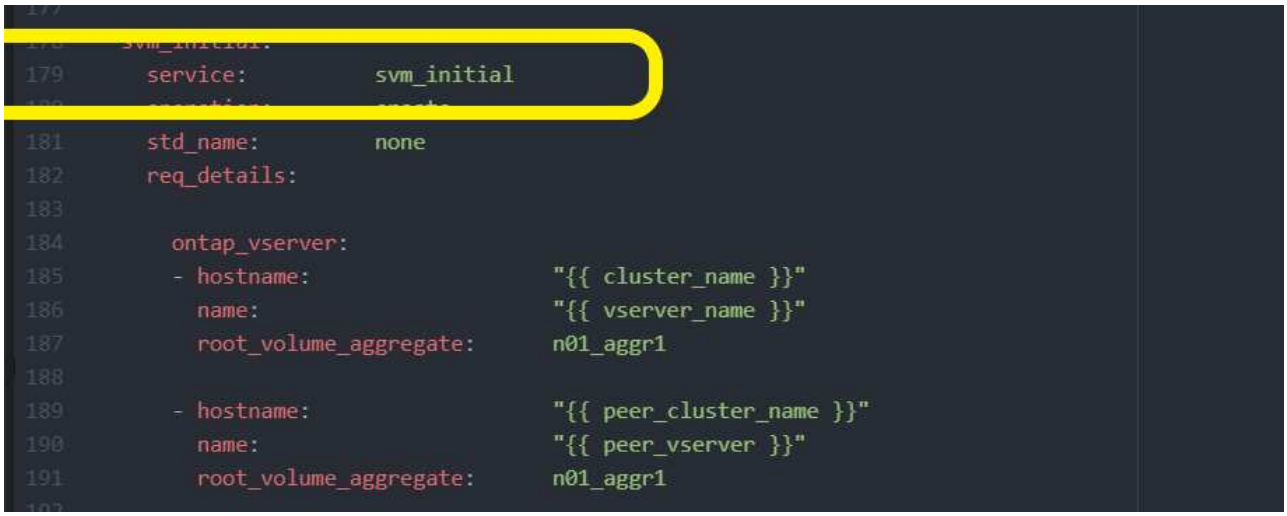
4. Defina la `raw_service_request` variable.

   Puede utilizar una de las siguientes opciones para definir la `raw_service_request` variable en `svm_initial` la `logic-tasks` carpeta:

   - **Opción 1**: Defina manualmente la `raw_service_request` variable.

     Abra el `tutorial-requests.yml` archivo con un editor y copie el contenido de la línea 179 a la

línea 222. Pegue el contenido bajo `raw service request` la variable en el nuevo `svm_initial.yml` archivo, como se muestra en los siguientes ejemplos:



Muestra el ejemplo

Archivo de ejemplo `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
```

- **Opción 2**: Usa una plantilla Jinja para definir la solicitud:

  También puede utilizar el siguiente formato de plantilla Jinja para obtener el `raw_service_request` valor.

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

5. Ejecute la solicitud:

```
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. Inicie sesión en cada instancia de ONTAP y valide la configuración.
7. Añada las interfaces de SVM.

   Defina el `ontap_interface` servicio en `svm_initial` el `services.yml` archivo y vuelva a ejecutar la solicitud:

```
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. Inicie sesión en cada instancia de ONTAP y compruebe que las interfaces de SVM se hayan configurado.

Paso 5: Opcionalmente, defina una solicitud de servicio de forma dinámica

En los pasos anteriores, `raw_service_request` la variable está codificada de forma fija. Esto es útil para el aprendizaje, desarrollo y pruebas. También puede generar dinámicamente una solicitud de servicio.

La siguiente sección proporciona una opción para producir dinámicamente el requerido `raw_service_request` si no desea integrarlo con sistemas de nivel superior.

- Si la `logic_operation` variable no está definida en el comando, el `logic.yml` archivo no importa ningún archivo de la `logic-tasks` carpeta. Esto significa que `raw_service_request` debe definirse fuera de Ansible y proporcionarse al marco en la ejecución.
- Un nombre de archivo de tarea en la `logic-tasks` carpeta debe coincidir con el valor de `logic_operation` la variable sin la extensión .yml.
- Los archivos de tareas de la `logic-tasks` carpeta definen dinámicamente a. `raw_service_request` El único requisito es que se defina una tarea válida `raw_service_request` como la última tarea del archivo correspondiente.

Cómo definir dinámicamente una solicitud de servicio

Hay varias formas de aplicar una tarea lógica para definir dinámicamente una solicitud de servicio. Algunas de estas opciones se enumeran a continuación:

- Uso de un archivo de tareas de Ansible desde la `logic-tasks` carpeta
- Llamada a un rol personalizado que devuelve datos adecuados para la conversión a un `raw_service_request` variable.
- Invocar otra herramienta fuera del entorno de Ansible para proporcionar los datos necesarios. Por ejemplo, una llamada API DE REST a Active IQ Unified Manager.

Los siguientes comandos de ejemplo definen dinámicamente una solicitud de servicio para cada cluster utilizando el `tutorial-requests.yml` archivo:

```
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
```

Paso 6: Implemente la solución ONTAP day 0/1

En esta etapa, debería haber completado lo siguiente:

- Revisó y modificó todos los archivos de `playbooks/inventory/group_vars/all` acuerdo con sus requisitos. Hay comentarios detallados en cada archivo para ayudarle a realizar los cambios.
- Se han agregado los archivos de tareas necesarios al `logic-tasks` directorio.
- Se han agregado los archivos de datos necesarios al `playbook/vars` directorio.

Utilice los siguientes comandos para poner en marcha la solución ONTAP day 0/1 y comprobar el estado de su implementación:

> En esta etapa, ya debería haber descifrado y modificado el `vault.yml` archivo y debe estar cifrado con su nueva contraseña.

- Ejecute el servicio ONTAP day 0:

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

- Ejecute el servicio ONTAP day 1:

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

- Aplicar configuración en todo el clúster:

```
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>
```

- Ejecute comprobaciones de estado:

```
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>
```

Personalizar la solución ONTAP day 0/1

Para personalizar la solución de ONTAP day 0/1 según sus requisitos, puede agregar o cambiar roles de Ansible.

Los roles representan los microservicios dentro del marco de Ansible. Cada microservicio realiza una operación. Por ejemplo, el día 0 de ONTAP es un servicio que contiene varios microservicios.

Añada roles de Ansible

Puede añadir roles de Ansible para personalizar la solución para su entorno. Los roles requeridos se definen por definiciones de servicio dentro del marco de Ansible.

Un rol debe cumplir los siguientes requisitos para ser utilizado como microservicio:

- Acepte una lista de argumentos en la `args` variable.
- Utilice la estructura «bloque, rescate, siempre» de Ansible con ciertos requisitos para cada bloque.
- Utilice un único módulo de Ansible y defina una única tarea dentro del bloque.
- Implemente todos los parámetros de módulo disponibles de acuerdo con los requisitos detallados en esta sección.

Estructura de microservicios requerida

Cada rol debe admitir las siguientes variables:

- `mode`: Si el modo se establece en `test` el rol intenta importar el `test.yml` que muestra lo que hace el rol sin ejecutarlo realmente.

  No siempre es posible implementar esto debido a ciertas interdependencias.

- `status`: El estado general de la ejecución de playbook. Si el valor no está definido en `success` el rol no se ejecuta.
- `args` : Una lista de diccionarios específicos de roles con claves que coinciden con los nombres de parámetros de rol.
- `global_log_messages`: Recopla mensajes de registro durante la ejecución de playbook. Se genera una entrada cada vez que se ejecuta el rol.
- `log_name`: El nombre utilizado para hacer referencia al rol dentro de las `global_log_messages` entradas.
- `task_descr`: Una breve descripción de lo que hace el rol.
- `service_start_time`: La marca de tiempo utilizada para rastrear la hora en que se ejecuta cada rol.
- `playbook_status`: El estado del libro de estrategia de Ansible.
- `role_result`: La variable que contiene la salida de rol y se incluye en cada mensaje dentro de las `global_log_messages` entradas.

Ejemplo de estructura de roles

El siguiente ejemplo proporciona la estructura básica de un rol que implementa un microservicio. Debe cambiar las variables de este ejemplo para su configuración.

Muestra el ejemplo

Estructura de rol básica:

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

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

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

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

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

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

```
            <MODULE_SPECIFIC_PARAMETERS>

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

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

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

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

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

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

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

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

Variables utilizadas en el rol de ejemplo:

- `<NAME>`: Un valor reemplazable que se debe proporcionar para cada microservicio.
- `<LOG_NAME>`: El nombre de forma corta del rol utilizado para fines de registro. Por ejemplo, `ONTAP_VOLUME`.
- `<TASK_DESCRIPTION>`: Una breve descripción de lo que hace el microservicio.
- `<MODULE_NAME>`: El nombre del módulo de Ansible para la tarea.

> El libro de estrategia de nivel superior `execute.yml` especifica `netapp.ontap` la colección. Si el módulo forma parte de la `netapp.ontap` colección, no es necesario especificar completamente el nombre del módulo.

- `<MODULE_SPECIFIC_PARAMETERS>`: Parámetros del módulo Ansible específicos del módulo utilizado para implementar el microservicio. En la siguiente lista se describen los tipos de parámetros y cómo se deben agrupar.
  - Parámetros necesarios: Se especifican todos los parámetros necesarios sin valor por defecto.
  - Parámetros que tienen un valor predeterminado específico del microservicio (no el mismo que un valor predeterminado especificado por la documentación del módulo).
  - Todos los parámetros restantes se utilizan `default(omit)` como valor predeterminado.

Uso de diccionarios de varios niveles como parámetros de módulo

Algunos módulos Ansible que proporcionan NetApp utilizan diccionarios de varios niveles para los parámetros de los módulos (por ejemplo, grupos de políticas de calidad de servicio fijos y adaptativos).

El uso `default(omit)` solo no funciona cuando se utilizan estos diccionarios, especialmente cuando hay más de uno y son mutuamente excluyentes.

Si necesita utilizar diccionarios de varios niveles como parámetros de módulo, debe dividir la funcionalidad en múltiples microservicios (roles) para que cada uno tenga garantizado que proporcione al menos un valor de diccionario de segundo nivel para el diccionario relevante.

Los siguientes ejemplos muestran grupos de políticas de calidad de servicio fijos y adaptativos divididos en dos microservicios.

El primer microservicio contiene valores de grupo de políticas de QoS fijos:

```
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)
}}"
```

El segundo microservicio contiene los valores del grupo de políticas de QoS adaptativo:

```
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) }}"
```

Información de copyright

Información de la marca comercial