Automazione NetApp

NetApp Automation

NetApp
October 23, 2024

Sommario

Automazione NetApp

Novità di automazione NetApp

NetApp aggiorna regolarmente le soluzioni di automazione, le API REST DEI prodotti e il software correlato per offrire nuove funzionalità, miglioramenti e correzioni dei bug.

27 luglio 2023

Cloud Volumes ONTAP

Il supporto per Cloud Volumes ONTAP è organizzato dall'ambiente di cloud pubblico. È prevista una nuova soluzione per il seguente ambiente cloud:

- "Cloud Volumes ONTAP per Google Cloud - burst nel cloud"

04 giugno 2023

NetApp "Catalogo di automazione BlueXP" è disponibile tramite l'interfaccia utente Web di BlueXP . Il catalogo di automazione fornisce accesso alle soluzioni che possono aiutarti con l'implementazione e l'integrazione automatizzate dei prodotti NetApp. La documentazione per queste soluzioni è organizzata in diverse aree funzionali o di prodotto, come descritto di seguito.

Amazon FSX per NetApp ONTAP

Due soluzioni Amazon FSX per NetApp ONTAP vengono fornite come segue:

- "Amazon FSX per NetApp ONTAP - burst nel cloud"
- "Amazon FSX per NetApp ONTAP - disaster recovery"

Azure NetApp Files

È inclusa una soluzione per facilitare la distribuzione di Oracle con Azure NetApp Files:

- "Oracle con Azure NetApp Files"

Cloud Volumes ONTAP

Il supporto per Cloud Volumes ONTAP è organizzato dall'ambiente di cloud pubblico. Le soluzioni iniziali sono fornite per due ambienti cloud come segue:

- "Cloud Volumes ONTAP AWS - burst nel cloud"
- "Cloud Volumes ONTAP, Azure - burst nel cloud"

Catalogo di automazione BlueXP

Panoramica del catalogo di automazione BlueXP

Il catalogo di automazione BlueXP è una raccolta di soluzioni di automazione disponibili per clienti, partner e dipendenti di NetApp. Il catalogo ha diverse caratteristiche e vantaggi.

Un'unica posizione per le tue esigenze di automazione

È possibile accedere a "Catalogo di automazione BlueXP" tramite l'interfaccia utente Web di BlueXP . In questo modo è possibile trovare un'unica posizione per gli script, i playbook e i moduli necessari per migliorare l'automazione e il funzionamento dei prodotti e dei servizi NetApp.

Le soluzioni vengono create e testate da NetApp

Tutte le soluzioni di automazione e gli script sono stati creati e testati da NetApp. Ogni soluzione è rivolta a casi d'utilizzo o richieste specifici del cliente. La maggior parte si concentra sull'integrazione con i servizi file e dati NetApp.

Documentazione

Ciascuna soluzione di automazione include la documentazione associata per iniziare. Sebbene sia possibile accedere alle soluzioni tramite l'interfaccia Web di BlueXP , tutta la documentazione è disponibile in questo sito. La documentazione è organizzata in base ai prodotti e ai servizi cloud di NetApp.

Solide fondamenta per il futuro

NetApp si impegna ad aiutare i nostri clienti a migliorare e ottimizzare l'automazione dei data center e degli ambienti cloud. Prevediamo di continuare a migliorare il catalogo di automazione BlueXP per soddisfare le esigenze dei clienti, le modifiche tecnologiche e la continua integrazione dei prodotti.

Vogliamo conoscere la tua opinione

Il team di automazione dell'ufficio Customer Experience Office (CXO) di NetApp vorrebbe ricevere una tua opinione. In caso di feedback, problemi o richieste di funzionalità, invia un'e-mail a team di automazione cxo.

Amazon FSX per NetApp ONTAP

Amazon FSX per NetApp ONTAP - burst nel cloud

Puoi usare questa soluzione di automazione per eseguire il provisioning di Amazon FSX per NetApp ONTAP con dei volumi e di un FlexCache associato.

Amazon FSX per NetApp ONTAP è anche chiamato **FSX per ONTAP**.

Informazioni sulla soluzione

Ad un livello elevato, il codice di automazione fornito con questa soluzione esegue le seguenti azioni:

- Esegui il provisioning di un file system FSX di destinazione per ONTAP
- Eseguire il provisioning delle Storage Virtual Machine (SVM) per il file system
- Creare una relazione di peering dei cluster tra i sistemi di origine e di destinazione

- Creare una relazione di peering delle SVM tra il sistema di origine e il sistema di destinazione per FlexCache
- Puoi anche creare volumi FlexVol usando FSX per ONTAP
- Crea un volume FlexCache in FSX per ONTAP con l'origine che punta allo storage on-premise

L'automazione si basa su Docker e Docker Compose, che devono essere installati nella macchina virtuale Linux come descritto di seguito.

Prima di iniziare

Per completare il provisioning e la configurazione, è necessario disporre dei seguenti elementi:

- È necessario scaricare la "Amazon FSX per NetApp ONTAP - burst nel cloud" soluzione di automazione tramite l'interfaccia utente Web di BlueXP . La soluzione viene predisposta come file `AWS_FSxN_BTC.zip`.
- Connettività di rete tra i sistemi di origine e di destinazione.
- Una VM Linux con le seguenti caratteristiche:
  - Distribuzione Linux basata su Debian
  - Implementato sullo stesso sottoinsieme VPC utilizzato per FSX per il provisioning ONTAP
- Account AWS.

Fase 1: Installazione e configurazione di Docker

Installare e configurare Docker in una macchina virtuale Linux basata su Debian.

Fasi

1. Preparare l'ambiente.

```
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. Installare Docker e verificare l'installazione.

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

3. Aggiungere il gruppo Linux richiesto con un utente associato.

   Controlla prima se il gruppo **docker** esiste nel tuo sistema Linux. In caso contrario, creare il gruppo e aggiungere l'utente. Per impostazione predefinita, l'utente della shell corrente viene aggiunto al gruppo.

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

4. Attivare le nuove definizioni di gruppo e utente

   Se è stato creato un nuovo gruppo con un utente, è necessario attivare le definizioni. Per fare questo, si può disconnettersi da Linux e poi tornare indietro. Oppure si può eseguire il seguente comando.

```
newgrp docker
```

Fase 2: Installare Docker Compose

Installare Docker Compose in una macchina virtuale Linux basata su Debian.

Fasi

1. Installazione di 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. Verificare che l'installazione sia riuscita.

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

Fase 3: Preparare l'immagine Docker

Occorre estrarre e caricare l'immagine Docker fornita con la soluzione di automazione.

Fasi

1. Copiare il file della soluzione `AWS_FSxN_BTC.zip` nella macchina virtuale in cui verrà eseguito il codice di automazione.

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

   Il parametro di input `private-key.pem` è il file della chiave privata utilizzato per l'autenticazione della macchina virtuale AWS (istanza EC2).

2. Individuare la cartella corretta con il file della soluzione e decomprimere il file.

```
unzip AWS_FSxN_BTC.zip
```

3. Passare alla nuova cartella `AWS_FSxN_BTC` creata con l'operazione di decompressione ed elencare i file. Dovrebbe essere visualizzato il file `aws_fsxn_flexcache_image_latest.tar.gz`.

```
ls -la
```

4. Caricare il file di immagine Docker. L'operazione di carico dovrebbe normalmente essere completata in pochi secondi.

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

5. Verificare che l'immagine Docker sia caricata.

```
docker images
```

Si dovrebbe vedere l'immagine Docker `aws_fsxn_flexcache_image` con il tag `latest`.

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

Fase 4: Creare il file dell'ambiente per le credenziali AWS

È necessario creare un file variabile locale per l'autenticazione utilizzando la chiave di accesso e segreta. Quindi aggiungere il file al `.env` file.

Fasi

1. Creare il `awsauth.env` file nella seguente posizione:

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

2. Aggiungere il seguente contenuto al file:

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

   Il formato **deve** essere esattamente come mostrato sopra senza spazi tra `key` e `value`.

3. Aggiungere il percorso assoluto del `.env` file utilizzando la `AWS_CREDS` variabile. Ad esempio:

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

Passaggio 5: Creare un volume esterno

È necessario un volume esterno per verificare che i file di stato di Terraform e altri file importanti siano persistenti. Questi file devono essere disponibili affinché Terraform possa eseguire il flusso di lavoro e le distribuzioni.

Fasi

1. Creare un volume esterno all'esterno di Docker Compose.

   Assicurarsi di aggiornare il nome del volume (ultimo parametro) al valore appropriato prima di eseguire il comando.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Aggiungere il percorso del volume esterno al `.env` file di ambiente utilizzando il comando:

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

   Ricordare di mantenere il contenuto del file esistente e la formattazione dei due punti. Ad esempio:

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

   Puoi invece aggiungere una condivisione NFS come volume esterno utilizzando un comando come:

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

3. Aggiornare le variabili Terraform.
   a. Passare alla cartella `aws_fsxn_variables`.
   b. Verificare che esistano i due file seguenti: `terraform.tfvars` E `variables.tf`.
   c. Aggiornare i valori in `terraform.tfvars` come richiesto per il proprio ambiente.

      Per ulteriori informazioni, vedere "Risorsa terraform: aws_fsx_ONTAP_file_system" .

Fase 6: Eseguire il provisioning di Amazon FSX per NetApp ONTAP e FlexCache

È possibile eseguire il provisioning di Amazon FSX per NetApp ONTAP e FlexCache.

Fasi

1. Accedere alla cartella principale (AWS_FSXN_BTC) ed eseguire il comando di provisioning.

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

   Questo comando crea due contenitori. Il primo container implementa FSX per ONTAP, mentre il secondo container crea il peering del cluster, il peering delle SVM, il volume di destinazione e FlexCache.

2. Monitorare il processo di provisioning.

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

Questo comando fornisce l'output in tempo reale, ma è stato configurato per acquisire i log attraverso il file `deployment.log`. È possibile modificare il nome di questi file di registro modificando il `.env` file e aggiornando le variabili `DEPLOYMENT_LOGS`.

Passaggio 7: Distruggi Amazon FSX per NetApp ONTAP e FlexCache

Facoltativamente, puoi eliminare e rimuovere Amazon FSX per NetApp ONTAP e FlexCache.

1. Impostare la variabile `flexcache_operation` nel `terraform.tfvars` file su "Destroy".
2. Accedere alla cartella principale (AWS_FSXN_BTC) ed eseguire il seguente comando.

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

Questo comando crea due contenitori. Il primo contenitore elimina FlexCache e il secondo contenitore elimina FSX per ONTAP.

3. Monitorare il processo di provisioning.

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

Amazon FSX per NetApp ONTAP - disaster recovery

Puoi usare questa soluzione di automazione per effettuare un backup di disaster recovery di un sistema di origine utilizzando Amazon FSX per NetApp ONTAP.

Amazon FSX per NetApp ONTAP è anche chiamato **FSX per ONTAP**.

Informazioni sulla soluzione

Ad un livello elevato, il codice di automazione fornito con questa soluzione esegue le seguenti azioni:

- Esegui il provisioning di un file system FSX di destinazione per ONTAP
- Eseguire il provisioning delle Storage Virtual Machine (SVM) per il file system
- Creare una relazione di peering dei cluster tra i sistemi di origine e di destinazione
- Creare una relazione di peering delle SVM tra il sistema di origine e il sistema di destinazione per SnapMirror
- Creare volumi di destinazione
- Crea una relazione SnapMirror tra i volumi di origine e destinazione
- Avvia il trasferimento SnapMirror tra i volumi di origine e di destinazione

L'automazione si basa su Docker e Docker Compose, che devono essere installati nella macchina virtuale Linux come descritto di seguito.

Prima di iniziare

Per completare il provisioning e la configurazione, è necessario disporre dei seguenti elementi:

- È necessario scaricare la "Amazon FSX per NetApp ONTAP - disaster recovery" soluzione di automazione tramite l'interfaccia utente Web di BlueXP . La soluzione viene confezionata come `FSxN_DR.zip`. Questo file zip contiene il `AWS_FSxN_Bck_Prov.zip` file che verrà utilizzato per distribuire la soluzione descritta in questo documento.
- Connettività di rete tra i sistemi di origine e di destinazione.
- Una VM Linux con le seguenti caratteristiche:
  - Distribuzione Linux basata su Debian
  - Implementato sullo stesso sottoinsieme VPC utilizzato per FSX per il provisioning ONTAP
- Un account AWS.

Fase 1: Installazione e configurazione di Docker

Installare e configurare Docker in una macchina virtuale Linux basata su Debian.

Fasi

1. Preparare l'ambiente.

```
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. Installare Docker e verificare l'installazione.

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

3. Aggiungere il gruppo Linux richiesto con un utente associato.

   Controlla prima se il gruppo **docker** esiste nel tuo sistema Linux. Se non esiste, creare il gruppo e aggiungere l'utente. Per impostazione predefinita, l'utente della shell corrente viene aggiunto al gruppo.

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

4. Attivare le nuove definizioni di gruppo e utente

   Se è stato creato un nuovo gruppo con un utente, è necessario attivare le definizioni. Per fare questo, si

può disconnettersi da Linux e poi tornare indietro. Oppure si può eseguire il seguente comando.

```
newgrp docker
```

Fase 2: Installare Docker Compose

Installare Docker Compose in una macchina virtuale Linux basata su Debian.

Fasi

1. Installazione di 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. Verificare che l'installazione sia riuscita.

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

Fase 3: Preparare l'immagine Docker

Occorre estrarre e caricare l'immagine Docker fornita con la soluzione di automazione.

Fasi

1. Copiare il file della soluzione `AWS_FSxN_Bck_Prov.zip` nella macchina virtuale in cui verrà eseguito il codice di automazione.

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

   Il parametro di input `private-key.pem` è il file della chiave privata utilizzato per l'autenticazione della macchina virtuale AWS (istanza EC2).

2. Individuare la cartella corretta con il file della soluzione e decomprimere il file.

```
unzip AWS_FSxN_Bck_Prov.zip
```

3. Passare alla nuova cartella `AWS_FSxN_Bck_Prov` creata con l'operazione di decompressione ed elencare i file. Dovrebbe essere visualizzato il file `aws_fsxn_bck_image_latest.tar.gz`.

```
ls -la
```

4. Caricare il file di immagine Docker. L'operazione di carico dovrebbe normalmente essere completata in pochi secondi.

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

5. Verificare che l'immagine Docker sia caricata.

```
docker images
```

Si dovrebbe vedere l'immagine Docker `aws_fsxn_bck_image` con il tag `latest`.

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

Fase 4: Creare il file dell'ambiente per le credenziali AWS

È necessario creare un file variabile locale per l'autenticazione utilizzando la chiave di accesso e segreta. Quindi aggiungere il file al `.env` file.

Fasi

1. Creare il `awsauth.env` file nella seguente posizione:

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

2. Aggiungere il seguente contenuto al file:

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

Il formato **deve** essere esattamente come mostrato sopra senza spazi tra `key` e `value`.

3. Aggiungere il percorso assoluto del `.env` file utilizzando la `AWS_CREDS` variabile. Ad esempio:

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

Passaggio 5: Creare un volume esterno

È necessario un volume esterno per verificare che i file di stato di Terraform e altri file importanti siano persistenti. Questi file devono essere disponibili affinché Terraform possa eseguire il flusso di lavoro e le distribuzioni.

Fasi

1. Creare un volume esterno all'esterno di Docker Compose.

   Assicurarsi di aggiornare il nome del volume (ultimo parametro) al valore appropriato prima di eseguire il comando.

   ```
   docker volume create aws_fsxn_volume
   ```

2. Aggiungere il percorso del volume esterno al `.env` file di ambiente utilizzando il comando:

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

   Ricordare di mantenere il contenuto del file esistente e la formattazione dei due punti. Ad esempio:

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

   Puoi invece aggiungere una condivisione NFS come volume esterno utilizzando un comando come:

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

3. Aggiornare le variabili Terraform.
   a. Passare alla cartella `aws_fsxn_variables`.
   b. Verificare che esistano i due file seguenti: `terraform.tfvars` E `variables.tf`.
   c. Aggiornare i valori in `terraform.tfvars` come richiesto per il proprio ambiente.

      Per ulteriori informazioni, vedere "Risorsa terraform: aws_fsx_ONTAP_file_system" .

Fase 6: Distribuzione della soluzione di backup

Puoi implementare e effettuare il provisioning della soluzione di backup per il disaster recovery.

Fasi

1. Accedere alla cartella principale (AWS_FSxN_Bck_Prov) ed eseguire il comando di provisioning.

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

   Questo comando crea tre contenitori. Il primo container implementa FSX per ONTAP. Il secondo container crea il peering del cluster, il peering delle SVM e il volume di destinazione. Il terzo contenitore crea la relazione SnapMirror e avvia il trasferimento SnapMirror.

2. Monitorare il processo di provisioning.

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

   Questo comando fornisce l'output in tempo reale, ma è stato configurato per acquisire i log attraverso il file

`deployment.log`. È possibile modificare il nome di questi file di registro modificando il `.env` file e aggiornando le variabili `DEPLOYMENT_LOGS`.

Azure NetApp Files

Installare Oracle utilizzando Azure NetApp Files

È possibile utilizzare questa soluzione di automazione per il provisioning di Azure NetApp Files Volumes e l'installazione di Oracle su una macchina virtuale disponibile. Oracle quindi utilizza i volumi per lo storage dei dati.

Informazioni sulla soluzione

Ad un livello elevato, il codice di automazione fornito con questa soluzione esegue le seguenti azioni:

- Configurare un account NetApp in Azure
- Configurare un pool di capacità dello storage su Azure
- Eseguire il provisioning dei volumi Azure NetApp Files in base alla definizione
- Creare i punti di montaggio
- Montare i volumi Azure NetApp Files sui punti di montaggio
- Installare Oracle sul server Linux
- Creare i listener e il database
- Creare i database inseribili (PDB)
- Avviare l'istanza listener e Oracle
- Installare e configurare l' `azacsnap`utilità per acquisire un'istantanea

Prima di iniziare

Per completare l'installazione è necessario disporre di quanto segue:

- È necessario scaricare la "Oracle con Azure NetApp Files" soluzione di automazione tramite l'interfaccia utente Web di BlueXP . La soluzione viene predisposta come file `na_oracle19c_deploy-master.zip`.
- Una VM Linux con le seguenti caratteristiche:
  - RHEL 8 (Standard_D8s_v3-RHEL-8)
  - Implementato sulla stessa rete virtuale di Azure utilizzata per il provisioning di Azure NetApp Files
- Un account Azure

La soluzione di automazione viene fornita come immagine ed eseguita con Docker e Docker Compose. È necessario installare entrambi questi componenti sulla macchina virtuale Linux come descritto di seguito.

Si dovrebbe anche registrare la VM con RedHat usando il comando `sudo subscription-manager register`. Il comando richiede di immettere le credenziali dell'account. Se necessario, è possibile creare un account in https://developers.redhat.com/.

Fase 1: Installazione e configurazione di Docker

Installare e configurare Docker in una macchina virtuale Linux RHEL 8.

Fasi

1. Installa il software Docker usando i seguenti comandi.

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

2. Avviare Docker e visualizzare la versione per confermare la riuscita dell'installazione.

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

3. Aggiungere il gruppo Linux richiesto con un utente associato.

   Controlla prima se il gruppo **docker** esiste nel tuo sistema Linux. In caso contrario, creare il gruppo e aggiungere l'utente. Per impostazione predefinita, l'utente della shell corrente viene aggiunto al gruppo.

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

4. Attivare le nuove definizioni di gruppo e utente

   Se è stato creato un nuovo gruppo con un utente, è necessario attivare le definizioni. Per fare questo, si può disconnettersi da Linux e poi tornare indietro. Oppure si può eseguire il seguente comando.

```
newgrp docker
```

Passaggio 2: Installare Docker Compose e le utility NFS

Installare e configurare Docker Compose insieme al pacchetto di utilità NFS.

Fasi

1. Installare Docker Compose e visualizzare la versione per confermare la riuscita dell'installazione.

```
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. Installare il pacchetto NFS Utilities.

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

Passaggio 3: Scaricare i file di installazione di Oracle

Scaricare i file di installazione e patch di Oracle richiesti e l' `azacsnap`utility.

Fasi

1. Accedi al tuo account Oracle in base alle tue esigenze.
2. Scaricare i seguenti file.

File	Descrizione
LINUX.X64_193000_db_home.zip	19,3 programma di installazione di base
p31281355_190000_Linux-x86-64.zip	Patch 19,8 RU
p6880880_190000_Linux-x86-64.zip	opatch versione 12.2.0.1.23
azacsnap_installer_v5.0.run	programma di installazione di azacsnap

3. Inserire tutti i file di installazione nella cartella `/tmp/archive`.
4. Assicurarsi che tutti gli utenti sul server di database abbiano accesso completo (lettura, scrittura, esecuzione) alla cartella `/tmp/archive`.

Fase 4: Preparare l'immagine Docker

Occorre estrarre e caricare l'immagine Docker fornita con la soluzione di automazione.

Fasi

1. Copiare il file della soluzione `na_oracle19c_deploy-master.zip` nella macchina virtuale in cui verrà eseguito il codice di automazione.

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

   Il parametro di input `private-key.pem` è il file della chiave privata utilizzato per l'autenticazione della macchina virtuale Azure.

2. Individuare la cartella corretta con il file della soluzione e decomprimere il file.

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

3. Passare alla nuova cartella `na_oracle19c_deploy-master` creata con l'operazione di decompressione ed elencare i file. Dovrebbe essere visualizzato il file `ora_anf_bck_image.tar`.

```
ls -lt
```

4. Caricare il file di immagine Docker. L'operazione di carico dovrebbe normalmente essere completata in pochi secondi.

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

5. Verificare che l'immagine Docker sia caricata.

```
docker images
```

Si dovrebbe vedere l'immagine Docker `ora_anf_bck_image` con il tag `latest`.

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

Passaggio 5: Creare un volume esterno

È necessario un volume esterno per verificare che i file di stato di Terraform e altri file importanti siano persistenti. Questi file devono essere disponibili affinché Terraform possa eseguire il flusso di lavoro e le distribuzioni.

Fasi

1. Creare un volume esterno all'esterno di Docker Compose.

Assicurarsi di aggiornare il nome del volume prima di eseguire il comando.

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

2. Aggiungere il percorso del volume esterno al `.env` file di ambiente utilizzando il comando:

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

Ricordare di mantenere il contenuto del file esistente e la formattazione dei due punti. Ad esempio:

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

3. Aggiornare le variabili Terraform.
   a. Passare alla cartella `ora_anf_variables`.
   b. Verificare che esistano i due file seguenti: `terraform.tfvars` E `variables.tf`.
   c. Aggiornare i valori in `terraform.tfvars` come richiesto per il proprio ambiente.

Passaggio 6: Installare Oracle

È ora possibile eseguire il provisioning e installare Oracle.

Fasi

1. Installare Oracle utilizzando la seguente sequenza di comandi.

```
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. Ricaricare le variabili Bash e confermare visualizzando il valore di `ORACLE_HOME` .
   a. `cd /home/oracle`
   b. `source .bash_profile`
   c. `echo $ORACLE_HOME`
3. Dovrebbe essere possibile accedere a Oracle.

```
sudo su oracle
```

Passaggio 7: Convalida dell'installazione di Oracle

Verificare che l'installazione di Oracle sia stata eseguita correttamente.

Fasi

1. Accedere al server Oracle Linux e visualizzare un elenco dei processi Oracle. Ciò conferma che l'installazione è stata completata come previsto e che il database Oracle è in esecuzione.

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

2. Accedere al database per esaminare la configurazione del database e verificare che le PDB siano state create correttamente.

```
sqlplus / as sysdba
```

   L'output dovrebbe essere simile a quanto segue:

```
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. Eseguire alcuni semplici comandi SQL per confermare la disponibilità del database.

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

Fase 8: Installare l'utilità azacsnap ed eseguire un backup dello snapshot

È necessario installare ed eseguire l' `azacsnap`utilità per eseguire un backup snapshot.

Fasi

1. Montare il contenitore.

```
docker-compose up azacsnap_install
```

2. Passare all'account utente snapshot.

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

3. Configurare un file di dettagli per il backup dello storage. Questo creerà il `azacsnap.json` file di configurazione.

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

4. Eseguire un backup snapshot.

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

Fase 9: Migrazione opzionale di un PDB on-premise nel cloud

Puoi anche migrare il PDB on-premise nel cloud.

Fasi

1. Impostare le variabili nei `tfvars` file in base alle esigenze dell'ambiente.
2. Eseguire la migrazione del PDB.

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

Cloud Volumes ONTAP per AWS

Cloud Volumes ONTAP per AWS - burst nel cloud

Questo articolo supporta la soluzione di automazione NetApp Cloud Volumes ONTAP per AWS, disponibile per i clienti NetApp nel catalogo di automazione BlueXP .

La soluzione di automazione Cloud Volumes ONTAP per AWS automatizza l'implementazione containerizzata di Cloud Volumes ONTAP per AWS utilizzando Terraform, consentendoti di implementare Cloud Volumes ONTAP per AWS rapidamente, senza interventi manuali.

Prima di iniziare

- È necessario scaricare la "Cloud Volumes ONTAP AWS - burst nel cloud"soluzione di automazione tramite l'interfaccia utente Web di BlueXP . La soluzione viene confezionata come `cvo_aws_flexcache.zip`.
- È necessario installare una macchina virtuale Linux sulla stessa rete di Cloud Volumes ONTAP.
- Dopo aver installato la VM Linux, è necessario seguire la procedura descritta in questa soluzione per installare le dipendenze richieste.

Fase 1: Installare Docker e Docker Compose

Installare Docker

I seguenti passaggi usano il software di distribuzione Linux Ubuntu 20,04 Debian come esempio. I comandi eseguiti dipendono dal software di distribuzione Linux utilizzato. Consultare la documentazione specifica del software di distribuzione Linux per la configurazione in uso.

Fasi

1. Installare Docker eseguendo i seguenti `sudo` comandi:

```
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. Verificare l’installazione:

```
docker –version
```

3. Verificare che sul sistema Linux sia stato creato un gruppo denominato "docker". Se necessario, creare il gruppo:

```
sudo groupadd docker
```

4. Aggiungere al gruppo l’utente che deve accedere a Docker:

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

5. Le modifiche vengono applicate dopo la disconnessione e la riconnessione al terminale. In alternativa, è possibile applicare immediatamente le modifiche:

```
newgrp docker
```

Installazione di Docker Compose

Fasi

1. Installare Docker Compose eseguendo i seguenti `sudo` comandi:

```
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. Verificare l’installazione:

```
docker-compose –version
```

Fase 2: Preparare l'immagine Docker

Fasi

1. Copiare la `cvo_aws_flexcache.zip` cartella nella VM Linux che si desidera utilizzare per distribuire 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` è il file della chiave privata per l'accesso senza password.
   - `awsuser` È il nome utente VM.
   - `IP_ADDRESS_OF_VM` È l'indirizzo IP della macchina virtuale.
   - `LOCATION_TO_BE_COPIED` è la posizione in cui verrà copiata la cartella.

2. Estrarre la `cvo_aws_flexcache.zip` cartella. È possibile estrarre la cartella nella directory corrente o in un percorso personalizzato.

   Per estrarre la cartella nella directory corrente, eseguire:

```
unzip cvo_aws_flexcache.zip
```

   Per estrarre la cartella in una posizione personalizzata, eseguire:

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

3. Dopo aver estratto il contenuto, accedere alla `CVO_Aws_Deployment` cartella ed eseguire il comando seguente per visualizzare i file:

```
 ls -la
```

   Viene visualizzato un elenco di file, simile al seguente esempio:

```
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. Individuare il `cvo_aws_flexcache_ubuntu_image.tar` file. Contenente l'immagine di Docker necessaria per implementare Cloud Volumes ONTAP per AWS.
5. Estrarre il file:

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

6. Attendere alcuni minuti per il caricamento dell'immagine Docker, quindi convalidare il caricamento corretto dell'immagine Docker:

```
docker images
```

Viene visualizzata un'immagine di Docker con il `latest` nome del `cvo_aws_flexcache_ubuntu_image` tag, come illustrato nell'esempio seguente:

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

Se necessario, è possibile modificare il nome dell'immagine di Docker. In caso di modifica del nome dell'immagine di Docker, assicurarsi di aggiornare il nome dell'immagine di Docker nei `docker-compose-deploy` file e. `docker-compose-destroy`

Passaggio 3: Creare file di variabili d'ambiente

A questo punto, è necessario creare due file di variabili d'ambiente. Un file è per l'autenticazione delle API di AWS Resource Manager utilizzando l'accesso ad AWS e le chiavi segrete. Il secondo file serve per impostare le variabili di ambiente in modo da consentire ai moduli BlueXP Terraform di individuare e autenticare le API AWS.

Fasi

1. Creare il `awsauth.env` file nella seguente posizione:

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

   a. Aggiungere il seguente contenuto al `awsauth.env` file:

      access_key=<> secret_key=<>

   Il formato **deve** essere esattamente come mostrato sopra.

2. Aggiungere il percorso assoluto del file al `.env` file.

   Immettere il percorso assoluto per il `awsauth.env` file di ambiente che corrisponde alla `AWS_CREDS` variabile di ambiente.

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

3. Accedere alla `cvo_aws_variable` cartella e aggiornare la chiave di accesso e la chiave segreta nel file delle credenziali.

   Aggiungere il seguente contenuto al file:

   aws_access_key_id=<> aws_secret_access_key=<>

   Il formato **deve** essere esattamente come mostrato sopra.

Passaggio 4: Aggiungere licenze Cloud Volumes ONTAP a BlueXP o sottoscrivere BlueXP

Puoi aggiungere licenze Cloud Volumes ONTAP a BlueXP o iscriverti a NetApp BlueXP nel marketplace AWS.

Fasi

1. Dal portale AWS, accedere a **SaaS** e selezionare **Iscriviti a NetApp BlueXP** .

   È possibile utilizzare lo stesso gruppo di risorse di Cloud Volumes ONTAP o un gruppo di risorse diverso.

2. Configurare il portale BlueXP per importare l'abbonamento SaaS in BlueXP .

   Puoi configurarlo direttamente dal portale AWS.

   Si viene reindirizzati al portale BlueXP per confermare la configurazione.

3. Confermare la configurazione nel portale BlueXP selezionando **Salva**.

Passaggio 5: Creare un volume esterno

È necessario creare un volume esterno per mantenere persistenti i file di stato di Terraform e altri file importanti. È necessario assicurarsi che i file siano disponibili affinché Terraform esegua il flusso di lavoro e le implementazioni.

Fasi

1. Creare un volume esterno all'esterno di Docker Compose:

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

Esempio:

```
docker volume create cvo_aws_volume_dst
```

2. Utilizzare una delle seguenti opzioni:

   a. Aggiungere un percorso di volume esterno al `.env` file di ambiente.

      È necessario seguire il formato esatto mostrato di seguito.

      Formato:

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

      Esempio:
      `PERSISTENT_VOL=cvo_aws_volume_dst:/cvo_aws`

   b. Aggiunta di condivisioni NFS come volume esterno.

      Assicurati che il container di Docker possa comunicare con le condivisioni NFS e che siano configurate le autorizzazioni corrette, come la lettura/scrittura.

      i. Aggiungi il percorso NFS share come percorso del volume esterno nel file Docker Compose, come illustrato sotto: Format:

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

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

3. Accedere alla `cvo_aws_variables` cartella.

   Nella cartella dovrebbe essere visualizzato il seguente file variabile:

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

4. Modificare i valori all'interno del `terraform.tfvars` file in base alle proprie esigenze.

   È necessario leggere la documentazione di supporto specifica quando si modifica uno dei valori delle variabili nel `terraform.tfvars` file. I valori possono variare in base a regione, zone di disponibilità e altri fattori supportati da Cloud Volumes ONTAP per AWS. Ciò comprende licenze, dimensioni del disco e dimensioni delle macchine virtuali per nodi singoli e coppie ha.

   Tutte le variabili di supporto per i moduli Connector e Cloud Volumes ONTAP Terraform sono già definite nel `variables.tf` file. È necessario fare riferimento ai nomi delle variabili nel `variables.tf` file prima di aggiungerlo al `terraform.tfvars` file.

5. A seconda delle proprie esigenze, è possibile attivare o disattivare FlexCache e FlexClone impostando le seguenti opzioni su `true` o `false`.

   I seguenti esempi abilitano FlexCache e FlexClone:

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

Fase 6: Implementare Cloud Volumes ONTAP per AWS

Utilizza i seguenti passaggi per implementare Cloud Volumes ONTAP per AWS.

Fasi

1. Dalla cartella principale, eseguire il comando seguente per attivare la distribuzione:

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

   Vengono attivati due container, il primo implementa Cloud Volumes ONTAP e il secondo invia dati telemetrici a AutoSupport.

   Il secondo contenitore attende fino a quando il primo non completa correttamente tutte le fasi.

2. Monitorare l'avanzamento del processo di distribuzione utilizzando i file di registro:

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

   Questo comando fornisce l'output in tempo reale e acquisisce i dati nei seguenti file di registro: `deployment.log`

   `telemetry_asup.log`

   È possibile modificare il nome di questi file di registro modificando il `.env` file utilizzando le seguenti variabili di ambiente:

   `DEPLOYMENT_LOGS`

   `TELEMETRY_ASUP_LOGS`

   Gli esempi seguenti mostrano come modificare i nomi dei file di registro:

   `DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

   `TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Al termine

Per rimuovere l'ambiente temporaneo e ripulire gli elementi creati durante il processo di distribuzione, è possibile attenersi alla seguente procedura.

Fasi

1. Se FlexCache è stato distribuito, impostare l'opzione seguente nel `terraform.tfvars` file delle variabili, in questo modo i volumi FlexCache vengono cancellati e viene rimosso l'ambiente temporaneo creato in precedenza.

   ```
   flexcache_operation = "destroy"
   ```

   Le opzioni possibili sono `deploy` e `destroy`.

2. Se FlexClone è stato distribuito, impostare l'opzione seguente nel `terraform.tfvars` file delle variabili, in questo modo i volumi FlexClone vengono cancellati e viene rimosso l'ambiente temporaneo creato in precedenza.

   ```
   flexclone_operation = "destroy"
   ```

   Le opzioni possibili sono `deploy` e `destroy`.

Cloud Volumes ONTAP per Azure

Cloud Volumes ONTAP per Azure - burst nel cloud

Questo articolo supporta la soluzione di automazione NetApp Cloud Volumes ONTAP per Azure, disponibile per i clienti NetApp nel catalogo di automazione BlueXP .

La soluzione di automazione Cloud Volumes ONTAP per Azure automatizza l'implementazione containerizzata di Cloud Volumes ONTAP per Azure utilizzando Terraform, consentendoti di implementare Cloud Volumes ONTAP per Azure rapidamente, senza interventi manuali.

Prima di iniziare

- È necessario scaricare la "Cloud Volumes ONTAP, Azure - burst nel cloud"soluzione di automazione tramite l'interfaccia utente Web di BlueXP . La soluzione viene confezionata come `CVO-Azure-Burst-To-Cloud.zip`.
- È necessario installare una macchina virtuale Linux sulla stessa rete di Cloud Volumes ONTAP.
- Dopo aver installato la VM Linux, è necessario seguire la procedura descritta in questa soluzione per installare le dipendenze richieste.

Fase 1: Installare Docker e Docker Compose

Installare Docker

I seguenti passaggi usano il software di distribuzione Linux Ubuntu 20,04 Debian come esempio. I comandi eseguiti dipendono dal software di distribuzione Linux utilizzato. Consultare la documentazione specifica del software di distribuzione Linux per la configurazione in uso.

Fasi

1. Installare Docker eseguendo i seguenti `sudo` comandi:

```
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. Verificare l’installazione:

```
docker –version
```

3. Verificare che sul sistema Linux sia stato creato un gruppo denominato "docker". Se necessario, creare il gruppo:

```
sudo groupadd docker
```

4. Aggiungere al gruppo l’utente che deve accedere a Docker:

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

5. Le modifiche vengono applicate dopo la disconnessione e la riconnessione al terminale. In alternativa, è possibile applicare immediatamente le modifiche:

```
newgrp docker
```

Installazione di Docker Compose

Fasi

1. Installare Docker Compose eseguendo i seguenti `sudo` comandi:

```
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. Verificare l’installazione:

```
docker-compose –version
```

Fase 2: Preparare l'immagine Docker

Fasi

1. Copiare la `CVO-Azure-Burst-To-Cloud.zip` cartella nella VM Linux che si desidera utilizzare per distribuire 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` è il file della chiave privata per l'accesso senza password.
   - `azureuser` È il nome utente VM.
   - `IP_ADDRESS_OF_VM` È l'indirizzo IP della macchina virtuale.
   - `LOCATION_TO_BE_COPIED` è la posizione in cui verrà copiata la cartella.

2. Estrarre la `CVO-Azure-Burst-To-Cloud.zip` cartella. È possibile estrarre la cartella nella directory corrente o in un percorso personalizzato.

   Per estrarre la cartella nella directory corrente, eseguire:

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

   Per estrarre la cartella in una posizione personalizzata, eseguire:

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

3. Dopo aver estratto il contenuto, accedere alla `CVO_Azure_Deployment` cartella ed eseguire il comando seguente per visualizzare i file:

```
 ls -la
```

   Viene visualizzato un elenco di file, simile al seguente esempio:

```
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. Individuare il `cvo_azure_flexcache_ubuntu_image_latest.tar.gz` file. Contenente l'immagine di Docker necessaria per implementare Cloud Volumes ONTAP per Azure.
5. Estrarre il file:

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

6. Attendere alcuni minuti per il caricamento dell'immagine Docker, quindi convalidare il caricamento corretto dell'immagine Docker:

```
docker images
```

Viene visualizzata un'immagine di Docker con il `latest` nome del `cvo_azure_flexcache_ubuntu_image_latest` tag, come illustrato nell'esempio seguente:

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

Passaggio 3: Creare file di variabili d'ambiente

A questo punto, è necessario creare due file di variabili d'ambiente. Un file è per l'autenticazione delle API di Azure Resource Manager utilizzando le credenziali principali del servizio. Il secondo file serve per impostare le variabili di ambiente in modo da consentire ai moduli BlueXP Terraform di individuare e autenticare le API di Azure.

Fasi

1. Creare un'identità di servizio.

Prima di poter creare i file delle variabili di ambiente, è necessario creare un'entità di servizio seguendo la procedura descritta in "Creare un'applicazione e un'entità di servizio di Azure Active Directory in grado di accedere alle risorse".

2. Assegnare il ruolo **Contributor** all'entità del servizio appena creata.
3. Creare un ruolo personalizzato.
   a. Individuare il `sp_role.json` file e verificare le autorizzazioni richieste nelle azioni elencate.
   b. Inserire queste autorizzazioni e associare il ruolo personalizzato all'entità di servizio appena creata.
4. Accedere a **certificati e segreti** e selezionare **nuovo segreto client** per creare il segreto client.

   Quando si crea il segreto client, è necessario registrare i dettagli dalla colonna **valore** perché non sarà possibile visualizzare nuovamente questo valore. È inoltre necessario registrare le seguenti informazioni:

   - ID client
   - ID abbonamento
   - ID tenant

   Queste informazioni sono necessarie per creare le variabili di ambiente. È possibile trovare informazioni sull'ID del client e sull'ID del tenant nella sezione **Panoramica** dell'interfaccia utente principale del servizio.
5. Creare i file di ambiente.
   a. Creare il `azureauth.env` file nella seguente posizione:

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

      i. Aggiungere il seguente contenuto al file:

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

      Il formato **deve** essere esattamente come mostrato sopra senza spazi tra la chiave e il valore.

   b. Creare il `credentials.env` file nella seguente posizione:

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

      i. Aggiungere il seguente contenuto al file:

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

   Il formato **deve** essere esattamente come mostrato sopra senza spazi tra la chiave e il valore.
6. Aggiungere i percorsi assoluti dei file al `.env` file.

   Immettere il percorso assoluto del `azureauth.env` file di ambiente nel `.env` file che corrisponde alla `AZURE_RM_CREDS` variabile di ambiente.

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

   Immettere il percorso assoluto del `credentials.env` file di ambiente nel `.env` file che corrisponde alla `BLUEXP_TF_AZURE_CREDS` variabile di ambiente.

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

Passaggio 4: Aggiungere licenze Cloud Volumes ONTAP a BlueXP o sottoscrivere BlueXP

Puoi aggiungere licenze Cloud Volumes ONTAP a BlueXP o iscriverti a NetApp BlueXP in Azure Marketplace.

Fasi

1. Dal portale Azure, accedere a **SaaS** e selezionare **Iscriviti a NetApp BlueXP** .
2. Selezionare il piano **Cloud Manager (di Cap PYGO per ora, WORM e servizi dati)**.

   È possibile utilizzare lo stesso gruppo di risorse di Cloud Volumes ONTAP o un gruppo di risorse diverso.

3. Configurare il portale BlueXP per importare l'abbonamento SaaS in BlueXP .

   È possibile configurare questa configurazione direttamente dal portale Azure accedendo a **Dettagli prodotto e piano** e selezionando l'opzione **Configura account ora**.

   Viene quindi eseguito il reindirizzamento al portale BlueXP per confermare la configurazione.

4. Confermare la configurazione nel portale BlueXP selezionando **Salva**.

Passaggio 5: Creare un volume esterno

È necessario creare un volume esterno per mantenere persistenti i file di stato di Terraform e altri file importanti. È necessario assicurarsi che i file siano disponibili affinché Terraform esegua il flusso di lavoro e le implementazioni.

Fasi

1. Creare un volume esterno all'esterno di Docker Compose:

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

   Esempio:

   ```
   docker volume create cvo_azure_volume_dst
   ```

2. Utilizzare una delle seguenti opzioni:

   a. Aggiungere un percorso di volume esterno al `.env` file di ambiente.

      È necessario seguire il formato esatto mostrato di seguito.

      Formato:

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

      Esempio:
      `PERSISTENT_VOL=cvo_azure_volume_dst:/cvo_azure`

   b. Aggiunta di condivisioni NFS come volume esterno.

      Assicurati che il container di Docker possa comunicare con le condivisioni NFS e che siano configurate le autorizzazioni corrette, come la lettura/scrittura.

i. Aggiungi il percorso NFS share come percorso del volume esterno nel file Docker Compose, come illustrato sotto: Format:

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

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

3. Accedere alla `cvo_azure_variables` cartella.

   Nella cartella dovrebbero essere visualizzati i seguenti file variabili:

   `terraform.tfvars`

   `variables.tf`

4. Modificare i valori all'interno del `terraform.tfvars` file in base alle proprie esigenze.

   È necessario leggere la documentazione di supporto specifica quando si modifica uno dei valori delle variabili nel `terraform.tfvars` file. I valori possono variare in base a regione, zone di disponibilità e altri fattori supportati da Cloud Volumes ONTAP per Azure. Ciò comprende licenze, dimensioni del disco e dimensioni delle macchine virtuali per nodi singoli e coppie ha.

   Tutte le variabili di supporto per i moduli Connector e Cloud Volumes ONTAP Terraform sono già definite nel `variables.tf` file. È necessario fare riferimento ai nomi delle variabili nel `variables.tf` file prima di aggiungerlo al `terraform.tfvars` file.

5. A seconda delle proprie esigenze, è possibile attivare o disattivare FlexCache e FlexClone impostando le seguenti opzioni su `true` o `false`.

   I seguenti esempi abilitano FlexCache e FlexClone:

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

6. Se necessario, è possibile recuperare il valore della variabile Terraform `az_service_principal_object_id` dal servizio Active Directory di Azure:
   a. Accedere a **applicazioni aziendali –> tutte le applicazioni** e selezionare il nome del Service Principal creato in precedenza.
   b. Copiare l'ID oggetto e inserire il valore per la variabile Terraform:

      `az_service_principal_object_id`

Fase 6: Implementare Cloud Volumes ONTAP per Azure

Per implementare Cloud Volumes ONTAP per Azure, procedere come segue.

Fasi

1. Dalla cartella principale, eseguire il comando seguente per attivare la distribuzione:

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

Vengono attivati due container, il primo implementa Cloud Volumes ONTAP e il secondo invia dati telemetrici a AutoSupport.

Il secondo contenitore attende fino a quando il primo non completa correttamente tutte le fasi.

2. Monitorare l'avanzamento del processo di distribuzione utilizzando i file di registro:

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

Questo comando fornisce l'output in tempo reale e acquisisce i dati nei seguenti file di registro:

`deployment.log`

`telemetry_asup.log`

È possibile modificare il nome di questi file di registro modificando il `.env` file utilizzando le seguenti variabili di ambiente:

`DEPLOYMENT_LOGS`

`TELEMETRY_ASUP_LOGS`

Gli esempi seguenti mostrano come modificare i nomi dei file di registro:

`DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

`TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Al termine

Per rimuovere l'ambiente temporaneo e ripulire gli elementi creati durante il processo di distribuzione, è possibile attenersi alla seguente procedura.

Fasi

1. Se FlexCache è stato distribuito, impostare l'opzione seguente nel `terraform.tfvars` file, in questo modo si puliscono i volumi FlexCache e si rimuove l'ambiente temporaneo creato in precedenza.

```
flexcache_operation = "destroy"
```

Le opzioni possibili sono `deploy` e `destroy`.

2. Se FlexClone è stato distribuito, impostare l'opzione seguente nel `terraform.tfvars` file, in questo modo si puliscono i volumi FlexClone e si rimuove l'ambiente temporaneo creato in precedenza.

```
flexclone_operation = "destroy"
```

Le opzioni possibili sono `deploy` e `destroy`.

Cloud Volumes ONTAP per Google Cloud

Cloud Volumes ONTAP per Google Cloud - burst nel cloud

Questo articolo supporta la soluzione di automazione NetApp Cloud Volumes ONTAP per Google Cloud, disponibile per i clienti NetApp nel catalogo di automazione BlueXP .

La soluzione di automazione di Cloud Volumes ONTAP per Google Cloud automatizza l'implementazione containerizzata di Cloud Volumes ONTAP per Google Cloud, consentendoti di implementare rapidamente Cloud Volumes ONTAP per Google Cloud, senza interventi manuali.

Prima di iniziare

- È necessario scaricare la "Cloud Volumes ONTAP per Google Cloud - burst nel cloud"soluzione di automazione tramite l'interfaccia utente Web di BlueXP . La soluzione viene confezionata come `cvo_gcp_flexcache.zip`.
- È necessario installare una macchina virtuale Linux sulla stessa rete di Cloud Volumes ONTAP.
- Dopo aver installato la VM Linux, è necessario seguire la procedura descritta in questa soluzione per installare le dipendenze richieste.

Fase 1: Installare Docker e Docker Compose

Installare Docker

I seguenti passaggi usano il software di distribuzione Linux Ubuntu 20,04 Debian come esempio. I comandi eseguiti dipendono dal software di distribuzione Linux utilizzato. Consultare la documentazione specifica del software di distribuzione Linux per la configurazione in uso.

Fasi

1. Installare Docker eseguendo i seguenti comandi:

```
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. Verificare l'installazione:

```
docker –version
```

3. Verificare che sul sistema Linux sia stato creato un gruppo denominato "docker". Se necessario, creare il gruppo:

```
sudo groupadd docker
```

4. Aggiungere al gruppo l'utente che deve accedere a Docker:

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

5. Le modifiche vengono applicate dopo la disconnessione e la riconnessione al terminale. In alternativa, è possibile applicare immediatamente le modifiche:

```
newgrp docker
```

Installazione di Docker Compose

Fasi

1. Installare Docker Compose eseguendo i seguenti `sudo` comandi:

```
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. Verificare l'installazione:

```
docker-compose –version
```

Fase 2: Preparare l'immagine Docker

Fasi

1. Copiare la `cvo_gcp_flexcache.zip` cartella nella VM Linux che si desidera utilizzare per distribuire 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` è il file della chiave privata per l'accesso senza password.
   - `gcpuser` È il nome utente VM.
   - `IP_ADDRESS_OF_VM` È l'indirizzo IP della macchina virtuale.
   - `LOCATION_TO_BE_COPIED` è la posizione in cui verrà copiata la cartella.

2. Estrarre la `cvo_gcp_flexcache.zip` cartella. È possibile estrarre la cartella nella directory corrente o in un percorso personalizzato.

   Per estrarre la cartella nella directory corrente, eseguire:

   ```
   unzip cvo_gcp_flexcache.zip
   ```

   Per estrarre la cartella in una posizione personalizzata, eseguire:

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

3. Dopo aver estratto il contenuto, eseguire il comando seguente per visualizzare i file:

   ```
    ls -la
   ```

   Viene visualizzato un elenco di file, simile al seguente esempio:

   ```
       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. Individuare il `cvo_gcp_flexcache_ubuntu_image.tar` file. Contenente l'immagine di Docker necessaria per implementare Cloud Volumes ONTAP per Google Cloud.
5. Estrarre il file:

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

6. Attendere alcuni minuti per il caricamento dell'immagine Docker, quindi convalidare il caricamento corretto dell'immagine Docker:

   ```
   docker images
   ```

   Viene visualizzata un'immagine di Docker con il `latest` nome del

`cvo_gcp_flexcache_ubuntu_image` tag, come illustrato nell'esempio seguente:

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

Se necessario, è possibile modificare il nome dell'immagine di Docker. In caso di modifica del nome dell'immagine di Docker, assicurarsi di aggiornare il nome dell'immagine di Docker nei `docker-compose-deploy` file e. `docker-compose-destroy`

Passaggio 3: Aggiornare il file JSON

In questa fase, è necessario aggiornare il `cxo-automation-gcp.json` file con una chiave di account di servizio per autenticare il provider Google Cloud.

1. Creare un account di servizio con autorizzazioni per distribuire Cloud Volumes ONTAP e BlueXP Connector. "Ulteriori informazioni sulla creazione di account di servizio."
2. Scaricare il file chiave per l'account e aggiornare il `cxo-automation-gcp.json` file con le informazioni del file chiave. Il `cxo-automation-gcp.json` file si trova nella `cvo_gcp_variables` cartella.

Esempio

```
{
  "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"
}
```

Il formato del file deve essere esattamente come mostrato sopra.

Passaggio 4: Sottoscrizione a BlueXP

Puoi iscriverti a NetApp BlueXP in Google Cloud Marketplace.

Fasi

1. Accedere a "Console Google Cloud" e selezionare **Subscribe to** (Iscriviti a NetApp BlueXP *).

2. Configurare il portale BlueXP per importare l'abbonamento SaaS in BlueXP .

   È possibile configurarlo direttamente da Google Cloud Platform. Verrà eseguito il reindirizzamento al portale BlueXP per confermare la configurazione.

3. Confermare la configurazione nel portale BlueXP selezionando **Salva**.

Per ulteriori informazioni, vedere "Gestire le credenziali e le sottoscrizioni di Google Cloud per BlueXP".

Passaggio 5: Abilitare le API Google Cloud richieste

Devi abilitare le seguenti API di Google Cloud nel tuo progetto per implementare Cloud Volumes ONTAP e Connector.

- API di Cloud Deployment Manager V2
- API Cloud Logging
- API Cloud Resource Manager
- API di Compute Engine
- API IAM (Identity and Access Management)

"Ulteriori informazioni sull'attivazione delle API"

Passaggio 6: Creare un volume esterno

È necessario creare un volume esterno per mantenere persistenti i file di stato di Terraform e altri file importanti. È necessario assicurarsi che i file siano disponibili affinché Terraform esegua il flusso di lavoro e le implementazioni.

Fasi

1. Creare un volume esterno all'esterno di Docker Compose:

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

   Esempio:

   ```
   docker volume create cvo_gcp_volume_dst
   ```

2. Utilizzare una delle seguenti opzioni:

   a. Aggiungere un percorso di volume esterno al `.env` file di ambiente.

      È necessario seguire il formato esatto mostrato di seguito.

      Formato:

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

      Esempio:
      `PERSISTENT_VOL=cvo_gcp_volume_dst:/cvo_gcp`

b. Aggiunta di condivisioni NFS come volume esterno.

   Assicurati che il container di Docker possa comunicare con le condivisioni NFS e che siano configurate le autorizzazioni corrette, come la lettura/scrittura.

   i. Aggiungi il percorso NFS share come percorso del volume esterno nel file Docker Compose, come illustrato sotto: Format:

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

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

3. Accedere alla `cvo_gcp_variables` cartella.

   Nella cartella dovrebbero essere visualizzati i seguenti file:

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

4. Modificare i valori all'interno del `terraform.tfvars` file in base alle proprie esigenze.

   È necessario leggere la documentazione di supporto specifica quando si modifica uno dei valori delle variabili nel `terraform.tfvars` file. I valori possono variare in base a regione, zone di disponibilità e altri fattori supportati da Cloud Volumes ONTAP per Google Cloud. Ciò comprende licenze, dimensioni del disco e dimensioni delle macchine virtuali per nodi singoli e coppie ha.

   Tutte le variabili di supporto per i moduli Connector e Cloud Volumes ONTAP Terraform sono già definite nel `variables.tf` file. È necessario fare riferimento ai nomi delle variabili nel `variables.tf` file prima di aggiungerlo al `terraform.tfvars` file.

5. A seconda delle proprie esigenze, è possibile attivare o disattivare FlexCache e FlexClone impostando le seguenti opzioni su `true` o `false`.

   I seguenti esempi abilitano FlexCache e FlexClone:

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

Fase 7: Implementare Cloud Volumes ONTAP per Google Cloud

Utilizza i seguenti passaggi per implementare Cloud Volumes ONTAP per Google Cloud.

Fasi

1. Dalla cartella principale, eseguire il comando seguente per attivare la distribuzione:

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

   Vengono attivati due container, il primo implementa Cloud Volumes ONTAP e il secondo invia dati telemetrici a AutoSupport.

Il secondo contenitore attende fino a quando il primo non completa correttamente tutte le fasi.

2. Monitorare l'avanzamento del processo di distribuzione utilizzando i file di registro:

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

Questo comando fornisce l'output in tempo reale e acquisisce i dati nei seguenti file di registro: `deployment.log`

`telemetry_asup.log`

È possibile modificare il nome di questi file di registro modificando il `.env` file utilizzando le seguenti variabili di ambiente:

`DEPLOYMENT_LOGS`

`TELEMETRY_ASUP_LOGS`

Gli esempi seguenti mostrano come modificare i nomi dei file di registro:

`DEPLOYMENT_LOGS=<your_deployment_log_filename>.log`

`TELEMETRY_ASUP_LOGS=<your_telemetry_asup_log_filename>.log`

Al termine

Per rimuovere l'ambiente temporaneo e ripulire gli elementi creati durante il processo di distribuzione, è possibile attenersi alla seguente procedura.

Fasi

1. Se FlexCache è stato distribuito, impostare l'opzione seguente nel `terraform.tfvars` file, in questo modo si puliscono i volumi FlexCache e si rimuove l'ambiente temporaneo creato in precedenza.

```
flexcache_operation = "destroy"
```

Le opzioni possibili sono `deploy` e `destroy`.

2. Se FlexClone è stato distribuito, impostare l'opzione seguente nel `terraform.tfvars` file, in questo modo si puliscono i volumi FlexClone e si rimuove l'ambiente temporaneo creato in precedenza.

```
flexclone_operation = "destroy"
```

Le opzioni possibili sono `deploy` e `destroy`.

ONTAP

Giorno 0/1

Panoramica della soluzione ONTAP Day 0/1

Puoi utilizzare la soluzione di automazione ONTAP Day 0/1 per implementare e configurare un cluster ONTAP utilizzando Ansible. La soluzione è disponibile nella "Catalogo di automazione BlueXP".

Opzioni flessibili di implementazione ONTAP

A seconda dei tuoi requisiti, puoi utilizzare l'hardware on-premise o simulare ONTAP per implementare e configurare un cluster ONTAP utilizzando Ansible.

Hardware on-premise

Puoi implementare questa soluzione utilizzando hardware on-premise che esegue ONTAP, come un FAS o un sistema AFF. Devi utilizzare una macchina virtuale Linux per implementare e configurare il cluster ONTAP utilizzando Ansible.

Simula ONTAP

Per implementare questa soluzione utilizzando un simulatore ONTAP, è necessario scaricare la versione più recente di simulate ONTAP dal sito di supporto NetApp. Simulate ONTAP è un simulatore virtuale per il software ONTAP. Simulate ONTAP viene eseguito in un hypervisor VMware su un sistema Windows, Linux o Mac. Per gli host Windows e Linux, è necessario utilizzare l'hypervisor VMware Workstation per eseguire questa soluzione. Se si dispone di un sistema operativo Mac, utilizzare l'hypervisor VMware Fusion.

Design a più strati

Il framework Ansible semplifica lo sviluppo e il riutilizzo dei task logici e dell'esecuzione dell'automazione. Il framework distingue tra le attività decisionali (livello logico) e le fasi di esecuzione (livello di esecuzione) nell'automazione. La comprensione del funzionamento di questi livelli consente di personalizzare la configurazione.

Un "playbook" Ansible esegue una serie di task dall'inizio alla fine. La `site.yml` guida contiene la `logic.yml` guida e la `execution.yml` guida.

Quando viene eseguita una richiesta, il `site.yml` playbook viene chiamato per primo il `logic.yml` playbook, quindi chiama il `execution.yml` playbook per eseguire la richiesta di servizio.

Non è necessario utilizzare il livello logico del framework. Il livello logico fornisce opzioni per espandere la capacità del framework oltre i valori hard-coded per l'esecuzione. Ciò consente di personalizzare le funzionalità del framework, se necessario.

Livello logico

Il livello logico è costituito dai seguenti elementi:

- `logic.yml`Il manuale
- File di operazioni logiche all'interno della `logic-tasks` directory

Il livello logico offre la possibilità di prendere decisioni complesse senza la necessità di una significativa integrazione personalizzata (ad esempio, la connessione a ServiceNOW). Il livello logico è configurabile e fornisce l'ingresso ai microservizi.

È inoltre prevista la capacità di bypassare il livello logico. Se si desidera ignorare il livello logico, non definire la `logic_operation` variabile. La invocazione diretta del `logic.yml` playbook offre la possibilità di effettuare qualche livello di debug senza esecuzione. È possibile utilizzare un'istruzione "debug" per verificare che il

valore di `raw_service_request` sia corretto.

Considerazioni importanti:

- Il `logic.yml` playbook ricerca la `logic_operation` variabile. Se la variabile è definita nella richiesta, carica un file di attività dalla `logic-tasks` directory. Il file di attività deve essere un file .yml. Se non esiste un file di attività corrispondente e la `logic_operation` variabile è definita, il livello logico non riesce.
- Il valore predefinito della `logic_operation` variabile è `no-op`. Se la variabile non è definita in modo esplicito, per impostazione predefinita è `no-op`, che non esegue alcuna operazione.
- Se la `raw_service_request` variabile è già definita, l'esecuzione procede al livello di esecuzione. Se la variabile non è definita, il livello logico non riesce.

Livello di esecuzione

Il livello di esecuzione è costituito dai seguenti elementi:

- `execution.yml`Il manuale

Il livello di esecuzione effettua le chiamate API per configurare un cluster ONTAP. Il `execution.yml` playbook richiede che la `raw_service_request` variabile sia definita al momento dell'esecuzione.

Supporto per la personalizzazione

È possibile personalizzare questa soluzione in vari modi a seconda delle proprie esigenze.

Le opzioni di personalizzazione includono:

- Modifica dei playbook Ansible
- Aggiunta di ruoli

Personalizza i file Ansible

La tabella seguente descrive i file Ansible personalizzabili contenuti in questa soluzione.

Posizione	Descrizione
`playbooks/inventory/hosts`	Contiene un singolo file con un elenco di host e gruppi.
`playbooks/group_vars/all/*`	Ansible fornisce un modo pratico per applicare le variabili a più host contemporaneamente. È possibile modificare uno o tutti i file contenuti in questa cartella, inclusi `cfg.yml` `clusters.yml` , `defaults.yml`, `services.yml`, `standards.yml` e `vault.yml`.
`playbooks/logic-tasks`	Supporta le attività decisionali all'interno di Ansible e mantiene la separazione di logica ed esecuzione. È possibile aggiungere file a questa cartella che corrispondono al servizio pertinente.
`playbooks/vars/*`	Valori dinamici utilizzati nei playbook e nei ruoli Ansible per consentire la personalizzazione, la flessibilità e la riutilizzabilità delle configurazioni. Se necessario, è possibile modificare uno o tutti i file contenuti in questa cartella.

Personalizzare i ruoli

Puoi anche personalizzare la soluzione aggiungendo o cambiando ruoli Ansible, anche chiamati microservizi.

Per ulteriori informazioni, vedere "Personalizza".

Preparare l'uso della soluzione ONTAP Day 0/1

Prima di implementare la soluzione di automazione, devi preparare l'ambiente ONTAP e installare e configurare Ansible.

Considerazioni iniziali di pianificazione

È necessario analizzare i requisiti e le considerazioni seguenti prima di utilizzare questa soluzione per implementare un cluster ONTAP.

Requisiti di base

Per utilizzare questa soluzione è necessario soddisfare i seguenti requisiti di base:

- Devi avere accesso al software ONTAP on-premise o tramite un simulatore ONTAP.
- È necessario sapere come utilizzare il software ONTAP.
- Devi sapere come utilizzare gli strumenti software di automazione Ansible.

Considerazioni sulla pianificazione

Prima di implementare questa soluzione di automazione, è necessario decidere:

- Posizione in cui eseguire il nodo di controllo Ansible.
- Sistema ONTAP (hardware on-premise) o simulatore ONTAP.
- Se è necessario personalizzare o meno.

Preparare il sistema ONTAP

Utilizzando un sistema ONTAP on-premise o simulando ONTAP, devi preparare l'ambiente prima di poter implementare la soluzione di automazione.

Facoltativamente, installare e configurare simulate ONTAP

Per implementare questa soluzione attraverso un simulatore di ONTAP, è necessario scaricare ed eseguire simulate ONTAP.

Prima di iniziare

- È necessario scaricare e installare l'hypervisor VMware che si intende utilizzare per eseguire simulate ONTAP.
  - Se si dispone di un sistema operativo Windows o Linux, utilizzare VMware Workstation.
  - Se si dispone di un sistema operativo Mac, utilizzare VMware Fusion.

> Se si utilizza un sistema operativo Mac OS, è necessario disporre di un processore Intel.

Fasi

Per installare due simulatori ONTAP nell'ambiente locale, attenersi alla procedura seguente:

1. Scaricare simulate ONTAP dal "Sito di supporto NetApp".

Anche se si installano due simulatori ONTAP, è sufficiente scaricare una sola copia del software.

2. Se non è già in esecuzione, avviare l'applicazione VMware.
3. Individuare il file del simulatore scaricato e fare clic con il pulsante destro del mouse per aprirlo con l'applicazione VMware.
4. Impostare il nome della prima istanza di ONTAP.
5. Attendere l'avvio del simulatore e seguire le istruzioni per creare un cluster a nodo singolo.

   Ripetere la procedura per la seconda istanza di ONTAP.

6. In alternativa, è possibile aggiungere un complemento di dischi completo.

   Da ciascun cluster, eseguire i seguenti comandi:

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

Stato del sistema ONTAP

Devi verificare lo stato iniziale del sistema ONTAP, sia on-premise che in esecuzione attraverso un simulatore ONTAP.

Verificare che siano soddisfatti i seguenti requisiti di sistema ONTAP:

- ONTAP è installato e in esecuzione senza cluster ancora definiti.
- ONTAP viene avviato e visualizza l'indirizzo IP per accedere al cluster.
- La rete è raggiungibile.
- Si dispone delle credenziali di amministratore.
- Viene visualizzato il banner del messaggio del giorno (MOTD) con l'indirizzo di gestione.

Installare il software di automazione richiesto

Questa sezione fornisce informazioni su come installare Ansible e preparare la soluzione di automazione per l'implementazione.

Installa Ansible

Ansible può essere installato su sistemi Linux o Windows.

Il metodo di comunicazione predefinito utilizzato da Ansible per comunicare con un cluster ONTAP è SSH.

Fare riferimento a "Introduzione a NetApp e Ansible: Installare Ansible" per installare Ansible.

Ansible deve essere installato sul nodo di controllo del sistema.

Scaricare e preparare la soluzione di automazione

Per scaricare e preparare la soluzione di automazione per la distribuzione, è possibile attenersi alla seguente procedura.

1. Scaricare la "ONTAP - giorno 0/1 controlli dello stato"soluzione di automazione tramite l'interfaccia utente Web di BlueXP . La soluzione viene confezionata come `ONTAP_DAY0_DAY1.zip`.
2. Estrarre la cartella zip e copiare i file nella posizione desiderata sul nodo di controllo all'interno dell'ambiente Ansible.

Configurazione iniziale del framework Ansible

Eseguire la configurazione iniziale del framework Ansible:

1. Passare a `playbooks/inventory/group_vars/all`.
2. Decrittografare il `vault.yml` file:

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

   Quando viene richiesta la password del vault, immettere la seguente password temporanea:

   ```
   NetApp123!
   ```

   > "NetApp123!" è una password temporanea per decrittografare il `vault.yml` file e la password del vault corrispondente. Dopo il primo utilizzo, è **necessario** crittografare il file utilizzando la propria password.

3. Modificare i seguenti file Ansible:

   - `clusters.yml` - Modificare i valori in questo file per adattarli all'ambiente.
   - `vault.yml` - Dopo aver decrittografato il file, modificare i valori del cluster ONTAP, del nome utente e della password in base all'ambiente in uso.
   - `cfg.yml` - Impostare il percorso del file per `log2file` e impostare `show_request` in `cfg` a `True` per visualizzare `raw_service_request` .

     La `raw_service_request` variabile viene visualizzata nei file di registro e durante l'esecuzione.

   > Ogni file elencato contiene commenti con istruzioni su come modificarlo in base alle proprie esigenze.

4. Crittografare nuovamente il `vault.yml` file:

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

   > Viene richiesto di scegliere una nuova password per il vault al momento della crittografia.

5. Navigare `playbooks/inventory/hosts` e impostare un interprete Python valido.
6. Implementare il `framework_test` servizio:

Il seguente comando esegue il `na_ontap_info` modulo con un `gather_subset` valore di `cluster_identity_info` . In questo modo, la configurazione di base risulta corretta e si verifica la possibilità di comunicare con il cluster.

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

Eseguire il comando per ciascun cluster.

Se l'operazione ha esito positivo, si dovrebbe visualizzare un output simile al seguente esempio:

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

Implementare il cluster ONTAP utilizzando la soluzione

Dopo aver completato la preparazione e il planning, sei pronto a utilizzare la soluzione ONTAP Day 0/1 per configurare rapidamente un cluster ONTAP utilizzando Ansible.

In qualsiasi momento durante le fasi di questa sezione, è possibile scegliere di testare una richiesta invece di eseguirla. Per testare una richiesta, modificare il `site.yml` playbook sulla riga di comando in `logic.yml`.

> La `docs/tutorial-requests.txt` posizione contiene la versione finale di tutte le richieste di servizio utilizzate durante questa procedura. In caso di difficoltà nell'esecuzione di una richiesta di servizio, è possibile copiare la richiesta pertinente dal `tutorial-requests.txt` file nella `playbooks/inventory/group_vars/all/tutorial-requests.yml` posizione e modificare i valori codificati come richiesto (indirizzo IP, nomi aggregati e così via). A questo punto, è possibile eseguire correttamente la richiesta.

Prima di iniziare

- È necessario che Ansible sia installato.
- È necessario aver scaricato la soluzione ONTAP Day 0/1 ed estratto la cartella nella posizione desiderata sul nodo di controllo Ansible.
- Lo stato del sistema ONTAP deve soddisfare i requisiti e l'utente deve disporre delle credenziali necessarie.
- È necessario aver completato tutte le attività richieste indicate nella "Preparatevi"sezione .

> Negli esempi di questa soluzione vengono utilizzati "Cluster_01" e "Cluster_02" come nomi per i due cluster. È necessario sostituire questi valori con i nomi dei cluster nel proprio ambiente.

Fase 1: Configurazione iniziale del cluster

A questo punto, è necessario eseguire alcune operazioni iniziali di configurazione del cluster.

Fasi

1. Individuare la `playbooks/inventory/group_vars/all/tutorial-requests.yml` posizione e rivedere la `cluster_initial` richiesta nel file. Apportare le modifiche necessarie al proprio ambiente.
2. Creare un file nella `logic-tasks` cartella per la richiesta di servizio. Ad esempio, creare un file denominato `cluster_initial.yml`.

   Copiare le seguenti righe nel nuovo file:

   ```
   - 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. Definire la `raw_service_request` variabile.

   È possibile utilizzare una delle seguenti opzioni per definire la `raw_service_request` variabile nel `cluster_initial.yml` file creato nella `logic-tasks` cartella:

   - **Opzione 1**: Definire manualmente la `raw_service_request` variabile.

     Aprire il `tutorial-requests.yml` file utilizzando un editor e copiare il contenuto dalla riga 11 alla riga 165. Incollare il contenuto sotto la `raw service request` variabile nel nuovo `cluster_initial.yml` file, come illustrato negli esempi seguenti:


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

Mostra esempio

File di esempio `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 }}"
```

- **Opzione 2**: Utilizzare un modello Jinja per definire la richiesta:

  È anche possibile utilizzare il seguente formato di modello Jinja per ottenere il `raw_service_request` valore.

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

4. Eseguire la configurazione iniziale del cluster per il primo cluster:

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

Prima di procedere, verificare che non vi siano errori.

5. Ripetere il comando per il secondo cluster:

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

Verificare che non siano presenti errori per il secondo cluster.

Quando scorri verso l'alto verso l'inizio dell'output Ansible dovresti vedere la richiesta inviata al framework, come mostrato nel seguente esempio:

Mostra esempio

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

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

6. Accedere a ciascuna istanza di ONTAP e verificare che la richiesta sia stata eseguita correttamente.

Fase 2: Configurare intercluster LIF

Ora puoi configurare i LIF intercluster LIF aggiungendo le definizioni LIF alla `cluster_initial` richiesta e definendo il `ontap_interface` microservizio.

La definizione del servizio e la richiesta lavorano insieme per determinare l'azione:

- Se si fornisce una richiesta di servizio per un microservizio che non è presente nelle definizioni di servizio, la richiesta non viene eseguita.
- Se si fornisce una richiesta di servizio con uno o più microservizi definiti nelle definizioni di servizio, ma omessi dalla richiesta, la richiesta non viene eseguita.

Il `execution.yml` playbook valuta la definizione del servizio analizzando l'elenco dei microservizi nell'ordine elencato:

- Se nella richiesta è presente una voce con una chiave dizionario corrispondente alla `args` voce contenuta nelle definizioni di microservizi, la richiesta viene eseguita.
- Se nella richiesta di servizio non è presente alcuna voce corrispondente, la richiesta viene ignorata senza errori.

Fasi

1. Passare al `cluster_initial.yml` file creato in precedenza e modificare la richiesta aggiungendo le seguenti righe alle definizioni della richiesta:

```
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. Eseguire il comando:

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

3. Effettua l'accesso a ciascuna istanza per verificare se le LIF sono state aggiunte al cluster:

Mostra esempio

```
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.
```

Il risultato mostra che le LIF sono state **non** aggiunte. Questo perché il `ontap_interface` microservizio deve ancora essere definito nel `services.yml` file.

4. Verificare che le LIF siano state aggiunte alla `raw_service_request` variabile.

Mostra esempio

Il seguente esempio mostra che le LIF sono state aggiunte alla richiesta:


```
            "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. Definire il `ontap_interface` microservizio in `cluster_initial` nel `services.yml` file.

   Copiare le seguenti righe nel file per definire il microservizio:

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

6. Ora che il `ontap_interface` microservizio è stato definito nella richiesta e nel `services.yml` file, eseguire nuovamente la richiesta:

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

7. Accedere a ciascuna istanza di ONTAP e verificare che le LIF siano state aggiunte.

Fase 3: In alternativa, configurare più cluster

Se necessario, puoi configurare più cluster nella stessa richiesta. Quando si definisce la richiesta, è necessario fornire i nomi delle variabili per ciascun cluster.

Fasi

1. Aggiungere una voce per il secondo cluster nel `cluster_initial.yml` file per configurare entrambi i cluster nella stessa richiesta.

   Nell'esempio seguente viene visualizzato il `ontap_aggr` campo dopo l'aggiunta della seconda voce.

```
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. Applicare le modifiche per tutti gli altri elementi in `cluster_initial`.
3. Aggiungere il peering dei cluster alla richiesta copiando le seguenti righe nel file:

```
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. Eseguire la richiesta 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>
```

Fase 4: Configurazione SVM iniziale

In questa fase della procedura è necessario configurare le SVM nel cluster.

Fasi

1. Aggiornare la `svm_initial` richiesta nel `tutorial-requests.yml` file per configurare un peer relationship SVM e SVM.

   È necessario configurare quanto segue:

   - SVM

- La relazione peer della SVM
- L'interfaccia SVM per ciascuna SVM

2. Aggiornare le definizioni delle variabili nelle definizioni delle `svm_initial` richieste. È necessario modificare le seguenti definizioni di variabile:
   - `cluster_name`
   - `vserver_name`
   - `peer_cluster_name`
   - `peer_vserver`

   Per aggiornare le definizioni, rimuovere il simbolo * '{}'* dopo `req_details` per la `svm_initial` definizione e aggiungere la definizione corretta.

3. Creare un file nella `logic-tasks` cartella per la richiesta di servizio. Ad esempio, creare un file denominato `svm_initial.yml`.

   Copiare le seguenti righe nel file:

```
- 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. Definire la `raw_service_request` variabile.

   È possibile utilizzare una delle seguenti opzioni per definire la `raw_service_request` variabile `svm_initial` nella `logic-tasks` cartella:

   - **Opzione 1**: Definire manualmente la `raw_service_request` variabile.

     Aprire il `tutorial-requests.yml` file utilizzando un editor e copiare il contenuto dalla riga 179 alla

riga 222. Incollare il contenuto sotto la `raw service request` variabile nel nuovo `svm_initial.yml` file, come illustrato negli esempi seguenti:

Mostra esempio

File di esempio `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
```

- **Opzione 2**: Utilizzare un modello Jinja per definire la richiesta:

  È anche possibile utilizzare il seguente formato di modello Jinja per ottenere il `raw_service_request` valore.

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

5. Eseguire la richiesta:

```
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. Accedere a ciascuna istanza di ONTAP e convalidare la configurazione.
7. Aggiungere le interfacce della SVM.

   Definire il `ontap_interface` servizio in `svm_initial` nel `services.yml` file ed eseguire nuovamente la richiesta:

```
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. Effettuare l'accesso a ciascuna istanza di ONTAP e verificare che le interfacce della SVM siano state configurate.

Fase 5: Se si desidera, definire una richiesta di servizio in modo dinamico

Nei passi precedenti, la `raw_service_request` variabile è codificata. Ciò è utile per l'apprendimento, lo sviluppo e il test. È inoltre possibile generare dinamicamente una richiesta di servizio.

La sezione seguente fornisce un'opzione per produrre dinamicamente il necessario `raw_service_request` se non si desidera integrarlo con sistemi di livello superiore.

- Se la `logic_operation` variabile non è definita nel comando, il `logic.yml` file non importa alcun file dalla `logic-tasks` cartella. Ciò significa che i `raw_service_request` devono essere definiti all'esterno di Ansible e forniti al framework al momento dell'esecuzione.
- Il nome del file di un'operazione nella `logic-tasks` cartella deve corrispondere al valore della `logic_operation` variabile senza estensione .yml.
- I file di attività nella `logic-tasks` cartella definiscono dinamicamente un `raw_service_request`. l'unico requisito è che un valido `raw_service_request` sia definito come l'ultima attività nel file pertinente.

Definizione dinamica di una richiesta di servizio

Esistono diversi modi per applicare un'attività logica per definire dinamicamente una richiesta di servizio. Di seguito sono elencate alcune di queste opzioni:

- Utilizzo di un file attività Ansible dalla `logic-tasks` cartella
- Richiamo di un ruolo personalizzato che restituisce dati adatti alla conversione in un ruolo `raw_service_request` variabile.
- Richiamo di un altro strumento all'esterno dell'ambiente Ansible per i dati richiesti. Ad esempio, una chiamata API REST a Active IQ Unified Manager.

I seguenti comandi di esempio definiscono dinamicamente una richiesta di servizio per ogni cluster utilizzando il `tutorial-requests.yml` file:

```
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
```

Fase 6: Distribuire la soluzione ONTAP Day 0/1

In questa fase, dovresti aver già completato quanto segue:

- Revisionato e modificato tutti i file in in `playbooks/inventory/group_vars/all` base alle proprie esigenze. Ogni file contiene commenti dettagliati che consentono di apportare le modifiche.
- Aggiunti tutti i file di attività richiesti alla `logic-tasks` directory.
- Aggiunti tutti i file di dati necessari alla `playbook/vars` directory.

Utilizzare i seguenti comandi per implementare la soluzione ONTAP Day 0/1 e verificare lo stato di salute della distribuzione:

> In questa fase, il file dovrebbe essere già stato decrittografato e modificato `vault.yml` e deve essere crittografato con la nuova password.

- Eseguire il servizio 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>
```

- Eseguire il servizio 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>
```

- Applicare le impostazioni a livello di cluster:

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

- Eseguire i controlli dello stato di salute:

```
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>
```

Personalizzare la soluzione ONTAP Day 0/1

Per personalizzare la soluzione ONTAP Day 0/1 in base ai tuoi requisiti, puoi aggiungere o modificare i ruoli Ansible.

I ruoli rappresentano i microservizi all'interno del framework Ansible. Ogni microservizio esegue un'operazione. Ad esempio, ONTAP Day 0 è un servizio che contiene più microservizi.

Aggiungi ruoli Ansible

Puoi aggiungere ruoli Ansible per personalizzare la soluzione per il tuo ambiente. I ruoli richiesti sono definiti dalle definizioni dei servizi all'interno del framework Ansible.

Un ruolo deve soddisfare i seguenti requisiti per essere utilizzato come microservizio:

- Accettare un elenco di argomenti nella `args` variabile.
- Utilizza la struttura Ansible "Block, rescue, Always" con determinati requisiti per ogni blocco.
- Utilizza un singolo modulo Ansible e definisci un singolo task all'interno del blocco.
- Implementare tutti i parametri del modulo disponibili in base ai requisiti descritti in questa sezione.

Struttura di microservizio richiesta

Ogni ruolo deve supportare le seguenti variabili:

- `mode`: Se la modalità è impostata sul `test` ruolo tenta di importare il `test.yml` che mostra cosa fa il ruolo senza eseguirlo.

  Non è sempre possibile implementare questo processo a causa di alcune interdipendenze.

- `status`: Lo stato generale dell'esecuzione del playbook. Se il valore non è impostato `success` sul ruolo non viene eseguito.
- `args` : Elenco di dizionari specifici per ruolo con chiavi che corrispondono ai nomi dei parametri del ruolo.
- `global_log_messages`: Raccoglie i messaggi di registro durante l'esecuzione del playbook. Ogni volta che viene eseguito il ruolo viene generata una voce.
- `log_name`: Il nome utilizzato per fare riferimento al ruolo all'interno delle `global_log_messages` voci.
- `task_descr`: Una breve descrizione delle funzioni del ruolo.
- `service_start_time`: La data e l'ora utilizzate per tenere traccia dell'ora di esecuzione di ciascun ruolo.
- `playbook_status`: Lo stato del playbook Ansible.
- `role_result`: La variabile che contiene l'output del ruolo ed è inclusa in ogni messaggio all'interno delle `global_log_messages` voci.

Esempio di struttura dei ruoli

Nell'esempio seguente viene fornita la struttura di base di un ruolo che implementa un microservizio. È necessario modificare le variabili in questo esempio per la propria configurazione.

Mostra esempio

Struttura dei ruoli di base:

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

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

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

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

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

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

```
            <MODULE_SPECIFIC_PARAMETERS>

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

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

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

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

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

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

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

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

Variabili utilizzate nel ruolo di esempio:

- `<NAME>`: Un valore sostituibile che deve essere fornito per ogni microservizio.
- `<LOG_NAME>`: Il nome breve del ruolo utilizzato per la registrazione. Ad esempio, `ONTAP_VOLUME`.
- `<TASK_DESCRIPTION>`: Una breve descrizione delle funzioni del microservizio.
- `<MODULE_NAME>`: Il nome del modulo Ansible per l'attività.

> Il playbook di livello superiore `execute.yml` specifica la `netapp.ontap` raccolta. Se il modulo fa parte dell' `netapp.ontap`insieme, non è necessario specificare completamente il nome del modulo.

- `<MODULE_SPECIFIC_PARAMETERS>`: Parametri del modulo Ansible specifici del modulo utilizzato per implementare il microservizio. Nell'elenco seguente vengono descritti i tipi di parametri e le relative modalità di raggruppamento.
  - Parametri richiesti: Tutti i parametri richiesti sono specificati senza alcun valore predefinito.
  - Parametri che hanno un valore predefinito specifico per il microservizio (non uguale a un valore predefinito specificato nella documentazione del modulo).
  - Tutti i parametri rimanenti utilizzano `default(omit)` come valore predefinito.

Utilizzo di dizionari multilivello come parametri del modulo

Alcuni moduli Ansible forniti da NetApp utilizzano dizionari multi-livello per i parametri dei moduli (ad esempio gruppi di policy QoS fissi e adattivi).

L'uso `default(omit)` da solo non funziona quando si utilizzano questi dizionari, specialmente quando ne esistono più di uno e si escludono a vicenda.

Se è necessario utilizzare dizionari multilivello come parametri del modulo, è necessario suddividere la funzionalità in più microservizi (ruoli) in modo che ciascuno di essi possa fornire almeno un valore del dizionario di secondo livello per il dizionario pertinente.

Gli esempi seguenti mostrano gruppi di criteri QoS fissi e adattivi suddivisi in due microservizi.

Il primo microservizio contiene valori di gruppo di criteri QoS fissi:

```
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)
}}"
```

Il secondo microservizio contiene i valori dei gruppi di criteri QoS adattivi:

```
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) }}"
```

API dei prodotti NetApp

ONTAP 9

NetApp ONTAP è il software per la gestione dei dati leader del settore per implementazioni cloud e on-premise. ONTAP include un'unica API REST comune che continua a essere espansa e migliorata con ogni release. Consulta la documentazione e le risorse correlate fornite di seguito per iniziare con l'automazione delle implementazioni di ONTAP utilizzando l'API REST ONTAP.

API REST di ONTAP

Puoi utilizzare l'API REST per automatizzare l'amministrazione delle implementazioni ONTAP.

- "Documentazione sull'automazione ONTAP"

Famiglia ONTAP

La documentazione della linea di prodotti ONTAP include tutto il necessario per installare e amministrare le implementazioni di ONTAP.

- "Documentazione del prodotto ONTAP"

Piano di controllo BlueXP

NetApp BlueXP BlueXP è un pannello di controllo unificato che offre una piattaforma multicloud ibrida per l'amministrazione dei servizi dati e dello storage in ambienti cloud pubblici e on-premise. È composto da diversi servizi o componenti distinti, ciascuno dei quali espone un'API REST associata. Consulta la documentazione e le risorse correlate fornite di seguito per iniziare con l'automazione del tuo ambiente BlueXP utilizzando le API REST BlueXP .

API REST BlueXP

Puoi utilizzare le varie API REST per automatizzare l'amministrazione dello storage e dei servizi dati gestiti da BlueXP .

- "Documentazione dell'API BlueXP"

Famiglia BlueXP

La documentazione della famiglia BlueXP include tutto ciò che serve per iniziare a utilizzare il piano di controllo BlueXP .

- "Documentazione BlueXP"

Controllo Astra

Astra Control è un software NetApp che offre una gestione dei dati integrata con l'applicazione dei cluster Kubernetes in più ambienti. I due modelli di implementazione condividono un'API REST comune. Consulta la documentazione e le risorse correlate fornite di seguito per iniziare con l'automazione delle implementazioni Astra utilizzando

l'API REST Astra Control.

API REST di Astra Control

Puoi utilizzare le API REST per automatizzare l'amministrazione delle implementazioni di Astra Control Service e Astra Control Center.

- "Documentazione di Astra Control Automation"

Famiglia Astra

La documentazione della linea Astra include tutto il necessario per installare e amministrare Astra e il relativo software.

- "Documentazione Astra"

Active IQ Unified Manager

Active IQ Unified Manager (in precedenza denominato OnCommand Unified Manager) offre gestione e monitoring completi dei sistemi ONTAP. Include inoltre un'API REST per automatizzare queste attività e consentire l'integrazione di terze parti sui sistemi supportati.

Vantaggi dell'API REST

L'utilizzo dell'API REST fornita con Active IQ Unified Manager comporta diversi vantaggi.

Funzionalità solide

Utilizzando l'API REST puoi accedere alle funzionalità di Active IQ Unified Manager per gestire la disponibilità dello storage, la capacità, la sicurezza, la protezione e i rischi di performance.

Consolidamento dell'automazione

Esiste un singolo endpoint REST disponibile per eseguire il provisioning e gestire i workload. Ciò fornisce un semplice approccio consolidato per implementare le policy degli obiettivi del livello di servizio, reindirizzare gli eventi a strumenti di terze parti e agire come gateway API per accedere all'API REST ONTAP a livello di singolo cluster.

Automazione a livello di data center della gestione ONTAP

È possibile automatizzare i flussi di lavoro di gestione e provisioning di ONTAP a livello di data center. È possibile automatizzare anche il monitoraggio e la creazione di report. Questo migliora l'efficienza e fornisce una base per l'aggregazione a livello di data center.

Ulteriori informazioni

Esistono diverse risorse disponibili per iniziare con l'API REST di Active IQ Unified Manager.

- "Introduzione alle API REST di Active IQ Unified Manager"
- "Documentazione dell'API Active IQ Unified Manager"
- "Moduli NetApp per Ansible"

Conoscenza e supporto

Risorse aggiuntive

Esistono ulteriori risorse a cui puoi accedere per ottenere assistenza e trovare ulteriori informazioni sui prodotti NetApp e sui servizi cloud.

Risorse per sviluppatori NetApp

- "DevNet di NetApp"

  Una posizione centrale per le risorse per gli sviluppatori che supportano i partner e i clienti NetApp.

- "NetApp io - il Pub"

  Articoli di blog e altre risorse per supportare sviluppatori e amministratori.

Risorse cloud di NetApp

- "NetApp BlueXP"

  Sito centrale per le soluzioni cloud di NetApp.

- "Console NetApp Cloud Central"

  Console di servizio NetApp Cloud Central con accesso.

Richiedi assistenza

NetApp fornisce supporto per i propri prodotti e servizi cloud in vari modi. Sono disponibili opzioni complete di supporto autonomo gratuito 24 ore su 24, 7 giorni su 7, come articoli della knowledge base (KB) e un forum della community.

Opzioni di supporto automatico

- "Knowledge base"

  Ricerca nella knowledge base per trovare articoli utili per la risoluzione dei problemi.

- "Community"

  Unisciti a una community NetApp per seguire le discussioni in corso o crearne di nuove.

- "Supporto NetApp"

  Accesso a strumenti per la risoluzione dei problemi, documentazione e assistenza tecnica.

Note legali

Le note legali forniscono l'accesso a dichiarazioni di copyright, marchi, brevetti e altro ancora.

Copyright

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

Marchi

NETAPP, il logo NETAPP e i marchi elencati nella pagina dei marchi NetApp sono marchi di NetApp, Inc. Altri nomi di società e prodotti potrebbero essere marchi dei rispettivi proprietari.

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

Brevetti

Un elenco aggiornato dei brevetti di proprietà di NetApp è disponibile all'indirizzo:

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

Direttiva sulla privacy

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

Open source

I file di avviso forniscono informazioni sul copyright e sulle licenze di terze parti utilizzate nel software NetApp.