NetApp自動化

NetApp Automation

NetApp

October 23, 2024

目錄

NetApp自動化

NetApp 自動化的新功能

NetApp 會定期更新自動化解決方案、產品 REST API 及相關軟體、為您提供新功能、增強功能及錯誤修正。

2023 年 7 月 27 日

Cloud Volumes ONTAP

Cloud Volumes ONTAP 的支援是由公有雲環境所組織。針對下列雲端環境提供新的解決方案：

- "Cloud Volumes ONTAP for Google Cloud ：連拍到雲端"

2023 年 6 月 04 日

NetApp "BlueXP 自動化目錄" 可透過 BlueXP 網路使用者介面取得。自動化目錄可讓您存取解決方案、協助您自動化部署及整合 NetApp 產品。這些解決方案的說明文件分為幾個不同的產品或功能領域、如下所述。

Amazon FSX for NetApp ONTAP 產品

以下提供兩種 Amazon FSX for NetApp ONTAP 解決方案：

- "Amazon FSX for NetApp ONTAP - 點到雲端"
- "Amazon FSX for NetApp ONTAP - 災難恢復"

Azure NetApp Files

我們提供的解決方案可協助您使用 Azure NetApp Files 部署 Oracle ：

- "使用 Azure NetApp Files 的 Oracle"

Cloud Volumes ONTAP

Cloud Volumes ONTAP 的支援是由公有雲環境所組織。針對兩種雲端環境提供的初始解決方案如下：

- "Cloud Volumes ONTAP AWS - 點到雲端"
- "Cloud Volumes ONTAP Azure ：連拍到雲端"

BlueXP 自動化目錄

BlueXP 自動化目錄總覽

BlueXP 自動化目錄是 NetApp 客戶、合作夥伴和員工可用的自動化解決方案集合。目錄有幾項功能和優點。

滿足您自動化需求的單一位置

您可以透過 BlueXP Web 使用者介面存取 "BlueXP 自動化目錄"。這可為指令碼、教戰手冊和模組提供單一位置、以強化 NetApp 產品和服務的自動化與操作。

解決方案是由 **NetApp** 所建立及測試

所有的自動化解決方案和指令碼都是由 NetApp 所建立和測試的。每個解決方案都針對特定的客戶使用案例或要求。最著重於與 NetApp 檔案和資料服務的整合。

文件

每個自動化解決方案都包含相關文件、可協助您開始使用。當您透過 BlueXP 網頁介面存取解決方案時、您可以在本網站取得所有文件。文件是根據 NetApp 產品和雲端服務來組織。

為未來奠定堅實基礎

NetApp 致力於協助客戶改善及簡化資料中心和雲端環境的自動化。我們預期會繼續加強 BlueXP 自動化目錄、以因應客戶需求、技術變更及持續的產品整合。

我們想聽聽您的意見

NetApp 客戶體驗辦公室（ CXO ）自動化團隊希望能與您聯絡。如果您有任何意見反應、問題或功能要求、請傳送電子郵件至 mailto ： ng-cxO-Automation 管理員 NetApp · com[CXO 自動化團隊]。

Amazon FSX for NetApp ONTAP 產品

Amazon FSX for NetApp ONTAP - 點到雲端

您可以使用此自動化解決方案來為 NetApp ONTAP 提供 Amazon FSX 、其中包含 Volume 和相關的 FlexCache 。

Amazon FSX for NetApp ONTAP 也稱為「 * 適用於 ONTAP * 的 FSX 」。

關於本解決方案

本解決方案所提供的自動化程式碼可在較高層級執行下列動作：

- 為 ONTAP 檔案系統配置目的地 FSX
- 為檔案系統佈建儲存虛擬機器（ SVM ）
- 在來源和目的地系統之間建立叢集對等關係
- 在 FlexCache 的來源系統和目的地系統之間建立 SVM 對等關係
- 您也可以選擇使用適用於 ONTAP 的 FSX 建立 FlexVol Volume

- 在適用於 ONTAP 的 FSX 中建立 FlexCache Volume 、來源指向內部儲存設備

自動化是以 Docker 和 Docker Compose 為基礎、必須安裝在 Linux 虛擬機器上、如下所述。

開始之前

您必須具備下列條件才能完成資源配置與組態：

- 您需要透過 BlueXP Web UI 下載 "Amazon FSX for NetApp ONTAP - 點到雲端"自動化解決方案。解決方案打包爲 file `AWS_FSxN_BTC.zip`。
- 來源與目的地系統之間的網路連線。
- 具有下列特性的 Linux VM ：
  - 以 Debian 為基礎的 Linux 套裝作業系統
  - 部署在用於 FSX 以進行 ONTAP 資源配置的同一 VPC 子集上
- AWS 帳戶。

步驟 **1** ：安裝及設定 **Docker**

在以 Debian 為基礎的 Linux 虛擬機器中安裝及設定 Docker 。

步驟

1. 準備好環境。

```
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. 安裝 Docker 並驗證安裝。

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

3. 將所需的 Linux 群組與相關的使用者一起新增。

   請先檢查 Linux 系統中是否存在 * 泊塢視窗 * 群組。如果沒有、請建立群組並新增使用者。根據預設、目前的 Shell 使用者會新增至群組。

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

4. 啟動新的群組和使用者定義

   如果您與使用者一起建立新群組、則需要啟動定義。若要這麼做、您可以先登出 Linux 、然後再重新登入。或者、您也可以執行下列命令。

```
newgrp docker
```

步驟 2 ：安裝 Docker Compose

在以 Debian 為基礎的 Linux 虛擬機器中安裝 Docker Compose 。

步驟

1. 安裝 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. 確認安裝成功。

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

步驟 3 ：準備 Docker 映像檔

您需要擷取並載入自動化解決方案隨附的 Docker 映像。

步驟

1. 將解決方案檔案複製 `AWS_FSxN_BTC.zip`到執行自動化程式碼的虛擬機器。

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

   輸入參數 `private-key.pem`是用於 AWS 虛擬機器驗證（ EC2 執行個體）的私密金鑰檔案。

2. 使用解決方案檔案瀏覽至正確的資料夾、然後解壓縮檔案。

```
unzip AWS_FSxN_BTC.zip
```

3. 瀏覽至以解壓縮作業建立的新資料夾 `AWS_FSxN_BTC`、並列出檔案。您應該會看到 file `aws_fsxn_flexcache_image_latest.tar.gz`。

```
ls -la
```

4. 載入 Docker 映像檔。負載作業通常應在數秒內完成。

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

5. 確認已載入 Docker 映像。

```
docker images
```

您應該會看到帶有標記的 `latest` Docker 映像 `aws_fsxn_flexcache_image`。

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

步驟 **4** ：建立 **AWS** 認證的環境檔案

您必須使用存取和秘密金鑰來建立驗證的本機變數檔案。然後將檔案新增至 `.env`檔案。

步驟

1. 在下列位置建立 `awsauth.env`檔案：

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

2. 將下列內容新增至檔案：

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

格式 *must * 與上述所示完全相同，且與 `value`之間沒有任何空格 `key`。

3. 使用變數將絕對檔案路徑新增至 `.env`檔案 `AWS_CREDS`。例如：

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

步驟 **5** ：建立外部磁碟區

您需要外部磁碟區、以確保 Terraform 狀態檔案和其他重要檔案持續存在。這些檔案必須可供 Terraform 執行工作流程和部署。

步驟

1. 在 Docker Compose 之外建立外部 Volume 。

執行命令之前、請務必將 Volume 名稱（最後參數）更新為適當的值。

```
docker volume create aws_fsxn_volume
```

2. 使用命令將外部磁碟區的路徑新增至 `.env`環境檔案：

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

   請記得保留現有的檔案內容和結腸格式。例如：

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

   您可以改用下列命令、將 NFS 共用新增為外部磁碟區：

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

3. 更新 Terraform 變數。
   a. 瀏覽至資料夾 `aws_fsxn_variables`。
   b. 確認存在以下兩個檔案： `terraform.tfvars`和 `variables.tf`。
   c. 視環境需求更新中的值 `terraform.tfvars`。

      如需詳細資訊、請參閱 "Terraform 資源： ONTAP 檔案系統" 。

步驟 6 ：為 NetApp ONTAP 和 FlexCache 配置 Amazon FSX

您可以為 NetApp ONTAP 和 FlexCache 配置 Amazon FSX 。

步驟

1. 瀏覽至資料夾根目錄（ AWS_FSXN_BTC ）、然後發出資源配置命令。

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

   此命令會建立兩個容器。第一個容器會部署適用於 ONTAP 的 FSX 、第二個容器則會建立叢集對等關係、 SVM 對等關係、目的地 Volume 和 FlexCache 。

2. 監控資源配置程序。

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

   此命令可即時提供輸出，但已設定為透過檔案擷取記錄 `deployment.log`。您可以通過編輯文件和更新變量 `DEPLOYMENT_LOGS`來更改這些日誌文件的名稱 `.env`。

步驟 7 ：銷毀 **Amazon FSX for NetApp ONTAP** 和 **FlexCache**

您可以選擇性地刪除和移除 Amazon FSX for NetApp ONTAP 和 FlexCache 。

1. 將檔案中的 `terraform.tfvars`變數設 `flexcache_operation`為「銷毀」。
2. 瀏覽至資料夾根目錄（ AWS_FSXN_BTC ）、然後發出下列命令。

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

此命令會建立兩個容器。第一個容器會刪除 FlexCache 、第二個容器則會刪除 ONTAP 的 FSX 。

3. 監控資源配置程序。

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

Amazon FSX for NetApp ONTAP - 災難恢復

您可以使用此自動化解決方案、使用 Amazon FSX for NetApp ONTAP 為來源系統進行災難恢復備份。

Amazon FSX for NetApp ONTAP 也稱為「 * 適用於 ONTAP * 的 FSX 」。

關於本解決方案

本解決方案所提供的自動化程式碼可在較高層級執行下列動作：

- 為 ONTAP 檔案系統配置目的地 FSX
- 為檔案系統佈建儲存虛擬機器（ SVM ）
- 在來源和目的地系統之間建立叢集對等關係
- 在 SnapMirror 的來源系統和目的地系統之間建立 SVM 對等關係
- 建立目的地Volume
- 在來源磁碟區和目的地磁碟區之間建立 SnapMirror 關係
- 在來源磁碟區和目的地磁碟區之間啟動 SnapMirror 傳輸

自動化是以 Docker 和 Docker Compose 為基礎、必須安裝在 Linux 虛擬機器上、如下所述。

開始之前

您必須具備下列條件才能完成資源配置與組態：

- 您需要透過 BlueXP Web UI 下載 "Amazon FSX for NetApp ONTAP - 災難恢復"自動化解決方案。解決方案包裝為 `FSxN_DR.zip`。此 zip 包含 `AWS_FSxN_Bck_Prov.zip`您將用來部署本文件所述解決方案的檔案。
- 來源與目的地系統之間的網路連線。
- 具有下列特性的 Linux VM ：

- 以 Debian 為基礎的 Linux 套裝作業系統
- 部署在用於 FSX 以進行 ONTAP 資源配置的同一 VPC 子集上

- AWS 帳戶。

步驟 **1** ：安裝及設定 **Docker**

在以 Debian 為基礎的 Linux 虛擬機器中安裝及設定 Docker 。

步驟

1. 準備好環境。

```
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. 安裝 Docker 並驗證安裝。

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

3. 將所需的 Linux 群組與相關的使用者一起新增。

   請先檢查 Linux 系統中是否存在 * 泊塢視窗 * 群組。如果不存在、請建立群組並新增使用者。根據預設、目前的 Shell 使用者會新增至群組。

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

4. 啟動新的群組和使用者定義

   如果您與使用者一起建立新群組、則需要啟動定義。若要這麼做、您可以先登出 Linux 、然後再重新登入。或者、您也可以執行下列命令。

```
newgrp docker
```

步驟 2 ：安裝 Docker Compose

在以 Debian 為基礎的 Linux 虛擬機器中安裝 Docker Compose 。

步驟

1. 安裝 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. 確認安裝成功。

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

步驟 3 ：準備 Docker 映像檔

您需要擷取並載入自動化解決方案隨附的 Docker 映像。

步驟

1. 將解決方案檔案複製 `AWS_FSxN_Bck_Prov.zip`到執行自動化程式碼的虛擬機器。

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

   輸入參數 `private-key.pem`是用於 AWS 虛擬機器驗證（ EC2 執行個體）的私密金鑰檔案。

2. 使用解決方案檔案瀏覽至正確的資料夾、然後解壓縮檔案。

```
unzip AWS_FSxN_Bck_Prov.zip
```

3. 瀏覽至以解壓縮作業建立的新資料夾 `AWS_FSxN_Bck_Prov`、並列出檔案。您應該會看到 file `aws_fsxn_bck_image_latest.tar.gz`。

```
ls -la
```

4. 載入 Docker 映像檔。負載作業通常應在數秒內完成。

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

5. 確認已載入 Docker 映像。

```
docker images
```

您應該會看到帶有標記的 `latest` Docker 映像 `aws_fsxn_bck_image`。

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

步驟 **4** ：建立 **AWS** 認證的環境檔案

您必須使用存取和秘密金鑰來建立驗證的本機變數檔案。然後將檔案新增至 `.env`檔案。

步驟

1. 在下列位置建立 `awsauth.env`檔案：

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

2. 將下列內容新增至檔案：

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

格式 *must * 與上述所示完全相同，且與 `value`之間沒有任何空格 `key`。

3. 使用變數將絕對檔案路徑新增至 `.env`檔案 `AWS_CREDS`。例如：

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

步驟 **5** ：建立外部磁碟區

您需要外部磁碟區、以確保 Terraform 狀態檔案和其他重要檔案持續存在。這些檔案必須可供 Terraform 執行工作流程和部署。

步驟

1. 在 Docker Compose 之外建立外部 Volume 。

執行命令之前、請務必將 Volume 名稱（最後參數）更新為適當的值。

```
docker volume create aws_fsxn_volume
```

2. 使用命令將外部磁碟區的路徑新增至 `.env`環境檔案：

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

請記得保留現有的檔案內容和結腸格式。例如：

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

您可以改用下列命令、將 NFS 共用新增為外部磁碟區：

PERSISTENT_VOL=nfs/mnt/document:/aws_fsx_bck

3. 更新 Terraform 變數。
   a. 瀏覽至資料夾 aws_fsxn_variables。
   b. 確認存在以下兩個檔案： terraform.tfvars`和 `variables.tf。
   c. 視環境需求更新中的值 terraform.tfvars。

      如需詳細資訊、請參閱 "Terraform 資源： ONTAP 檔案系統" 。

步驟 **6** ：部署備份解決方案

您可以部署和配置災難恢復備份解決方案。

步驟

1. 瀏覽至資料夾根目錄（ AWS_FSxN_Bck_Prov ）、然後發出資源配置命令。

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

此命令會建立三個容器。第一個容器會部署適用於 ONTAP 的 FSX 。第二個容器會建立叢集對等關係、 SVM 對等關係和目的地 Volume 。第三個容器會建立 SnapMirror 關係、並起始 SnapMirror 傳輸。

2. 監控資源配置程序。

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

此命令可即時提供輸出，但已設定為透過檔案擷取記錄 deployment.log。您可以通過編輯文件和更新變量 DEPLOYMENT_LOGS`來更改這些日誌文件的名稱 `.env。

Azure NetApp Files

使用 **Azure NetApp Files** 安裝 **Oracle**

您可以使用此自動化解決方案來配置 Azure NetApp Files Volume 、並在可用的虛擬機器上安裝 Oracle 。然後 Oracle 將這些磁碟區用於資料儲存。

關於本解決方案

本解決方案所提供的自動化程式碼可在較高層級執行下列動作：

- 在 Azure 上設定 NetApp 帳戶
- 在 Azure 上設定儲存容量集區
- 根據定義配置 Azure NetApp Files 磁碟區
- 建立掛載點
- 將 Azure NetApp Files 磁碟區掛載至掛載點
- 在 Linux 伺服器上安裝 Oracle
- 建立接聽程式和資料庫
- 建立易插拔資料庫（ PDB ）
- 啓動偵聽器和 Oracle 實例
- 安裝並設定 `azacsnap`公用程式以拍攝快照

開始之前

您必須具備下列條件才能完成安裝：

- 您需要透過 BlueXP Web UI 下載 "使用 Azure NetApp Files 的 Oracle"自動化解決方案。解決方案打包爲 file `na_oracle19c_deploy-master.zip`。
- 具有下列特性的 Linux VM ：
  - RHEL 8 （ Standard_D8s_v3-RHEL-8 ）
  - 部署在用於 Azure NetApp Files 資源配置的同一個 Azure 虛擬網路上
- Azure 帳戶

自動化解決方案以映像形式提供、並使用 Docker 和 Docker Compose 執行。您需要如下所述、在 Linux 虛擬機器上安裝這兩個項目。

您也應該使用命令向 RedHat 註冊 VM `sudo subscription-manager register` 。命令會提示您輸入帳戶認證。如有需要、您可以在 https://developers.redhat.com/. 建立帳戶

步驟 **1** ：安裝及設定 **Docker**

在 RHEL 8 Linux 虛擬機器中安裝及設定 Docker 。

步驟

1. 使用下列命令安裝 Docker 軟體。

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

2. 啟動 Docker 並顯示版本以確認安裝成功。

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

3. 將所需的 Linux 群組與相關的使用者一起新增。

   請先檢查 Linux 系統中是否存在 * 泊塢視窗 * 群組。如果沒有、請建立群組並新增使用者。根據預設、目前的 Shell 使用者會新增至群組。

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

4. 啟動新的群組和使用者定義

   如果您與使用者一起建立新群組、則需要啟動定義。若要這麼做、您可以先登出 Linux 、然後再重新登入。或者、您也可以執行下列命令。

```
newgrp docker
```

步驟 **2** ：安裝 **Docker Compose** 和 **NFS** 公用程式

安裝及設定 Docker Compose 及 NFS 公用程式套件。

步驟

1. 安裝 Docker Compose 並顯示版本、以確認安裝成功。

```
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. 安裝 NFS 公用程式套件。

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

步驟 **3** ：下載 **Oracle** 安裝檔案

下載所需的 Oracle 安裝和修補程式檔案 `azacsnap`、以及公用程式。

步驟

1. 視需要登入您的 Oracle 帳戶。
2. 下載下列檔案。

檔案	說明
LINUX.X64_193000_db_home.zip	19.3 基礎安裝程式
p31281355_190000_Linux-x86-64.zip	19.8 RU 修補程式
p6880880_190000_Linux-x86-64.zip	opatch 版本 12.2 · 2 · 1 · 23
azacsnap_installer_v5.0.run	azacSnap 安裝程式

3. 將所有安裝文件放在文件夾中 `/tmp/archive`。
4. 確保資料庫伺服器上的所有使用者都能完整存取資料夾（讀取、寫入、執行） `/tmp/archive`。

步驟 **4** ：準備 **Docker** 映像檔

您需要擷取並載入自動化解決方案隨附的 Docker 映像。

步驟

1. 將解決方案檔案複製 `na_oracle19c_deploy-master.zip`到執行自動化程式碼的虛擬機器。

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

輸入參數 `private-key.pem`是用於 Azure 虛擬機器驗證的私密金鑰檔案。

2. 使用解決方案檔案瀏覽至正確的資料夾、然後解壓縮檔案。

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

3. 瀏覽至以解壓縮作業建立的新資料夾 `na_oracle19c_deploy-master`、並列出檔案。您應該會看到 file `ora_anf_bck_image.tar`。

```
ls -lt
```

4. 載入 Docker 映像檔。負載作業通常應在數秒內完成。

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

5. 確認已載入 Docker 映像。

```
docker images
```

您應該會看到帶有標記的 `latest` Docker 映像 `ora_anf_bck_image`。

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

步驟 5 ：建立外部磁碟區

您需要外部磁碟區、以確保 Terraform 狀態檔案和其他重要檔案持續存在。這些檔案必須可供 Terraform 執行工作流程和部署。

步驟

1. 在 Docker Compose 之外建立外部 Volume 。

   請務必先更新磁碟區名稱、再執行命令。

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

2. 使用命令將外部磁碟區的路徑新增至 `.env`環境檔案：

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

   請記得保留現有的檔案內容和結腸格式。例如：

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

3. 更新 Terraform 變數。
   a. 瀏覽至資料夾 `ora_anf_variables`。
   b. 確認存在以下兩個檔案： `terraform.tfvars`和 `variables.tf`。
   c. 視環境需求更新中的值 `terraform.tfvars`。

步驟 6 ：安裝 Oracle

您現在可以配置和安裝 Oracle 。

步驟

1. 使用下列命令順序安裝 Oracle 。

   ```
   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. 重新載入 Bash 變數、並顯示的值以確認 `ORACLE_HOME`。
   a. `cd /home/oracle`
   b. `source .bash_profile`
   c. `echo $ORACLE_HOME`
3. 您應該可以登入 Oracle 。

```
sudo su oracle
```

步驟 **7** ：驗證 **Oracle** 安裝

您應該確認 Oracle 安裝成功。

步驟

1. 登入 Linux Oracle 伺服器、並顯示 Oracle 程序清單。這會確認安裝已如預期完成、且 Oracle 資料庫正在執行中。

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

2. 登入資料庫以檢查資料庫組態、並確認已正確建立 PDB 。

```
sqlplus / as sysdba
```

   您應該會看到類似下列的輸出：

```
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. 執行幾個簡單的 SQL 命令、確認資料庫可用。

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

步驟 8 ：安裝 **azacsnap** 公用程式並執行快照備份

您必須安裝並執行 `azacsnap`公用程式、才能執行快照備份。

步驟

1. 安裝容器。

```
docker-compose up azacsnap_install
```

2. 切換至 Snapshot 使用者帳戶。

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

3. 設定儲存備份詳細資料檔案。這將會建立 `azacsnap.json`組態檔案。

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

4. 執行快照備份。

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

步驟 9 ：選擇性地將內部部署的 **PDB** 移轉至雲端

您可以選擇性地將內部部署的 PDB 移轉至雲端。

步驟

1. 視環境需求設定檔案中的變數 `tfvars`。
2. 移轉 PDB 。

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

AWS 適用的 Cloud Volumes ONTAP

Cloud Volumes ONTAP for AWS - 點到雲端

本文支援 NetApp Cloud Volumes ONTAP for AWS 自動化解決方案、 NetApp 客戶可從 BlueXP 自動化目錄取得此解決方案。

Cloud Volumes ONTAP for AWS 自動化解決方案使用 Terraform 將 Cloud Volumes ONTAP for AWS 的容器化部署作業自動化、讓您能夠快速部署 Cloud Volumes ONTAP for AWS 、無需任何手動介入。

開始之前

- 您必須透過 BlueXP Web UI 下載"Cloud Volumes ONTAP AWS - 點到雲端"自動化解決方案。解決方案包裝為 `cvo_aws_flexcache.zip`。
- 您必須在與 Cloud Volumes ONTAP 相同的網路上安裝 Linux VM 。
- 安裝 Linux VM 之後、您必須遵循本解決方案中的步驟來安裝必要的相依性。

步驟 1 ：安裝 Docker 和 Docker Compose

安裝Docker

以下步驟以 Ubuntu 20.04 Debian Linux 發佈軟體為例。您執行的命令取決於您所使用的 Linux 發佈軟體。請參閱特定的 Linux 發佈軟體文件以瞭解您的組態。

步驟

1. 執行下列命令來安裝 Docker `sudo` ：

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

2. 驗證安裝：

```
docker –version
```

3. 確認您的 Linux 系統上已建立名為「泊塢視窗」的群組。如有必要、請建立群組：

```
sudo groupadd docker
```

4. 將需要存取 Docker 的使用者新增至群組：

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

5. 您的變更會在登出並重新登入終端機後套用。或者、您也可以立即套用變更：

```
newgrp docker
```

安裝 Docker Compose

步驟

1. 執行下列命令來安裝 Docker Compose `sudo` ：

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

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

2. 驗證安裝：

```
docker-compose –version
```

步驟 2 ：準備 Docker 映像檔

步驟

1. 將資料夾複製 `cvo_aws_flexcache.zip`到您要用來部署 Cloud Volumes ONTAP 的 Linux VM ：

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

   - `private-key.pem`是您的私密金鑰檔案、無需密碼即可登入。
   - `awsuser`是 VM 使用者名稱。
   - `IP_ADDRESS_OF_VM`是 VM IP 位址。
   - `LOCATION_TO_BE_COPIED`為資料夾的複製位置。

2. 解壓縮 `cvo_aws_flexcache.zip`資料夾。您可以擷取目前目錄或自訂位置中的資料夾。

   若要擷取目前目錄中的資料夾、請執行：

```
unzip cvo_aws_flexcache.zip
```

   若要擷取自訂位置中的資料夾、請執行：

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

3. 擷取內容之後、請瀏覽至 `CVO_Aws_Deployment`資料夾並執行下列命令以檢視檔案：

```
 ls -la
```

您應該會看到類似下列範例的檔案清單：

```
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. 找到 `cvo_aws_flexcache_ubuntu_image.tar`檔案。其中包含部署 Cloud Volumes ONTAP for AWS 所需的 Docker 映像。
5. 解壓縮檔案：

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

6. 等待幾分鐘、讓 Docker 映像檔載入、然後驗證 Docker 映像檔是否成功載入：

```
docker images
```

您應該會看到一個以 `latest`標記命名的 Docker 映像 `cvo_aws_flexcache_ubuntu_image`、如下列範例所示：

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

> 您可以視需要變更 Docker 映像名稱。如果您變更 Docker 映像名稱、請務必更新和 `docker-compose-destroy`檔案中的 Docker 映像名稱 `docker-compose-deploy`。

步驟 **3** ：建立環境變數檔案

在此階段、您必須建立兩個環境變數檔案。其中一個檔案是使用 AWS 存取和秘密金鑰來驗證 AWS 資源管理員 API 。第二個檔案用於設定環境變數、讓 BlueXP Terraform 模組能夠找出並驗證 AWS API 。

步驟

1. 在下列位置建立 `awsauth.env`檔案：

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

   a. 將下列內容新增至 `awsauth.env`檔案：

      access_key=<> secret _key=<>

   格式 *must * 與上圖完全相同。

2. 將絕對檔案路徑新增至 `.env`檔案。

   輸入與環境變數對應的環境檔案 `AWS_CREDS`**絕對路徑** `awsauth.env`。

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

3. 瀏覽至 `cvo_aws_variable`資料夾、並更新認證檔案中的存取和秘密金鑰。

   將下列內容新增至檔案：

   AWS 存取金鑰識別碼 =<> AWS_secret 存取金鑰 =<>

   格式 *must * 與上圖完全相同。

步驟 **4** ：將 **Cloud Volumes ONTAP** 授權新增至 **BlueXP** 或訂閱 **BlueXP**

您可以將 Cloud Volumes ONTAP 授權新增至 BlueXP 、或在 AWS Marketplace 中訂閱 NetApp BlueXP 。

步驟

1. 從 AWS 入口網站瀏覽至 * SaaS * 、然後選取 * 訂閱 NetApp BlueXP * 。

   您可以使用與 Cloud Volumes ONTAP 相同的資源群組或不同的資源群組。

2. 設定 BlueXP 入口網站、將 SaaS 訂閱匯入 BlueXP 。

   您可以直接從 AWS 入口網站進行設定。

   系統會將您重新導向至 BlueXP 入口網站、以確認組態。

3. 選取 * 儲存 * 、確認 BlueXP 入口網站中的組態。

步驟 **5** ：建立外部磁碟區

您應該建立外部磁碟區、以保留 Terraform 狀態檔案及其他重要檔案。您必須確定 Terraform 可以使用這些檔案來執行工作流程和部署。

步驟

1. 在 Docker Compose 之外建立外部 Volume ：

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

範例：

```
docker volume create cvo_aws_volume_dst
```

2. 請使用下列其中一個選項：

   a. 新增外部磁碟區路徑至 `.env`環境檔案。

      您必須遵循如下所示的確切格式。

      格式：

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

      範例：
      `PERSISTENT_VOL=cvo_aws_volume_dst:/cvo_aws`

   b. 將 NFS 共用新增為外部磁碟區。

      請確定 Docker 容器可以與 NFS 共用通訊、而且已設定正確的權限、例如讀取 / 寫入。

      i. 將 NFS 共用路徑新增為 Docker Compose 檔案中外部 Volume 的路徑、如下所示：格式：

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

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

3. 瀏覽至 `cvo_aws_variables`資料夾。

   您應該會在資料夾中看到下列變數檔案：

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

4. 根據您的需求變更檔案內的值 `terraform.tfvars`。

   修改檔案中的任何變數值時、您必須閱讀特定的支援文件 `terraform.tfvars`。這些值會因地區、可用度區域和 Cloud Volumes ONTAP for AWS 支援的其他因素而異。這包括單一節點和高可用度（ HA ）配對的授權、磁碟大小和 VM 大小。

   Connector 和 Cloud Volumes ONTAP Terraform 模組的所有支援變數都已定義在檔案中 `variables.tf`。在新增至檔案之前、您必須先參考檔案 `terraform.tfvars`中的變數名稱 `variables.tf`。

5. 根據您的需求，您可以將下列選項設定為或，以啟用或 `false`停用 `FlexCache` 和 `FlexClone` `true`。

   下列範例可啟用 FlexCache 和 FlexClone ：

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

步驟 6 ：部署 Cloud Volumes ONTAP for AWS

請使用下列步驟來部署 Cloud Volumes ONTAP for AWS 。

步驟

1. 從根資料夾執行下列命令以觸發部署：

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

   觸發兩個容器、第一個容器會部署 Cloud Volumes ONTAP 、第二個容器則會將遙測資料傳送至 AutoSupport 。

   第二個容器會等待、直到第一個容器成功完成所有步驟為止。

2. 使用記錄檔監控部署程序的進度：

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

   此命令會即時提供輸出、並擷取下列記錄檔中的資料：
   `deployment.log`

   `telemetry_asup.log`

   您可以使用下列環境變數編輯檔案、以變更這些記錄檔的名稱 `.env`：

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   下列範例說明如何變更記錄檔名稱：

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

完成後

您可以使用下列步驟移除暫存環境、並清除部署程序期間建立的項目。

步驟

1. 如果您已部署 FlexCache 、請在變數檔案中設定下列選項 `terraform.tfvars`、這樣會清除 FlexCache 磁碟區、並移除先前建立的暫存環境。

```
flexcache_operation = "destroy"
```

> 可能的選項有 `deploy`和 `destroy`。

2. 如果您已部署 FlexClone 、請在變數檔案中設定下列選項 `terraform.tfvars`、這樣會清除 FlexClone 磁碟區、並移除先前建立的暫存環境。

```
flexclone_operation = "destroy"
```

> 可能的選項有 `deploy`和 `destroy`。

適用於 **Azure Cloud Volumes ONTAP**

Cloud Volumes ONTAP for Azure ：連拍到雲端

本文支援 NetApp Cloud Volumes ONTAP for Azure Automation 解決方案、 NetApp 客戶可從 BlueXP Automation 目錄取得此解決方案。

Cloud Volumes ONTAP for Azure Automation 解決方案使用 Terraform 將 Cloud Volumes ONTAP for Azure 的容器化部署自動化、讓您能夠快速部署 Cloud Volumes ONTAP for Azure 、無需任何手動介入。

開始之前

- 您必須透過 BlueXP Web UI 下載"Cloud Volumes ONTAP Azure ：連拍到雲端"自動化解決方案。解決方案包裝為 `CVO-Azure-Burst-To-Cloud.zip`。
- 您必須在與 Cloud Volumes ONTAP 相同的網路上安裝 Linux VM 。
- 安裝 Linux VM 之後、您必須遵循本解決方案中的步驟來安裝必要的相依性。

步驟 **1** ：安裝 **Docker** 和 **Docker Compose**

安裝**Docker**

以下步驟以 Ubuntu 20.04 Debian Linux 發佈軟體為例。您執行的命令取決於您所使用的 Linux 發佈軟體。請參閱特定的 Linux 發佈軟體文件以瞭解您的組態。

步驟

1. 執行下列命令來安裝 Docker `sudo` ：

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

2. 驗證安裝：

```
docker –version
```

3. 確認您的 Linux 系統上已建立名為「泊塢視窗」的群組。如有必要、請建立群組：

```
sudo groupadd docker
```

4. 將需要存取 Docker 的使用者新增至群組：

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

5. 您的變更會在登出並重新登入終端機後套用。或者、您也可以立即套用變更：

```
newgrp docker
```

安裝 **Docker Compose**

步驟

1. 執行下列命令來安裝 Docker Compose `sudo` ：

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

2. 驗證安裝：

```
docker-compose -version
```

步驟 2 ：準備 Docker 映像檔

步驟

1. 將資料夾複製 `CVO-Azure-Burst-To-Cloud.zip`到您要用來部署 Cloud Volumes ONTAP 的 Linux VM ：

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

   - `private-key.pem`是您的私密金鑰檔案、無需密碼即可登入。
   - `azureuser`是 VM 使用者名稱。
   - `IP_ADDRESS_OF_VM`是 VM IP 位址。
   - `LOCATION_TO_BE_COPIED`為資料夾的複製位置。

2. 解壓縮 `CVO-Azure-Burst-To-Cloud.zip`資料夾。您可以擷取目前目錄或自訂位置中的資料夾。

   若要擷取目前目錄中的資料夾、請執行：

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

   若要擷取自訂位置中的資料夾、請執行：

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

3. 擷取內容之後、請瀏覽至 `CVO_Azure_Deployment`資料夾並執行下列命令以檢視檔案：

```
 ls -la
```

   您應該會看到類似下列範例的檔案清單：

```
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. 找到 `cvo_azure_flexcache_ubuntu_image_latest.tar.gz`檔案。其中包含部署 Cloud Volumes ONTAP for Azure 所需的 Docker 映像。
5. 解壓縮檔案：

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

6. 等待幾分鐘、讓 Docker 映像檔載入、然後驗證 Docker 映像檔是否成功載入：

```
docker images
```

您應該會看到一個以 `latest`標記命名的 Docker 映像 `cvo_azure_flexcache_ubuntu_image_latest`、如下列範例所示：

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

步驟 **3** ：建立環境變數檔案

在此階段、您必須建立兩個環境變數檔案。其中一個檔案是使用服務主體認證來驗證 Azure Resource Manager API 。第二個檔案用於設定環境變數、讓 BlueXP Terraform 模組能夠找出並驗證 Azure API 。

步驟

1. 建立服務主體。

   在建立環境變數檔案之前，您必須遵循中的步驟來建立服務主體"建立可存取資源的 Azure Active Directory 應用程式和服務主體"。

2. 將 * 貢獻者 * 角色指派給新建立的服務主體。
3. 建立自訂角色。

a. 找到 `sp_role.json`檔案、並在列出的動作下檢查所需的權限。

b. 插入這些權限、並將自訂角色附加至新建立的服務主體。

4. 瀏覽至 * 憑證與機密 * 、然後選取 * 新用戶端機密 * 以建立用戶端機密。

   當您建立用戶端機密時、必須從 * 值 * 欄記錄詳細資料、因為您將無法再次看到此值。您也必須記錄下列資訊：

   - 用戶端 ID
   - 訂閱 ID
   - 租戶 ID

   您需要這些資訊來建立環境變數。您可以在「服務主要使用者介面」的 * 總覽 * 區段中找到用戶端 ID 和租戶 ID 資訊。

5. 建立環境檔案。

   a. 在下列位置建立 `azureauth.env`檔案：

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

      i. 將下列內容新增至檔案：

         ClientID=<> clientSecret = <> 訂閱 Id=<> TenantId=<>

      格式 *must * 與上述所示完全相同、且在金鑰與值之間沒有任何空格。

   b. 在下列位置建立 `credentials.env`檔案：

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

      i. 將下列內容新增至檔案：

         Azure 租戶 ID=<> Azure 用戶端機密 =<> Azure 用戶端 ID =<> Azure 訂閱 ID=<>

   格式 *must * 與上述所示完全相同、且在金鑰與值之間沒有任何空格。

6. 將絕對檔案路徑新增至 `.env`檔案。

   在對應環境變數的檔案 `AZURE_RM_CREDS`**中輸入環境檔案的** `.env`**絕對路徑** `azureauth.env`。

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

   在對應環境變數的檔案 `BLUEXP_TF_AZURE_CREDS`**中輸入環境檔案的** `.env`**絕對路徑** `credentials.env`。

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

步驟 4 ：將 Cloud Volumes ONTAP 授權新增至 BlueXP 或訂閱 BlueXP

您可以將 Cloud Volumes ONTAP 授權新增至 BlueXP 、或在 Azure Marketplace 中訂閱 NetApp BlueXP 。

步驟

1. 從 Azure 入口網站瀏覽至 * SaaS * 、然後選取 * 訂閱 NetApp BlueXP * 。
2. 選擇 * Cloud Manager （按小時上限 PYGO 、 WORM 和資料服務） * 計畫。

   您可以使用與 Cloud Volumes ONTAP 相同的資源群組或不同的資源群組。

3. 設定 BlueXP 入口網站、將 SaaS 訂閱匯入 BlueXP 。

   您可以直接從 Azure 入口網站設定此功能、方法是瀏覽 * 產品與方案詳細資料 * 、然後選取 * 立即設定帳戶 * 選項。

   然後您將被重新導向至 BlueXP 入口網站以確認組態。

4. 選取 * 儲存 * 、確認 BlueXP 入口網站中的組態。

步驟 **5** ：建立外部磁碟區

您應該建立外部磁碟區、以保留 Terraform 狀態檔案及其他重要檔案。您必須確定 Terraform 可以使用這些檔案來執行工作流程和部署。

步驟

1. 在 Docker Compose 之外建立外部 Volume ：

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

   範例：

   ```
   docker volume create cvo_azure_volume_dst
   ```

2. 請使用下列其中一個選項：

   a. 新增外部磁碟區路徑至 `.env`環境檔案。

      您必須遵循如下所示的確切格式。

      格式：

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

      範例：
      ```
      PERSISTENT_VOL=cvo_azure_volume_dst:/cvo_azure
      ```

   b. 將 NFS 共用新增為外部磁碟區。

      請確定 Docker 容器可以與 NFS 共用通訊、而且已設定正確的權限、例如讀取 / 寫入。

      i. 將 NFS 共用路徑新增為 Docker Compose 檔案中外部 Volume 的路徑、如下所示：格式：

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

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

3. 瀏覽至 `cvo_azure_variables`資料夾。

   您應該會在資料夾中看到下列變數檔案：

   `terraform.tfvars`

   `variables.tf`

4. 根據您的需求變更檔案內的值 `terraform.tfvars`。

   修改檔案中的任何變數值時、您必須閱讀特定的支援文件 `terraform.tfvars`。這些值會因地區、可用度區域和 Cloud Volumes ONTAP for Azure 支援的其他因素而異。這包括單一節點和高可用度（ HA ）配對的授權、磁碟大小和 VM 大小。

   Connector 和 Cloud Volumes ONTAP Terraform 模組的所有支援變數都已定義在檔案中 `variables.tf`。在新增至檔案之前、您必須先參考檔案 `terraform.tfvars`中的變數名稱 `variables.tf`。

5. 根據您的需求，您可以將下列選項設定為或，以啟用或 `false`停用 `FlexCache` 和 `FlexClone` `true`。

   下列範例可啟用 FlexCache 和 FlexClone ：

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

6. 如有必要、您可以從 Azure Active Directory 服務擷取 Terraform 變數的值 `az_service_principal_object_id`：
   a. 瀏覽至 * 企業應用程式 → 所有應用程式 * 、然後選取您先前建立的服務主體名稱。
   b. 複製物件 ID 並插入 Terraform 變數的值：

      `az_service_principal_object_id`

步驟 6 ：部署適用於 Azure 的 Cloud Volumes ONTAP

請依照下列步驟部署適用於 Azure 的 Cloud Volumes ONTAP 。

步驟

1. 從根資料夾執行下列命令以觸發部署：

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

   觸發兩個容器、第一個容器會部署 Cloud Volumes ONTAP 、第二個容器則會將遙測資料傳送至 AutoSupport 。

第二個容器會等待、直到第一個容器成功完成所有步驟為止。

2. 使用記錄檔監控部署程序的進度：

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

此命令會即時提供輸出、並擷取下列記錄檔中的資料：

`deployment.log`

`telemetry_asup.log`

您可以使用下列環境變數編輯檔案、以變更這些記錄檔的名稱 `.env`：

`DEPLOYMENT_LOGS`

`TELEMETRY_ASUP_LOGS`

下列範例說明如何變更記錄檔名稱：

`DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

`TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

完成後

您可以使用下列步驟移除暫存環境、並清除部署程序期間建立的項目。

步驟

1. 如果您部署了 FlexCache 、請在檔案中設定下列選項 `terraform.tfvars`、這樣會清除 FlexCache 磁碟區、並移除先前建立的暫存環境。

```
flexcache_operation = "destroy"
```

可能的選項有 `deploy`和 `destroy`。

2. 如果您部署了 FlexClone 、請在檔案中設定下列選項 `terraform.tfvars`、這樣會清除 FlexClone 磁碟區、並移除先前建立的暫存環境。

```
flexclone_operation = "destroy"
```

可能的選項有 `deploy`和 `destroy`。

適用於 Google Cloud Cloud Volumes ONTAP

Cloud Volumes ONTAP for Google Cloud ：連拍到雲端

本文支援 NetApp Cloud Volumes ONTAP for Google Cloud Automation 解決方案、

NetApp 客戶可從 BlueXP 自動化目錄中取得此解決方案。

Cloud Volumes ONTAP for Google Cloud Automation 解決方案可自動化 Cloud Volumes ONTAP for Google Cloud 的容器化部署、讓您快速部署 Cloud Volumes ONTAP for Google Cloud 、無需手動介入。

開始之前

- 您必須透過 BlueXP Web UI 下載"Cloud Volumes ONTAP for Google Cloud ：連拍到雲端"自動化解決方案。解決方案包裝為 `cvo_gcp_flexcache.zip`。
- 您必須在與 Cloud Volumes ONTAP 相同的網路上安裝 Linux VM 。
- 安裝 Linux VM 之後、您必須遵循本解決方案中的步驟來安裝必要的相依性。

步驟 1 ：安裝 Docker 和 Docker Compose

安裝Docker

以下步驟以 Ubuntu 20.04 Debian Linux 發佈軟體為例。您執行的命令取決於您所使用的 Linux 發佈軟體。請參閱特定的 Linux 發佈軟體文件以瞭解您的組態。

步驟

1. 執行下列命令來安裝 Docker ：

```
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. 驗證安裝：

```
docker –version
```

3. 確認您的 Linux 系統上已建立名為「泊塢視窗」的群組。如有必要、請建立群組：

```
sudo groupadd docker
```

4. 將需要存取 Docker 的使用者新增至群組：

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

5. 您的變更會在登出並重新登入終端機後套用。或者、您也可以立即套用變更：

```
newgrp docker
```

安裝 Docker Compose

步驟

1. 執行下列命令來安裝 Docker Compose `sudo` ：

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

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

2. 驗證安裝：

```
docker-compose –version
```

步驟 2 ：準備 Docker 映像檔

步驟

1. 將資料夾複製 `cvo_gcp_flexcache.zip`到您要用來部署 Cloud Volumes ONTAP 的 Linux VM ：

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

   - `private-key.pem`是您的私密金鑰檔案、無需密碼即可登入。
   - `gcpuser`是 VM 使用者名稱。
   - `IP_ADDRESS_OF_VM`是 VM IP 位址。
   - `LOCATION_TO_BE_COPIED`為資料夾的複製位置。

2. 解壓縮 `cvo_gcp_flexcache.zip`資料夾。您可以擷取目前目錄或自訂位置中的資料夾。

   若要擷取目前目錄中的資料夾、請執行：

```
unzip cvo_gcp_flexcache.zip
```

   若要擷取自訂位置中的資料夾、請執行：

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

3. 擷取內容之後、請執行下列命令以檢視檔案：

```
 ls -la
```

您應該會看到類似下列範例的檔案清單：

```
    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. 找到 `cvo_gcp_flexcache_ubuntu_image.tar`檔案。其中包含部署 Cloud Volumes ONTAP for Google Cloud 所需的 Docker 映像。
5. 解壓縮檔案：

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

6. 等待幾分鐘、讓 Docker 映像檔載入、然後驗證 Docker 映像檔是否成功載入：

```
docker images
```

您應該會看到一個以 `latest`標記命名的 Docker 映像 `cvo_gcp_flexcache_ubuntu_image`、如下列範例所示：

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

您可以視需要變更 Docker 映像名稱。如果您變更 Docker 映像名稱、請務必更新和 `docker-compose-destroy`檔案中的 `Docker 映像名稱 `docker-compose-deploy。

步驟 3 ：更新 JSON 檔案

在此階段、您必須使用服務帳戶金鑰來更新 `cxo-automation-gcp.json`檔案、以驗證 Google Cloud 供應商的身分。

1. 建立具有部署 Cloud Volumes ONTAP 和 BlueXP Connector 權限的服務帳戶。"深入瞭解如何建立服務帳戶。"
2. 下載帳戶的金鑰檔、並使用金鑰檔案資訊更新 `cxo-automation-gcp.json`檔案。 `cxo-automation-gcp.json`檔案位於資料夾中 `cvo_gcp_variables。

   範例

```
{
  "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"
}
```

   檔案格式必須與上述內容完全相同。

步驟 4 ：訂閱 BlueXP

您可以在 Google Cloud Marketplace 中訂閱 NetApp BlueXP 。

步驟

1. 瀏覽"Google Cloud 主控台"並選取 * 訂閱 NetApp BlueXP * 。
2. 設定 BlueXP 入口網站、將 SaaS 訂閱匯入 BlueXP 。

   您可以直接從 Google Cloud Platform 進行設定。系統會將您重新導向至 BlueXP 入口網站、以確認組態。

3. 選取 * 儲存 * 、確認 BlueXP 入口網站中的組態。

如需更多資訊、請參閱 "管理 BlueXP 的 Google Cloud 認證和訂閱"。

步驟 **5** ：啟用必要的 **Google Cloud API**

您必須在專案中啟用下列 Google Cloud API 、才能部署 Cloud Volumes ONTAP 和 Connector 。

- Cloud Deployment Manager V2 API
- 雲端記錄 API
- Cloud Resource Manager API
- 運算引擎 API
- 身分識別與存取管理（ IAM ） API

"深入瞭解如何啟用API"

步驟 **6** ：建立外部磁碟區

您應該建立外部磁碟區、使 Terraform 狀態檔案和其他重要檔案持續存在。您必須確定 Terraform 可以使用這些檔案來執行工作流程和部署。

步驟

1. 在 Docker Compose 之外建立外部 Volume ：

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

   範例：

   ```
   docker volume create cvo_gcp_volume_dst
   ```

2. 請使用下列其中一個選項：

   a. 新增外部磁碟區路徑至 `.env`環境檔案。

      您必須遵循如下所示的確切格式。

      格式：

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

      範例：
      `PERSISTENT_VOL=cvo_gcp_volume_dst:/cvo_gcp`

   b. 將 NFS 共用新增為外部磁碟區。

      請確定 Docker 容器可以與 NFS 共用通訊、而且已設定正確的權限、例如讀取 / 寫入。

      i. 將 NFS 共用路徑新增為 Docker Compose 檔案中外部 Volume 的路徑、如下所示：格式：

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

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

3. 瀏覽至 `cvo_gcp_variables`資料夾。

   您應該會在資料夾中看到下列檔案：

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

4. 根據您的需求變更檔案內的值 `terraform.tfvars`。

   修改檔案中的任何變數值時、您必須閱讀特定的支援文件 `terraform.tfvars`。這些值會因地區、可用度區域和 Cloud Volumes ONTAP for Google Cloud 支援的其他因素而異。這包括單一節點和高可用度（ HA ）配對的授權、磁碟大小和 VM 大小。

   Connector 和 Cloud Volumes ONTAP Terraform 模組的所有支援變數都已定義在檔案中 `variables.tf`。在新增至檔案之前、您必須先參考檔案 `terraform.tfvars`中的變數名稱 `variables.tf`。

5. 根據您的需求，您可以將下列選項設定為或，以啟用或 `false`停用 `FlexCache` 和 `FlexClone` `true`。

   下列範例可啟用 FlexCache 和 FlexClone ：

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

步驟 7 ：部署 Cloud Volumes ONTAP for Google Cloud

請使用下列步驟部署 Cloud Volumes ONTAP for Google Cloud 。

步驟

1. 從根資料夾執行下列命令以觸發部署：

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

   觸發兩個容器、第一個容器會部署 Cloud Volumes ONTAP 、第二個容器則會將遙測資料傳送至 AutoSupport 。

   第二個容器會等待、直到第一個容器成功完成所有步驟為止。

2. 使用記錄檔監控部署程序的進度：

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

   此命令會即時提供輸出、並擷取下列記錄檔中的資料：
   `deployment.log`

```
telemetry_asup.log
```

您可以使用下列環境變數編輯檔案、以變更這些記錄檔的名稱 `.env`：

```
DEPLOYMENT_LOGS

TELEMETRY_ASUP_LOGS
```

下列範例說明如何變更記錄檔名稱：

```
DEPLOYMENT_LOGS=<your_deployment_log_filename>.log

TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log
```

完成後

您可以使用下列步驟移除暫存環境、並清除部署程序期間建立的項目。

步驟

1. 如果您部署了 FlexCache 、請在檔案中設定下列選項 `terraform.tfvars`、這樣會清除 FlexCache 磁碟區、並移除先前建立的暫存環境。

```
flexcache_operation = "destroy"
```

> 可能的選項有 `deploy`和 `destroy`。

2. 如果您部署了 FlexClone 、請在檔案中設定下列選項 `terraform.tfvars`、這樣會清除 FlexClone 磁碟區、並移除先前建立的暫存環境。

```
flexclone_operation = "destroy"
```

> 可能的選項有 `deploy`和 `destroy`。

ONTAP

第 0/1 天

ONTAP Day 0/1 解決方案總覽

您可以使用 ONTAP Day 0/1 自動化解決方案、使用 Ansible 來部署和設定 ONTAP 叢集。您可以從"BlueXP 自動化目錄"取得解決方案。

靈活的 ONTAP 部署選項

視您的需求而定、您可以使用內部部署硬體或模擬 ONTAP 、使用 Ansible 來部署和設定 ONTAP 叢集。

內部部署硬體

您可以使用執行 ONTAP 的內部部署硬體來部署此解決方案、例如 FAS 或 AFF 系統。您必須使用 Linux VM 來使用 Ansible 來部署和設定 ONTAP 叢集。

模擬 **ONTAP**

若要使用 ONTAP 模擬器部署此解決方案、您必須從 NetApp 支援網站下載最新版的模擬 ONTAP 。模擬 ONTAP 是 ONTAP 軟體的虛擬模擬器。模擬 ONTAP 在 Windows 、 Linux 或 Mac 系統上的 VMware Hypervisor 中執行。對於 Windows 和 Linux 主機、您必須使用 VMware Workstation Hypervisor 來執行此解決方案。如果您有 Mac OS 、請使用 VMware Fusion Hypervisor 。

分層設計

Ansible 架構可簡化自動化執行與邏輯工作的開發與重複使用。此架構可區分決策工作（邏輯層）和自動化的執行步驟（執行層）。瞭解這些層的運作方式、可讓您自訂組態。

Ansible 「教戰手冊」從開始到結束都會執行一系列工作。 `site.yml`本教戰手冊包含了 `logic.yml`教戰手冊和 `execution.yml`教戰手冊。

執行要求時、教戰手冊會 `site.yml`先呼叫 `logic.yml`教戰手冊、然後呼叫 `execution.yml`教戰手冊以執行服務要求。

您不需要使用架構的邏輯層。邏輯層提供選項、可將架構的功能擴充至硬式編碼的執行值以外的範圍。這可讓您視需要自訂架構功能。

邏輯層

邏輯層由下列項目組成：

- 教 `logic.yml`戰手冊
- 目錄中的邏輯工作檔案 `logic-tasks`

邏輯層提供複雜決策的功能、無需大量自訂整合（例如連線至 ServiceNow ）。邏輯層是可設定的、可為微服務提供輸入。

此外也提供繞過邏輯層的功能。如果您想略過邏輯層、請勿定義 `logic_operation`變數。直接呼叫 `logic.yml`教戰手冊可讓您在不執行的情況下進行某種程度的偵錯。您可以使用「 DEBUG 」陳述式來驗證的值是否 `raw_service_request`正確。

重要考量：

- 教戰手冊會 `logic.yml`搜尋 `logic_operation`變數。如果在要求中定義變數、它會從目錄載入工作檔案 `logic-tasks`。工作檔案必須是 .yml 檔案。如果沒有相符的工作檔案且已定義變數、則 `logic_operation`邏輯層會失敗。
- 變數的預設值 `logic_operation`為 `no-op`。如果未明確定義變數、則預設為 `no-op`、不會執行任何作業。
- 如果 `raw_service_request`已定義變數、則執行會繼續執行至執行層。如果未定義變數、則邏輯層會失敗。

執行層

執行層由下列項目組成：

- 教 `execution.yml`戰手冊

執行層會呼叫 API 以設定 ONTAP 叢集。 `execution.yml`教戰手冊要求 `raw_service_request`在執行時定義變數。

支援自訂

您可以根據自己的需求、以各種方式自訂此解決方案。

自訂選項包括：

- 修改 Ansible 教戰手冊
- 新增角色

自訂 **Ansible** 檔案

下表說明本解決方案所包含的可自訂 Ansible 檔案。

位置	說明
`playbooks/inventory/hosts`	包含一個包含主機和群組清單的檔案。
`playbooks/group_vars/all/*`	Ansible 提供一種便利的方法、可一次將變數套用至多個主機。您可以修改此文件夾中的任何或所有文件，包括 `cfg.yml`、、 `clusters.yml` `defaults.yml` `services.yml` `standards.yml`、和 `vault.yml`。
`playbooks/logic-tasks`	支援 Ansible 內部的決策工作、並維持邏輯與執行的分離。您可以將檔案新增至此資料夾、以對應至相關服務。
`playbooks/vars/*`	Ansible 教戰手冊和角色中使用的動態值、可實現組態的自訂、靈活度及重新使用。如有必要、您可以修改此資料夾中的任何或所有檔案。

自訂角色

您也可以透過新增或變更 Ansible 角色（也稱為微服務）來自訂解決方案。如需詳細資訊"自訂"、請參閱。

準備使用 **ONTAP Day 0/1** 解決方案

部署自動化解決方案之前、您必須先準備好 ONTAP 環境、並安裝及設定 Ansible 。

初始規劃考量

在使用此解決方案部署 ONTAP 叢集之前、您應該先檢閱下列需求和考量事項。

基本需求

若要使用此解決方案、您必須符合下列基本要求：

- 您必須能夠在內部部署或透過 ONTAP 模擬器存取 ONTAP 軟體。
- 您必須知道如何使用 ONTAP 軟體。
- 您必須知道如何使用 Ansible 自動化軟體工具。

規劃考量

部署此自動化解決方案之前、您必須先決定：

- 執行 Ansible 控制節點的位置。

- ONTAP 系統、內部部署硬體或 ONTAP 模擬器。
- 您是否需要自訂。

準備 ONTAP 系統

無論您是使用內部部署 ONTAP 系統或模擬 ONTAP 、都必須先準備好環境、才能部署自動化解決方案。

您也可以選擇安裝及設定模擬 ONTAP

如果您想要透過 ONTAP 模擬器部署此解決方案、則必須下載並執行模擬 ONTAP 。

開始之前

- 您必須下載並安裝要用來執行模擬 ONTAP 的 VMware Hypervisor 。
  - 如果您有 Windows 或 Linux 作業系統、請使用 VMware Workstation 。
  - 如果您有 Mac OS 、請使用 VMware Fusion 。

    如果您使用的是 Mac OS 、則必須使用 Intel 處理器。

步驟

請使用下列程序在您的本機環境中安裝兩個 ONTAP 模擬器：

1. 從下載模擬 ONTAP "NetApp 支援網站"。

   雖然您安裝了兩個 ONTAP 模擬器、但只需下載一份軟體複本。

2. 如果尚未執行、請啟動 VMware 應用程式。
3. 找到下載的模擬器檔案、然後按一下滑鼠右鍵、以 VMware 應用程式開啟該檔案。
4. 設定第一個 ONTAP 執行個體的名稱。
5. 等待模擬器開機、並依照指示建立單一節點叢集。

   針對第二個 ONTAP 執行個體重複步驟。

6. 或者、您也可以新增完整的磁碟補充。

   從每個叢集執行下列命令：

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

ONTAP 系統狀態

您必須驗證 ONTAP 系統的初始狀態、無論是內部部署或透過 ONTAP 模擬器執行。

確認符合下列 ONTAP 系統需求：

- ONTAP 已安裝並在尚未定義叢集的情況下執行。
- ONTAP 已開機並顯示存取叢集的 IP 位址。
- 可連線至網路。
- 您擁有管理認證。
- 當日訊息（ MOTD ）橫幅會顯示管理位址。

安裝所需的自動化軟體

本節提供如何安裝 Ansible 及準備部署自動化解決方案的資訊。

安裝 Ansible

Ansible 可以安裝在 Linux 或 Windows 系統上。

Ansible 用於與 ONTAP 叢集通訊的預設通訊方法是 SSH 。

請參閱"NetApp與Ansible快速入門：安裝Ansible"以安裝 Ansible 。

> Ansible 必須安裝在系統的控制節點上。

下載並準備自動化解決方案

您可以使用下列步驟下載並準備部署的自動化解決方案。

1. 透過 BlueXP Web UI 下載"ONTAP - Day 0/1 ；健全狀況檢查"自動化解決方案。解決方案包裝為 `ONTAP_DAY0_DAY1.zip`。
2. 解壓縮 zip 資料夾、並將檔案複製到 Ansible 環境中控制節點上的所需位置。

初始 Ansible 架構組態

執行 Ansible 架構的初始組態：

1. 瀏覽至 `playbooks/inventory/group_vars/all`。
2. 解密 `vault.yml`檔案：

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

   當系統提示您輸入資料保險箱密碼時、請輸入下列暫時密碼：

   ```
   NetApp123!
   ```

"NetApp123!" 是一種用來解密檔案和對應資料保險箱密碼的暫 `vault.yml`存密碼。第一次使用後、您 * 必須 * 使用自己的密碼來加密檔案。

3. 修改下列 Ansible 檔案：
   - `clusters.yml`- 修改此檔案中的值以符合您的環境。
   - `vault.yml`- 解密檔案後、請修改 ONTAP 叢集、使用者名稱和密碼值、以符合您的環境。
   - `cfg.yml`- 設定的檔案路徑 `log2file`，並在 [設定] 底下 `cfg`**設定** `show_request`**為** `True`[顯示 `raw_service_request`] 。

     此 `raw_service_request`變數會在記錄檔和執行期間顯示。

     列出的每個檔案都包含註解、並說明如何根據您的需求進行修改。

4. 重新加密 `vault.yml`檔案：

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

   系統會提示您在加密時為資料保險箱選擇新密碼。

5. 瀏覽 `playbooks/inventory/hosts`並設定有效的 Python 解譯器。
6. 部署 `framework_test`服務：

   下列命令會以值為的 `cluster_identity_info`**方式執行** `na_ontap_info`**模組** `gather_subset`。這會驗證基本組態是否正確、並確認您可以與叢集通訊。

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

   為每個叢集執行命令。

   如果成功、您應該會看到類似下列範例的輸出：

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

使用解決方案部署 **ONTAP** 叢集

完成準備和規劃之後、您就可以使用 ONTAP Day 0/1 解決方案、使用 Ansible 快速設定 ONTAP 叢集。

在本節步驟中的任何時間、您都可以選擇測試要求、而非實際執行要求。若要測試要求、請將命令列上的教戰手冊變更 `site.yml`為 `logic.yml`。

> `docs/tutorial-requests.txt`此位置包含此程序中所使用之所有服務要求的最終版本。如果執行服務要求時遇到困難、您可以將相關要求從檔案複製 `tutorial-requests.txt`到該 `playbooks/inventory/group_vars/all/tutorial-requests.yml`位置、並視需要修改硬編碼值（ IP 位址、集合名稱等）。接著您應該能夠成功執行要求。

開始之前

- 您必須安裝 Ansible 。
- 您必須已下載 ONTAP Day 0/1 解決方案、並將資料夾解壓縮至 Ansible 控制節點上所需的位置。
- ONTAP 系統狀態必須符合要求、而且您必須擁有必要的認證。
- 您必須已完成本節所述的所有必要工作"準備"。

> 本解決方案中的範例使用「 Cluster_01 」和「 Cluster_02 」作為兩個叢集的名稱。您必須使用環境中叢集的名稱來取代這些值。

步驟 **1** ：初始叢集組態

在此階段、您必須執行一些初始叢集組態步驟。

步驟

1. 瀏覽至該 `playbooks/inventory/group_vars/all/tutorial-requests.yml`位置、並檢閱 `cluster_initial`檔案中的要求。為您的環境進行任何必要的變更。
2. 在資料夾中建立服務要求的檔案 `logic-tasks`。例如，建立名為的檔案 `cluster_initial.yml`。

   將下列各行複製到新檔案：

```
- 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. 定義 `raw_service_request`變數。

   您可以使用下列其中一個選項、在您在資料夾中建立的檔案 `logic-tasks`中定義 `raw_service_request`變數 `cluster_initial.yml`：

   。* 選項 1* ：手動定義 `raw_service_request`變數。

   使用編輯器開啟 `tutorial-requests.yml`檔案、並將內容從第 `11` 行複製到第 `165` 行。將內容貼到新檔案中的變數 `cluster_initial.yml`下方 `raw service request`、如下列範例所示：

顯示範例

範例 `cluster_initial.yml`檔案：

```
- 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 }}"
```

- * 選項 2* ：使用忍者範本定義要求：

  您也可以使用下列 Jinja 範本格式來取得 `raw_service_request`值。

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

4. 為第一個叢集執行初始叢集組態：

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

繼續之前、請確認沒有錯誤。

5. 對第二個叢集重複執行命令：

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

確認第二個叢集沒有錯誤。

往上捲動至 Ansible 輸出的開頭時、您應該會看到傳送至架構的要求、如下列範例所示：

顯示範例


```
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",
```

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

6. 登入每個 ONTAP 執行個體、並驗證要求是否成功。

步驟 **2** ：設定叢集間的生命週期

您現在可以將 LIF 定義新增至要求並定義微服務、 `ontap_interface`以設定叢集間的生命 `cluster_initial`體。

服務定義和要求會共同運作、以決定行動：

- 如果您提供的微服務服務請求不在服務定義中、則不會執行該要求。
- 如果您在服務定義中定義了一或多個微服務、但在要求中省略、則不會執行該要求。

教戰手冊會 `execution.yml`依所列順序掃描微服務清單、以評估服務定義：

- 如果要求中有一個項目的字典金鑰與微服務定義中包含的項目相符 `args`、則會執行該要求。
- 如果服務要求中沒有相符的項目、則會跳過該要求、而不會發生錯誤。

步驟

1. 瀏覽至 `cluster_initial.yml`您先前建立的檔案、並在要求定義中新增下列行以修改要求：

```
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. 執行命令：

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

3. 登入每個執行個體、檢查是否已將生命期新增至叢集：

   顯示範例

```
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.
```

   輸出顯示已添加 * 非 * 的生命。這是因為 `ontap_interface`仍需要在檔案中定義微服務 `services.yml`。

4. 確認已將生命項新增至 `raw_service_request`變數。

顯示範例

以下範例顯示已將生命提升新增至要求：

```
            "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. 在檔案中 `services.yml`的下定義 `ontap_interface`微服務 `cluster_initial`。

   將下列各行複製到檔案中以定義微服務：

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

6. 現在已 `ontap_interface`在要求和檔案中定義微服務 `services.yml`、請再次執行要求：

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

7. 登入每個 ONTAP 執行個體、並確認已新增生命。

步驟 **3** ：選擇性地設定多個叢集

如果需要、您可以在同一個要求中設定多個叢集。定義要求時、您必須為每個叢集提供變數名稱。

步驟

1. 在檔案中新增第二個叢集的項目 `cluster_initial.yml`、以便在同一個要求中設定兩個叢集。

   下列範例顯示 `ontap_aggr`新增第二個項目之後的欄位。

```
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. 對下的所有其他項目套用變更 `cluster_initial`。
3. 將下列各行複製到檔案中、將叢集對等關係新增到要求：

```
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. 執行 Ansible 要求：

```
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>
```

步驟 4 ：初始 SVM 組態

在本程序的這個階段、您可以在叢集中設定 SVM 。

步驟

1. 更新 `svm_initial`檔案中的要求 `tutorial-requests.yml`、以設定 SVM 和 SVM 對等關係。

   您必須設定下列項目：

   - SVM
   - SVM 對等關係

- 每個 SVM 的 SVM 介面

2. 更新要求定義中的變數定義 `svm_initial`。您必須修改下列變數定義：
   - `cluster_name`
   - `vserver_name`
   - `peer_cluster_name`
   - `peer_vserver`

   若要更新定義、請在 `svm_initial`定義之後移除 * 「 {} 」 * 、然後 `req_details`新增正確的定義。

3. 在資料夾中建立服務要求的檔案 `logic-tasks`。例如，建立名為的檔案 `svm_initial.yml`。

   將下列各行複製到檔案：

```
- 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. 定義 `raw_service_request`變數。

   您可以使用下列其中一個選項、在資料夾中 `logic-tasks`定義 `raw_service_request`的變數 `svm_initial`：

   - * 選項 1* ：手動定義 `raw_service_request`變數。

     使用編輯器開啟 `tutorial-requests.yml`檔案、並將內容從第 179 行複製到第 222 行。將內容貼到新檔案中的變數 `svm_initial.yml`下方 `raw service request、如下列範例所示：

```
    service:          svm_initial
    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
```

顯示範例

範例 `svm_initial.yml`檔案：

```
- 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
```

。* 選項 2* ：使用忍者範本定義要求：

您也可以使用下列 Jinja 範本格式來取得 `raw_service_request`值。

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

5. 執行要求：

```
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. 登入每個 ONTAP 執行個體並驗證組態。
7. 新增 SVM 介面。

在檔案中的 `services.yml`下定義 `ontap_interface`服務 `svm_initial`、然後再次執行要求：

```
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. 登入每個 ONTAP 執行個體、並確認 SVM 介面已設定完成。

步驟 **5** ：選擇性地動態定義服務要求

在先前的步驟中、 `raw_service_request`變數是硬式編碼的。這對學習、開發和測試都很有用。您也可以動態產生服務要求。

如果您不想將其與較高層級的系統整合、則下節提供動態產生所需的選項 `raw_service_request`。

- 如果未在命令中定義變數、 `logic.yml`則 `logic_operation`檔案不會從資料夾匯入任何檔案 `logic-tasks`。這表示`raw_service_request`必須在 Ansible 之外定義、並提供給執行架構。
- 資料夾中的工作檔案名稱必須符合不含 .yml 副檔名 `logic-tasks`的變數值 `logic_operation`。
- 資料夾中的工作檔案 `logic-tasks`會動態定義 `raw_service_request`。唯一的需求是將有效`raw_service_request`定義為相關檔案中的最後一項工作。

如何動態定義服務要求

有多種方法可以套用邏輯工作來動態定義服務要求。以下列出其中一些選項：

- 使用資料夾中的 Ansible 工作檔案 `logic-tasks`
- 啟動可傳回適合轉換為 varaible 之資料的自訂角色 `raw_service_request`。
- 在 Ansible 環境以外調用其他工具以提供所需的資料。例如、 REST API 呼叫 Active IQ Unified Manager 。

下列命令範例會使用檔案動態定義每個叢集的服務要求 `tutorial-requests.yml`：

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

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

步驟 **6** ：部署 **ONTAP Day 0/1** 解決方案

在此階段、您應該已經完成下列工作：

- 根據您的需求檢閱及修改中的所有檔案 `playbooks/inventory/group_vars/all`。每個檔案中都有詳細的註解、可協助您進行變更。
- 已將任何必要的工作檔案新增至 `logic-tasks`目錄。
- 已將任何必要的資料檔案新增至 `playbook/vars`目錄。

使用下列命令部署 ONTAP Day 0/1 解決方案、並驗證部署的健全狀況：

在此階段、您應該已經解密並修改 `vault.yml`檔案、而且必須使用新密碼來加密。

- 執行 ONTAP Day 0 服務：

```
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>
```

- 執行 ONTAP Day 1 服務：

```
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>
```

- 套用叢集整體設定：

```
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>
```

- 執行健全狀況檢查：

```
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>
```

自訂 **ONTAP Day 0/1** 解決方案

若要根據您的需求自訂 ONTAP Day 0/1 解決方案、您可以新增或變更 Ansible 角色。

角色代表 Ansible 架構內的微服務。每項微服務都會執行一項作業。例如、 ONTAP Day 0 是包含多個微服務的服務。

新增 **Ansible** 角色

您可以新增 Ansible 角色、針對您的環境自訂解決方案。必要角色是由 Ansible 架構內的服務定義所定義。

角色必須符合下列需求、才能用作微服務：

- 接受變數中的引數清單 `args`。
- 使用 Ansible 「區塊、救援、永遠」結構、並針對每個區塊提供特定需求。
- 使用單一 Ansible 模組、在區塊內定義單一工作。
- 根據本節所詳述的要求實作每個可用的模組參數。

必要的微服務架構

每個角色都必須支援下列變數：

- `mode`：如果模式設置爲角色，則 `test`會嘗試導入，以顯示角色在未實際執行的情況下執行的 `test.yml`操作。

  

  由於某些相依性、因此不一定能實作此項目。

- `status`：教戰手冊執行的整體狀態。如果值未設定為角色、則 `success`不會執行。
- `args`：具有與角色參數名稱匹配的關鍵字的角色特定字典列表。
- `global_log_messages`：在執行教戰手冊期間收集記錄訊息。每次執行角色時都會產生一個項目。
- `log_name`：用於引用條目的角色的名稱 `global_log_messages`。
- `task_descr`：職務內容的簡短說明。
- `service_start_time`：用於追蹤每個角色執行時間的時間戳記。
- `playbook_status`： Ansible 教戰手冊的狀態。
- `role_result`：包含角色輸出的變數、會包含在項目中的每則訊息 `global_log_messages`中。

角色結構範例

以下範例提供實作微服務之角色的基本結構。您必須針對組態變更本範例中的變數。

顯示範例

基本角色結構：

```
- 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 ] }}"
```

範例角色中使用的變數：

- <NAME>：必須為每個微服務提供可更換的值。
- <LOG_NAME>：用於記錄的角色的簡短名稱。例如 ONTAP_VOLUME：。
- <TASK_DESCRIPTION>：微服務的功能簡介。
- <MODULE_NAME>：任務的 Ansible 模塊名稱。

> 最上層的 `execute.yml`教戰手冊會指定 `netapp.ontap`集合。如果模組是集合的一部分、則 `netapp.ontap`不需要完整指定模組名稱。

- <MODULE_SPECIFIC_PARAMETERS>：特定於用於實施微服務的模塊的可接受模塊參數。下列清單說明參數類型及其分組方式。
  - 必要參數：指定所有必要參數時不使用預設值。
  - 具有微服務特定預設值的參數（與模組文件所指定的預設值不同）。
  - 所有剩餘參數都會用 `default(omit)`作預設值。

使用多層字典做為模組參數

某些 NetApp 提供的 Ansible 模組會使用多層級字典來處理模組參數（例如、固定和調適性 QoS 原則群組）。

使用這些字典時、單獨使用 `default(omit)`並不適用、尤其是當有多個字典且彼此互斥時。

如果您需要使用多層字典做為模組參數、則應將功能分割成多個微服務（角色）、以保證每個字典都能為相關字典提供至少一個二層字典值。

下列範例顯示固定和自適應 QoS 原則群組、可在兩個微服務之間分割。

第一個微服務包含固定的 QoS 原則群組值：

```
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)
}}"
```

第二個微服務包含調適性 QoS 原則群組值：

```
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) }}"
```

NetApp 產品 API

版本9 ONTAP

NetApp ONTAP 是業界領先的雲端和內部部署資料管理軟體。ONTAP 包含單一通用 REST API 、每個版本都會持續擴充和增強。請參閱下列文件及相關資源、以開始使用 ONTAP REST API 自動化 ONTAP 部署。

靜態API ONTAP

您可以使用 REST API 來自動化 ONTAP 部署的管理。

- "自動化文件ONTAP"

ONTAP 系列

ONTAP 系列文件包含您安裝及管理 ONTAP 部署所需的一切。

- "產品文件ONTAP"

BlueXP 控制面板

NetApp BlueXP BlueXP 是統一化的控制平台、提供混合式多雲端平台、可在內部部署和公有雲環境中管理儲存和資料服務。它由多個不同的服務或元件組成、每個服務或元件都會公開相關的 REST API 。請參閱下列文件及相關資源、以開始使用 BlueXP REST API 自動化 BlueXP 環境。

BlueXP REST API

您可以使用各種 REST API 來自動化管理 BlueXP 所管理的儲存和資料服務。

- "BlueXP API 文件"

BlueXP 系列

BlueXP 系列文件包含開始使用 BlueXP 控制面板所需的一切資料。

- "BlueXP文件"

Astra Control

Astra Control 是 NetApp 軟體產品、可在多種環境中提供 Kubernetes 叢集的應用程式感知資料管理功能。這兩種部署模式共用一個通用的 REST API 。請參閱下列文件和相關資源、開始使用 Astra Control REST API 自動化您的 Astra 部署。

Astra Control REST API

您可以使用 REST API 將 Astra Control Service 和 Astra Control Center 部署的管理自動化。

- "Astra Control Automation文件"

Astra系列

Astra 系列文件包含您安裝和管理 Astra 及相關軟體所需的一切。

- "Astra文件"

Active IQ Unified Manager

Active IQ Unified Manager （前身為 OnCommand Unified Manager ）可為您的 ONTAP 系統提供全方位的管理與監控功能。它也包含 REST API 、可將這些工作自動化、並在支援的系統上啟用協力廠商整合。

REST API 的優點

使用 Active IQ Unified Manager 隨附的 REST API 有幾項優點。

強大的功能

使用 REST API 、您可以存取 Active IQ Unified Manager 功能來管理儲存可用度、容量、安全性、保護和效能風險。

自動化整合

有單一 REST 端點可供配置及管理工作負載。這提供簡單的整合方法、可實作服務層級目標原則、將事件重新導向至協力廠商工具、並作為 API 閘道、在個別叢集層級存取 ONTAP REST API 。

ONTAP 管理的資料中心層級自動化

您可以在資料中心層級將 ONTAP 資源配置與管理工作流程自動化。監控與報告也可以自動化。如此可提升效率、並為資料中心層級集合提供基礎。

取得更多資訊

有幾種資源可協助您開始使用 Active IQ Unified Manager REST API 。

- "開始使用Active IQ Unified Manager 靜態API"
- "Active IQ Unified Manager API 文件"
- "適用於 Ansible 的 NetApp 模組"

知識與支援

其他資源

您可以存取其他資源來取得協助、並尋找更多關於 NetApp 產品和雲端服務的資訊。

NetApp 開發人員資源

- "NetApp DevNet"

  支援 NetApp 合作夥伴和客戶的開發人員資源的中央位置。

- "NetApp IO - The Pub"

  部落格文章和其他資源、以支援開發人員和管理員。

NetApp雲端資源

- "NetApp BlueXP"

  NetApp雲端解決方案的中央網站。

- "NetApp Cloud Central主控台"

  使用登入功能的NetApp Cloud Central服務主控台。

取得協助

NetApp 以多種方式支援其產品和雲端服務。我們全年無休提供豐富的免費自助支援選項、例如知識庫（KB）文章和社群論壇。

自我支援選項

- "知識庫"

  搜尋知識庫、找出有助於疑難排解問題的文章。

- "社群"

  加入 NetApp 社群、追蹤後續的討論或建立新的討論。

- "NetApp支援"

  存取疑難排解工具、文件及技術支援協助。

法律聲明

法律聲明提供版權聲明、商標、專利等存取權限。

版權

"https://www.netapp.com/company/legal/copyright/"

商標

NetApp 、 NetApp 標誌及 NetApp 商標頁面上列出的標章均為 NetApp 、 Inc. 的商標。其他公司與產品名稱可能為其各自所有者的商標。

"https://www.netapp.com/company/legal/trademarks/"

專利

如需最新的 NetApp 擁有專利清單、請參閱：

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

隱私權政策

"https://www.netapp.com/company/legal/privacy-policy/"

開放原始碼

通知檔案提供有關 NetApp 軟體所使用之協力廠商版權與授權的資訊。